Cutting Edge Robotics

Cutting Edge Robotics

Edited by

Vedran Kordic

Aleksandar Lazinica

Munir Merdan

pro literatur Verlag

Published by the plV pro literatur Verlag Robert Mayer-Scholz

plV pro literatur Verlag Robert Mayer-Scholz
Mammendorf
Germany

Abstracting and non-profit use of the material is permitted with credit to the source. Statements and opinions expressed in the chapters are these of the individual contributors and not necessarily those of the editors or publisher. No responsibility is accepted for the accuracy of information contained in the published articles. Publisher assumes no responsibility liability for any damage or injury to persons or property arising out of the use of any materials, instructions, methods or ideas contained inside. After this work has been published by the Advanced Robotic Systems International, authors have the right to republish it, in whole or part, in any publication of which they are an author or editor, and the make other personal use of the work.

www.ars-journal.com
Additional copies can be obtained from:
publication@ars-journal.com

First published 2005

Typeface Book Anitqua 10/14 pt.

Printed in Croatia

A catalog record for this book is available from the German Library.

Cutting Edge Robotics / edited by Vedran Kordic, Aleksandar Lazinica, Munir Merdan.
p. cm.
ISBN 3-86611-038-3
1. Robotics. 2. Artificial Intelligence. 3. Mechatronics I. Kordic, Vedran. II. Lazinica, Aleksandar. III. Merdan, Munir

Contents

III. Navigation

IV. Adaptive and Learning Systems

V. Multi-Robot Systems

VI. Human-Robot Interaction

VII. Service Robotics

VIII. Legged Robots

IX. Robot Manipulators

X. Mechatronics

Preface

Robotics research, especially mobile robotics is a young field. Its roots include many engineering and science disciplines, from mechanical, electrical and electronics engineering to computer, cognitive and social sciences. Each of this parent fields is exciting in its own way. Each of the field has its share in different books.

This book is the result of inspirations and contributions from many researchers worldwide. It presents a collection of wide range research results of robotics scientific community. Various aspects of current research in robotics area are explored and discussed. We have tried to investigate the mainly important research areas of the really wide, wide range or robotic science.

We hope you will enjoy reading the book as much as we have enjoyed bringing it together for you. The book presents effort by a lot of people. We would like to thank all the researchers and especially to the chapter authors who entrusted us with their best work. It is their work that enabled us to collect the material for this book. Of course, great acknowledgments to the people who had invest their time to review all manuscripts and choose only the best one.

The short description of every chapter follows.

The book begins with researches in robot modelling & design, in which different approaches in kinematical, dynamical and other design issues of mobile robots are discussed. Modelling is a first step in designing a new system; kinematics is the most basic study of how mechanical systems behave. In mobile robotics, we need to understand the mechanical behaviour of the robot both in order to design appropriate robots for tasks and to understand how to create control software for an instance of mobile robot hardware. This chapter presents different researches in design of various robot systems.

One of the most important tasks of an autonomous system of any kind is to acquire knowledge about its environment. This is done by taking and extracting information from measurements of different sensor systems. In second chapter the various sensor systems are presented, but the major part of the chapter is devoted to robotic vision systems. Why? Because the vision is most powerful sense in all living beings. It provides the robot with enormous amount of information about the environment and enables rich, intelligent interaction in dynamic environments.

Chapter III is devoted to robot navigation and presents different navigation architectures. The book started with design of robot systems, went through sensors for determining the robot's environmental context. We now turn our attention to the robot's cognitive level. Cognition generally represents the purposeful decision-making and execution that a system utilizes to achieve its highest-order goals. The specific aspect of cognition is navigation competence. Given partial knowledge about its environment and a goal position or series of positions, navigation encompasses the ability of the robot to act based on its knowledge and sensor values so as to reach its goal positions as efficiently and as reliably as possible. Within the mobile robotics research community, a great many approaches have been proposed for solving the navigation problem. In this chapter, only some of them are presented.

Biological systems have always been, especially in the latest time, inspiration in designing the robot systems. Especially in designing the robot cognitive level (robot controllers). Robots are now capable of learning; various neural and fuzzy systems are incorporated in their “mind”. The chapter IV is devoted to research on adaptive and learning systems in mobile robots area.

There are many reasons why designing a multi-robot systems, but the most frequent answer is that the researchers are able to design more robust and reliable systems by combining not so reliable but redundant components. This field has been growing enormously in the last decade. The chapter V speaks about different application areas of multi-robot systems.

Other emerging field is discussed in chapter VI - the human- robot interaction. HRI is a cross-disciplinary area, which poses barriers to meaningful research, synthesis, and technology transfer.

Mobile robots are extensively used in various terrains to handle situations inaccessible to man. Legged robots in particular are tasked to move in uneven terrain. Hence the primary problem of these robots is locomotion in an autonomous fashion. Chapter VII gives a great tutorial on legged robot systems and one research overview on design of a humanoid robot.

Technical advances in the fields of sensor, control, and drive technology enabled intelligent robotic systems to be implemented in areas other than industrial production. The different examples of service robots are showed in chapter VIII.

Chapter IX is oriented to industrial robots, i.e. robot manipulators. Robots are nowadays primarily used industry and that is a reason to include this chapter in the book. Chapter gives different research results primarily in control of robot manipulators.

The term "mechatronics" has been used for about 35 years. It is derived from the observation of the synergy achieved through the integration of mechanical, electrical and information technologies in the design, manufacture and operation of industrial products and processes. Synergies may be in terms of performance, physical dimension, cost, power efficiency, time for development, dealing with complexity, and so on. Different mechatronic systems oriented on robotics are explored in the last chapter of the book - chapter X.

-I-

Modelling & Design

Dynamic Modelling and Adaptive Traction Control for Mobile robots

Abdulgani Albagul, Wahyudi Martono & Riza Muhida

1. Introduction

Mobile robots have been used in many application such as moving material between work stations. They can also be found in many areas such as industrial, medical, environmental and even domestic machines. Research on mobile robots has mounted and attracted so much attention in recent years since they are increasingly used in wide range of applications (Albagul, 2001); (Karam & Albagul, 1998); (DeSantis, 1995); (Yutaka & Yuta, 1988). At the beginning, most of research have been directed to the use of kinematic models of the mobile robots to achieve and accomplished the motion control (DeSantis, 1995); (Yutaka & Yuta, 1988); (Nelson & Cox, 1988). Later on, the research has taken another approach and has focused on robots with additional sensory system to develop autonomous guidance path planning systems (Nelson & Cox, 1988). This direction has led to produce sophisticated sensory systems that can learn about the operating environment and hence evaluating path constraints to the path planning objective itself (Wilfong, 1988). However, some research has also addressed some topics related to dynamic characteristics of the motion which are essential to path tracking objective. Shiller (Shiller, 1991) has studied the problem of computing suitable trajectories in the face of varying terrain topography and under road holding constraints. The problems of road handling and traction become very important when the robot is subjected to dynamic variations. These variations include changes in robot inertia and centre of gravity which caused by the variable carrying load. The changes in the terrain topography, texture or in wheel properties due to wear, contamination or deformation play a major role in the robot motion. These variations can easily affect the traction properties and hence the robot movement and may cause slippage to occur. Therefore, it is very important for the robot to be able to avoid slippage and handle is consequences. This requires some learning mechanism which will try to adapt the trajectory planning strategy to cope with any condition. This paper presents the work that has been done to explore the issue of dynamic modelling and motion control under varying loading conditions. This leads to the development of a dynamic model for a three wheeled mobile robot structure (Albagul, 2001); (Karam & Albagul, 1998). The model includes the capability for representing any shape of cart with variable density to accommodate any changes in the robot structure. The motion dynamic for the robot can be achieved by applying a number of forces acting at any point and in any direction. In fact, this model is not restricted to wheeled robot. It can be used to obtain general free body motion dynamics. For the purpose of this work, the motive forces driving the robot are exerted by two independent DC motors each driving on of the rear wheels. This configuration is commonly used in autonomous vehicles and is called differential drive mechanism. The simulation model of the robot dynamics is implemented in the Matlab computational environment. The exploration of the inertia variation and traction conditions whilst using standard motor control algorithms for given motion trajectories will be presented and discussed. The over all task of motion control eventually translates into deriving the required input to the DC motors. This can be divided into kinematics path trajectory planning and translation into the dynamic motor rotation trajectories. In order to execute accurate paths, the motor motion trajectories must consider both constraints do to the motor and load dynamics and those due to chrematistics. Elements of feed forward (pre-planning) and feedback are required to optimize the trajectories. Cross coupling effects between the two drive motors can also be compensated for and this provides for a further degree of accuracy in the path following objective In the face of varying conditions, autonomous adaptation requires that some measure of performance is made available. For the vehicle this will be in the form of position, orientation and wheel speed measurements. These provide feedback on dynamic path tracking accuracy and also on slippage. This feedback will enable both 'low' and 'high' level adaption of the vehicle motion in order to suit new conditions thus enabling the motion to be optimized continuously without reduction in efficiency and/or loss of stability.

2. Robot Model

The considered robot is modelled as a three dimensional rigid body which consists of a number of small particles connected together. The particles have a cubic shape with uniform density. This is to ease mathematical analysis which will be carried out on the body without reducing the generality of the model since the cubes can be made arbitrarily small and numerous. The shape and dimensional definition of each particle is shown in Fig. 1. By using the general model, the mass, inertia and centre of gravity of each particle can be calculated. Then the overall mass, inertia and centre of gravity of the entire body of the robot can be obtained. In Fig. 1, the axes of reference frame of the entire body are denoted as X_0, Y_0, and Z_0, while , Y_n, Z_n are the axes of the reference frame for particle n. The original coordinates of each particle, n, are denoted as X_{n0}, Y_{n0}, Z_{n0}, while X_{nmax}, Y_{nmax}, Z_{nmax} are the maximum dimensions of the particle relative to the origin of the particle reference frame. The centre of gravity of the particle to its reference frame is X_{ncg}, Y_{ncg} and Z_{ncg}, while the center of gravity of the entire body is X_{cg}, Y_{cg}, Z_{cg}. Fig. 2 shows a 3D model of the robot created by Matlab tool.

2.1 Kinematic Model

Most of kinematic models of mobile robots assume that no tire slippage occurs, so the inputs to the system are right and left wheel angular velocities ω_r and ω_l respectively. Then the motion of the robot can be described by the simple kinematics of rigid bodies. In order to determine the robot motion, it is so important to define the position and orientation the robot as the location and orientation of the centre of gravity, p(X_0, Y_0, θ_0), relative to the fixed world frame. With the assumption of no lateral or longitudinal tire slip, the linear velocities of the right and left wheels can be expressed as:

$$v_r = R_t \omega_r \hat{i} \tag{1}$$

$$v_l = R_t \omega_l \hat{i} \tag{2}$$

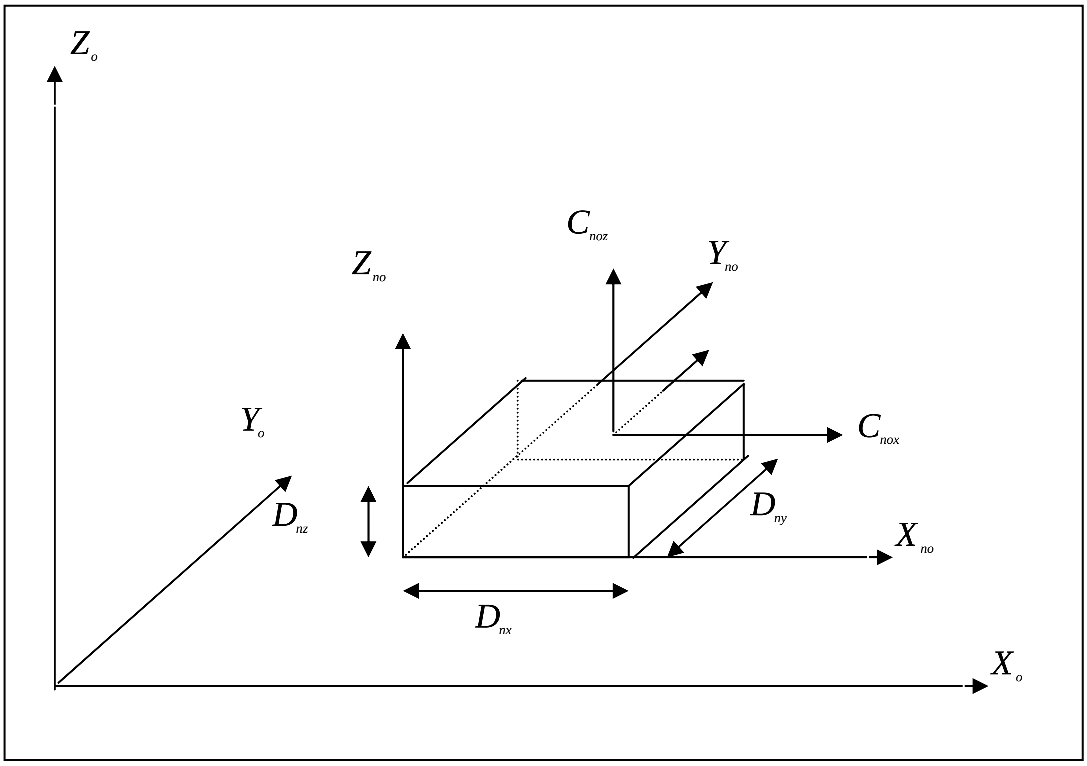


Figure 1. Particle dimensional definition

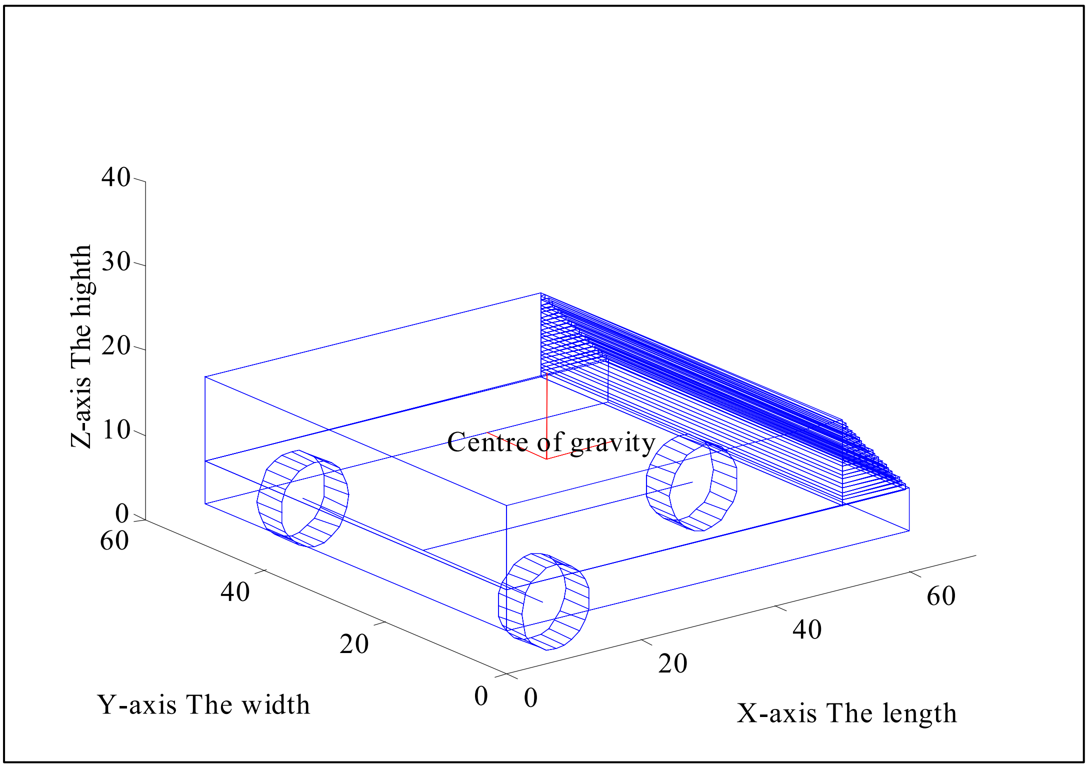


Figure 2. A 3D model of the robot created by the Matlab

The equations of motion of the rigid cart with respect to each wheel are:

$$V_R = V_g + r\hat{k}\,(-L_r\hat{i} - \frac{T_r}{2}\,\hat{j}) = u\,\hat{i} + v\,\hat{j} + r\frac{T_r}{2}\hat{i} - rL_r\,\hat{j} \tag{3}$$

$$V_L = V_g + r\hat{k}\,(-L_r\hat{i} + \frac{T_r}{2}\hat{j}) = u\,\hat{i} + v\,\hat{j} - r\frac{T_r}{2}\hat{i} - rL_r\,\hat{j} \tag{4}$$

where V_g is the velocity vector of the centre of gravity, p(X_0, Y_0, θ_0), L_R is the distance from the rear axle to the centre of gravity, Tr is the distance between the two driving wheels, υ, v and r are the forward, the lateral and the yaw velocities respectively. Using equations 1 - 4, the full kinematic model for the robot can be obtained as:

$$u = \ (\omega_r + \omega_l)\frac{R_t}{2} \tag{5}$$

$$r = (\omega_l - \omega_r)\frac{R_t}{T_r} \tag{6}$$

$$v = (\omega_l - \omega_r)\frac{L_R R_t}{T_r} \tag{7}$$

Then the heading, velocities and position of the robot in the world coordinate system can be obtained.

$$\vartheta_0 = \int r\,dt \tag{8}$$

$$V_X = u\cos\vartheta_0 - v\sin\vartheta_0 \tag{9}$$

$$V_Y = u\cos\vartheta_0 + v\sin\vartheta_0 \tag{10}$$

$$X_0 = \int V_X\,dt \tag{11}$$

$$Y_0 = \int V_Y\,dt \tag{12}$$

where, V_X and V_Y are the velocity components of the vehicle relative to the world frame, X_0, Y_0 and θ_0 are the position and heading of the robot in the world frame as shown in Fig. 3.

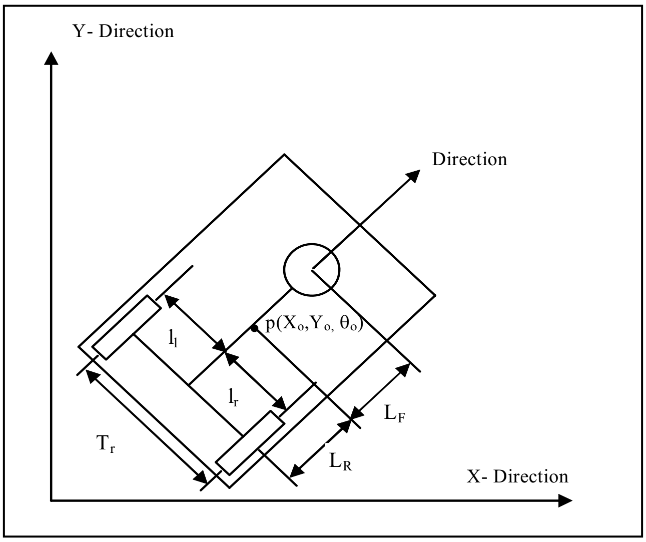


Figure 3. Robot geometry and coordination

2.2 Dynamic Model

In order to obtain the dynamic model of the robot, forces must be applied, analyzed and moments must be taken about some point on the robot, in this case the centre of gravity. Since the robot is a three-degree of freedom 3DOF, which allows only movement in the longitudinal and lateral directions along with angular displacement, the equation of force and moment can be expressed as:

$$\sum F_X = m(\dot{u} - vr) \tag{13}$$

$$\sum F_Y = m(\dot{v} - ur) \tag{14}$$

$$\sum M_Z = I_Z \dot{r} \tag{15}$$

The forces acting on the robot are those forces exerted by the right and left driving wheels. These forces are proportional to the applied torque minus the amount of torque required to accelerate the wheels, gearbox and rotor. The applied torque is divided into two ways; linear torque to accelerate the robot and angular torque to accelerate the wheels, gearbox and rotor.

$$T_{app} = T_{lin} + T_{ang} \tag{16}$$

where Tapp, Tlin and Tang are the applied, the linear and the angular torques respectively. The linear torque is converted into a longitudinal force at the tire/ground interface and expressed by:

$$F_X = \frac{T_{lin}}{R_t} \tag{17}$$

and the angular torque is:

$$T_{ang} = I_Z\, \dot{\omega} = I_Z\, \frac{\dot{u}}{R_t} \tag{18}$$

$$F_X = \frac{T_{app} - T_{ang}}{R_t} = \frac{R_t T_{app} - I_Z\, \dot{u}}{R_t^2} \tag{19}$$

Using equation 19 the distinction can be made to obtain the forces exerted be the right and left driving wheel as follows:

$$F_{Xr} = \frac{R_t T_{appr} - I_Z\, \dot{u}_r}{R_t^2} \tag{20}$$

$$F_{Xl} = \frac{R_t T_{appl} - I_Z\, \dot{u}_l}{R_t^2} \tag{21}$$

The dynamic equations describing the motion of the robot in terms of accelerations are:

$$\dot{V}_X = \frac{F_{Xr} + F_{Xl}}{m} + V_Y\, \omega \tag{22}$$

$$\dot{V}_Y = \frac{F_{Yr} + F_{Yl}}{m} - V_X\ \omega \tag{23}$$

$$\dot{r} = \frac{L_r F_{Xr} - L_l F_{Xl} - L_R(F_{Yr} + F_{Yl})}{I_Z} \tag{24}$$

The longitudinal, lateral and yaw velocities are simply the integral of their accelerations. Thus, the potion and heading of the robot can be obtained in the same manner as in the previous section.

3. Path Planning

Path planning is the subject that deals with the displacement of the robot in a prior known environment. It plays a major role in building an effective sophisticated mobile robot. Path planning as well as trajectory generation are required prior to the movement of the robot. The robot is desired to move from a starting position to a goal point in the workspace. A point on the robot is specified and designated to follow the required path. There are many methods to generate paths in terms of smoothness and curvature as well as continuity. Some of these methods are complicated and time consuming but produce very smooth paths. In this paper an effective method for path and trajectory generation is adopted. This method consists of straight lines joined by circular arc segments with a specified radius and turning angle. These circular segments are to avoid stoppage and provide continuity for the robot. This method is based on some parameters that should be known to define the path.
These parameters are:

1.) The start and end coordinates of each straight line section, or alternatively the length and heading of the section,
2.) The start of each circular arc segment, its radius and turning angle, which corresponds to the change of orientation.

Fig. 4 shows a simple path that consists of two straight lines joined by an arc with a radius rarc and a turning angle, θ_{arc}, of 90°.

3.1 Velocity Trajectory Generation

In this section, the generation of a velocity trajectory profile for a given continuous geometric path is treated as a function of time. As mentioned in the previous section that the path is divided into a number of sections connected together in series, each section can be set to have its own target of maximum velocity, called the section velocity. Also each section has been divided into three regions for acceleration, constant velocity and deceleration with respective lengths, d_A, d_C and d_D as well as time duration t_A, t_C and t_D plus angles and radius for the circular arc sections. The basis of the velocity trajectory is to accelerate in each section to reach the specified maximum velocity of that section if possible within the section distance. If the maximum velocity achieved for the section the robot must be able to decelerate to next maximum velocity of the next section to maintain the continuity of the path tracking. The intelligent of the algorithm is that it can work in no more than two operations. The first operation is to accelerate the robot and the second one is to decelerate it. The first operation will check if the maximum velocity of the section can be achieved, with the given acceleration, from the start velocity within the length of the section. If the maximum velocity can be reached then the acceleration distance and time are computed and the remaining distance of the section is driven at the maximum velocity. If the maximum velocity is not reachable, the robot will then either accelerate throughout the whole section if it is followed by another one or just accelerate and decelerate if it is one section. The exit velocity of a section must automatically be the start velocity of the next section. The second operation deals with calculating the distances and times for acceleration and deceleration from the start, section or the maximum velocity to the exit velocity of each section. If the section length is insufficient to decelerate to the exit velocity then the start velocity must be changed to another new velocity that the section can be completed with the desired exit velocity. It is required that the maximum velocity of the first section must be as low as possible for the robot to be able to decelerate normally to the start velocity of the next section if it is lower. A high maximum start velocity could only occur if the trajectory generation algorithm is used on-line.

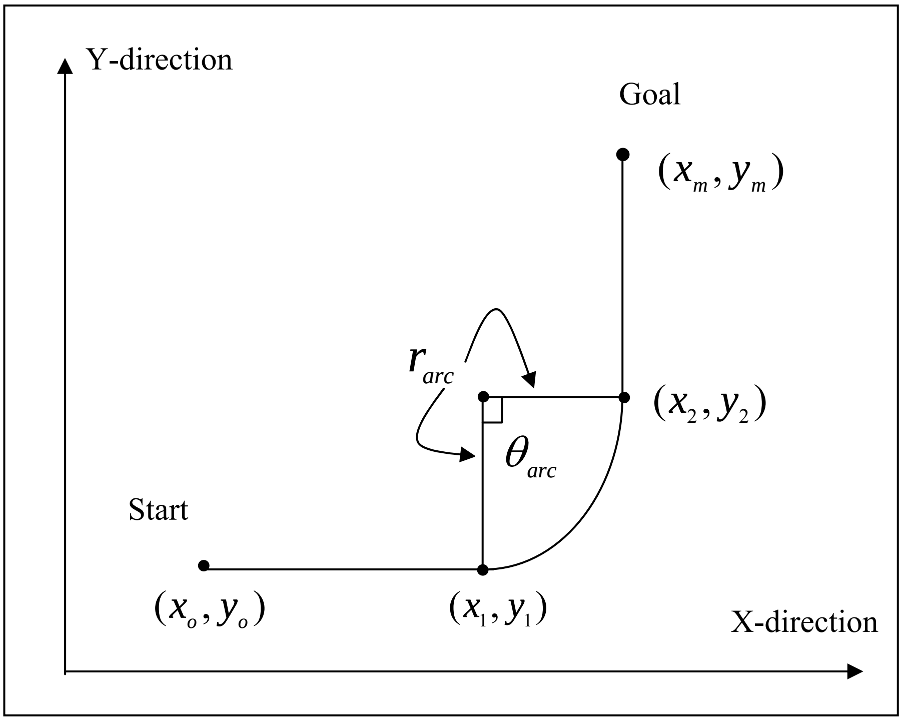


Figure 4. Example of path definition

3.2 Explanation of the Algorithm

In the first step, each section is set to have a start velocity equal the previous section exit velocity for the sake of continuity and may be to avoid slippage if the robot is known to operate in some slippery sections or if it is executing a tight turn. If the start velocity is too high to decelerate to the exit velocity then it must be set to a lower velocity.

The second step is to check if the length of the section large enough to allow the robot to reach for the maximum velocity if so then the time and distance, necessary for acceleration from the start velocity to the maximum velocity of the current section, are calculated as well as the time to drive the remaining distance with the constant maximum velocity. Otherwise the resultant velocity of constant acceleration is the end velocity of the section. First and second steps together ensure that there is no demand for velocity increase greater than the robot actuators can deliver with fixed acceleration.

In the third step, the end velocity of the current section is set to equal to the start velocity of the next section. This is to maintain the smoothness and continuity of the motion and also that the deceleration in each section starts in time to complete the section and reach the start velocity of the next section.

Forth step is dealing with calculating the time and distance needed for the robot to decelerate from the highest velocity reached during the execution of the section to its end velocity. The highest velocity is the maximum velocity if the section is long enough. Otherwise it is the peak velocity if the section contains only acceleration and deceleration parts.

4. Motion Control

The task of the controller is to achieve various goals and desired features for the robot motion. It is also designed to execute the planned sequences of motions correctly in the presence of any error.

The control strategies in this work are based on controlling the traction forces causing the motion as well as maintaining the tracking of the desired path without any slippage or deviation.

Traction motion control has some desired feature such as:

1.) Maintaining the fastest possible acceleration and deceleration,
2.) Maintaining the desired path following accuracy,
3.) Maintaining the robot stability during the manoeuvres,
4.) Preventing the robot from slipping or sliding.

The block diagram of the mobile robot control system is shown in Fig. 5.

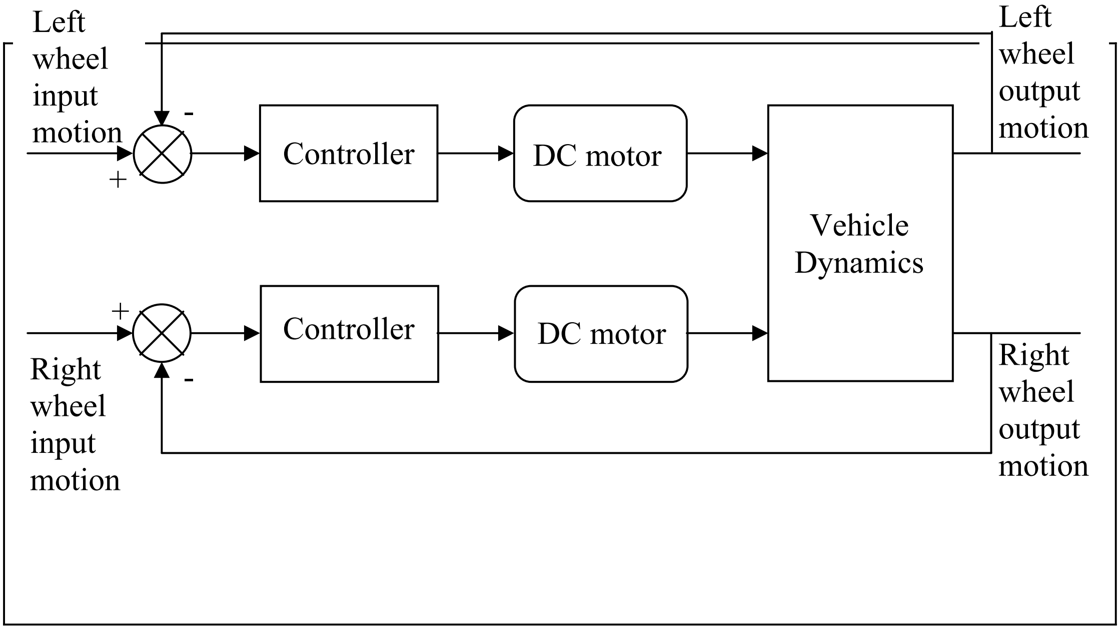


Figure 5. Block diagram of mobile robot control system

The controller tasks consist of two parts. The first part is the 'Low level' control strategy which deals with the dynamic changes in the robot. The second one is the 'High level' control strategy which deals with the changes in the environment in which the robot operates. The combined Low and High level motion control for the robot is depicted in Fig. 6.

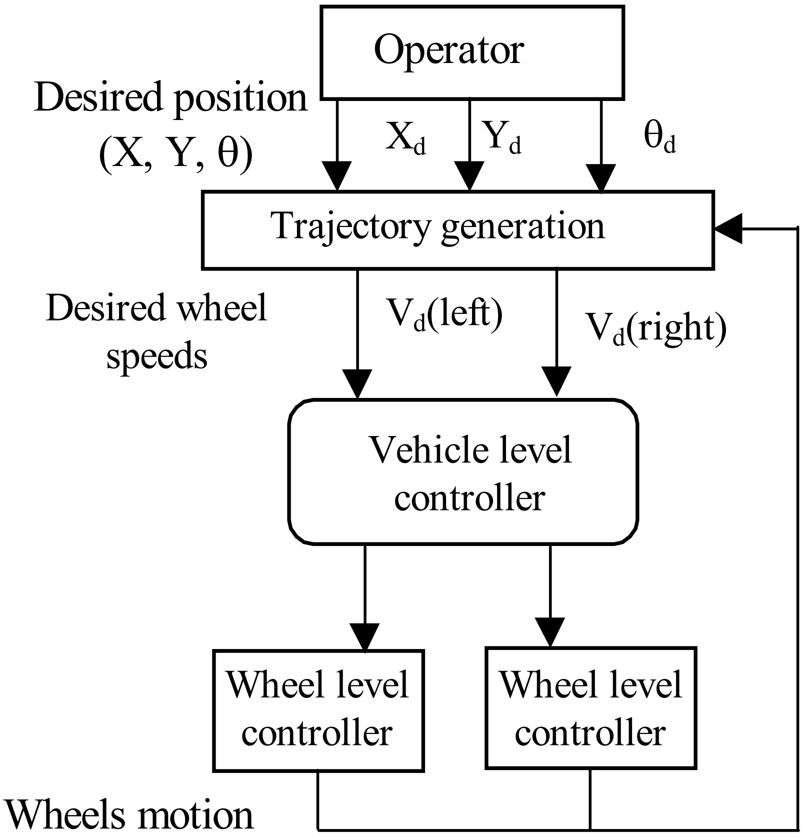


Figure 6. Motion control of the mobile robot

4.1 Low Level Control Strategy

In this strategy, an adaptive motion control algorithm based on the Pole placement self-tuning adaptive controller is considered. The controller has been designed with a PID structure to estimate the changes in the system dynamic parameters. The self-tuning adaptive control structure offers a good framework to estimate the model parameters. This can be achieved through the estimation mechanism. Then the controller is implemented and is able to cope with the dynamic changes and take the proper action to control the motion. However, it can not estimate or cope with the changes in the surface conditions. The basic block diagram for the self-tuning adaptive motion controller is shown in Fig. 7.

4.2 Combined Control Strategy

In this strategy, a combined motion control base on 'Low' and 'High' level controls is adopted and presented in this section. The combined controller takes in consideration both the robot dynamics and the environment structure. This is the main fundamental difference between the combined control and the Low level control which does not include the environment structure and its conditions as shown in figure.

The two main issues to look for, when including the environment structure, are the presence of obstacles in the surrounding and the condition of surface such as smoothness and dryness. This paper concentrates and emphasizes on the state of the surface in order to examine the robot motion under different surface conditions compare the performance with the Low level control strategy. Fig. 8 shows the robot movement under different surface conditions.

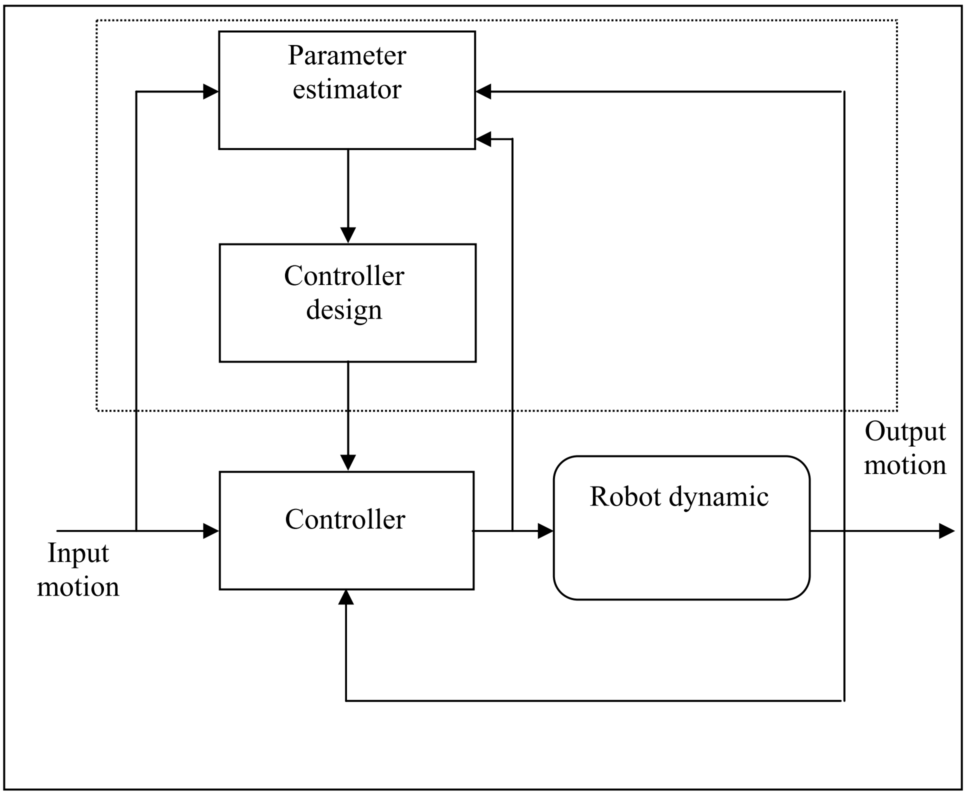


Figure 7. Self-tuning adaptive controller structure

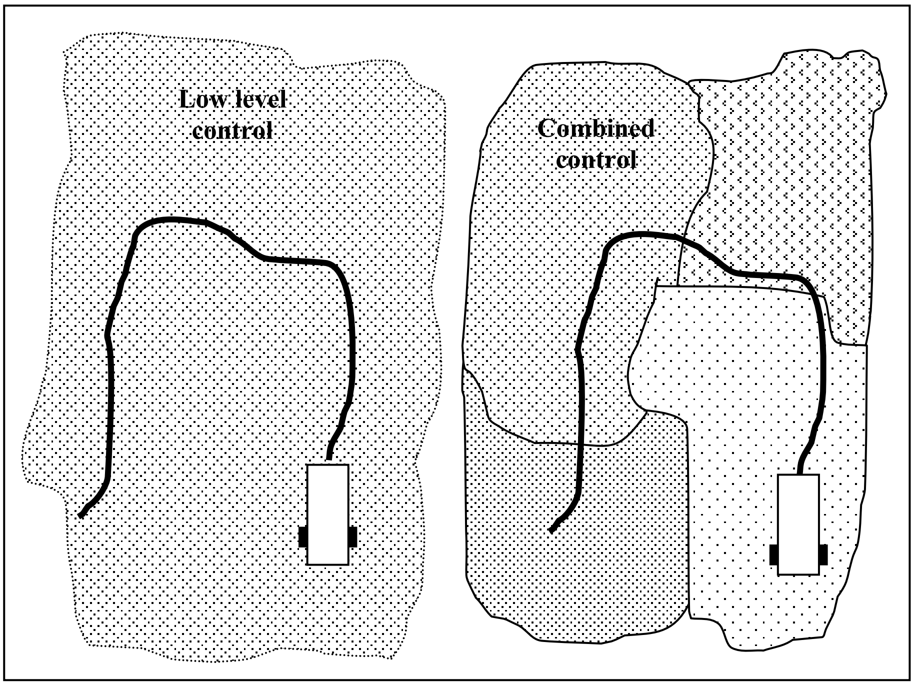


Figure 8. Robot motion control under different surface conditions

5. Simulation and Results

A general dynamic model to represent the vehicle as a three dimensional rigid body has been built using Matlab software. The model will calculate the mass, centre of gravity and the inertia of the entire body. These values are used in the dynamic model of the robot

which the controller operates on. An arbitrary over all path has been used to test the motion control of the robot. The effect of the 'low' level adaptive and the combined controllers has been investigated on the robot model with changes in the state of the surface condition (i.e. different friction coefficients). The results indicate that the combined 'low' and 'high' level adaptive controller is able to compensate and cope with the changes on the path condition and provide better path tracking. Selected sections of the velocity trajectories for the right and left wheels have been chosen to illustrate the difference between the performance of the 'low' level adaptive controller and the combined controller under these conditions. Also the displacement error in the over all path-tracking are presented to show the advantages of the combined controller. Fig. 9. and Fig10 correspond to performance of the robot after the change in the friction coefficient (i.e. from μ=0.8 to μ=0.4) with the following parameters:
mass (m)= 40 [Kg]; Inertia (J)= 2.627 [Kg/m2]; L_F= 0.3[m]; L_R = 0.1 [m]. Both controllers subjected to the changes in the surface condition (i.e. friction coefficient).
The technical specification for the vehicle motion is subject to some practical constraints which the controller must take into account.

These constraints are:

Maximum linear velocity:	V_{max}=2 (m/s)
Maximum linear acceleration:	A_{max}=1.5 (m/s2)
Maximum angular velocity:	ω_{max}=2 (rad/s)
Maximum angular acceleration:	ω_{max}=1.5 (rad/s2)

The actuators are sufficiently powerful to achieve these maxima within their own operating constraints on maximum voltage. Fig. 9 shows the effect of the low level controller. Meanwhile Fig. 10 shows the performance of the combined controller on the Robot.

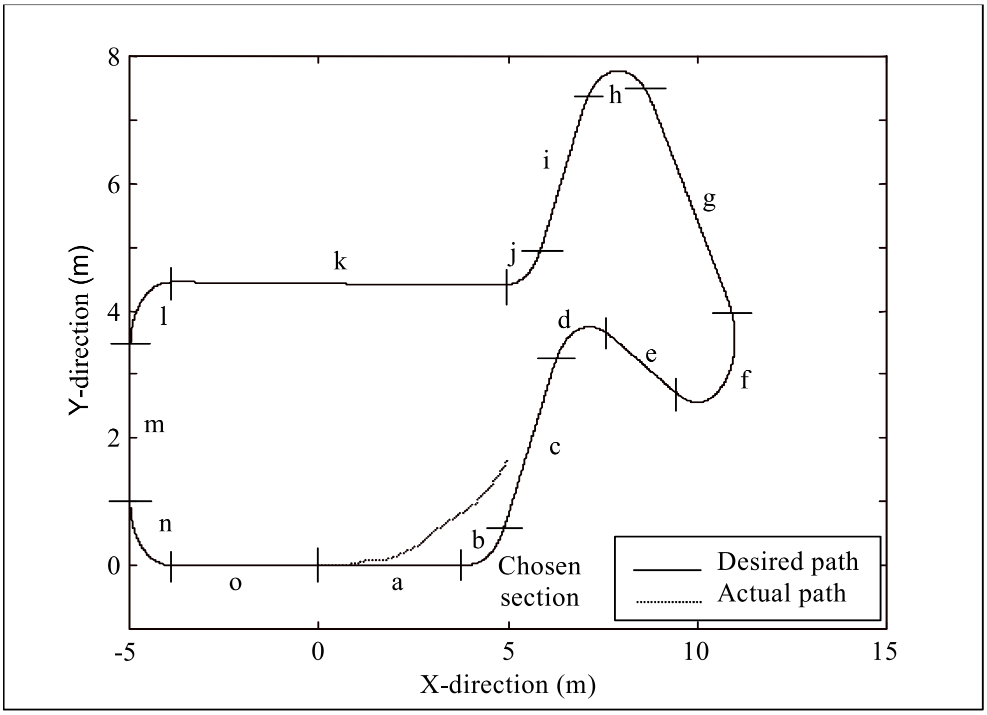


Figure 9(a). Path-tracking after changes in the friction coefficient

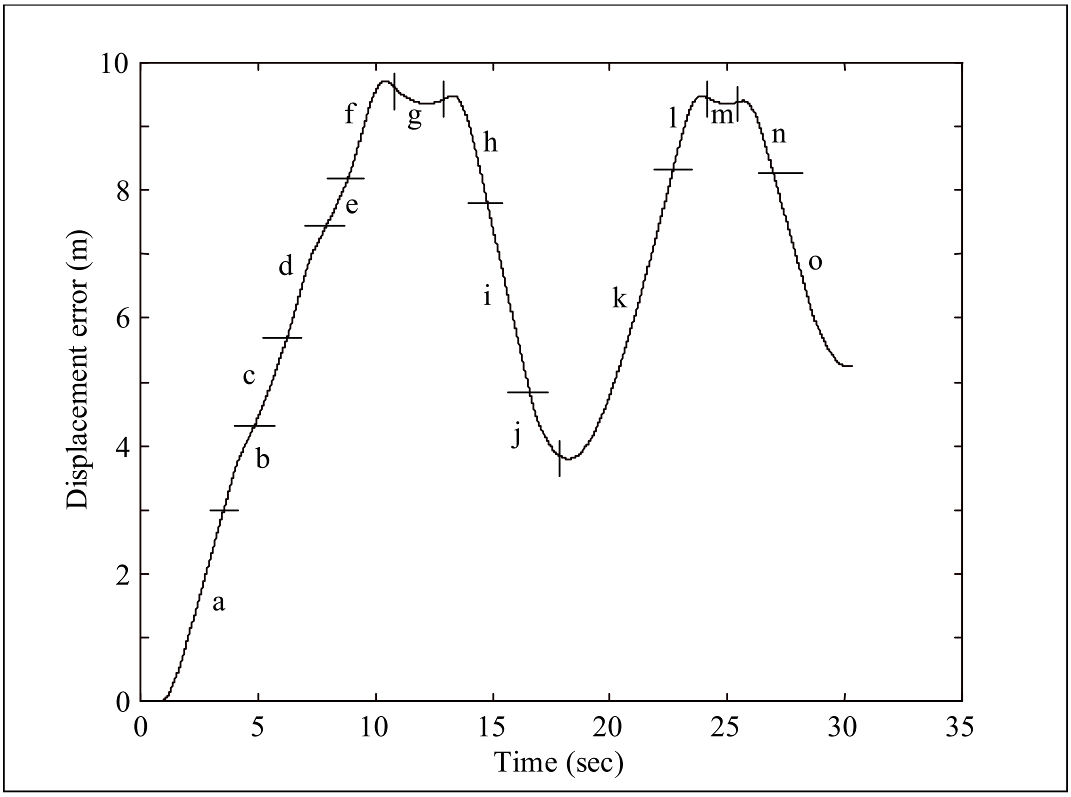


Figure 9(b). Displacement error in path-tracking

Fig. 9(a) shows the path-tracking of the robot. It can be seen that once the slippage occurred, the robot could not follow the desired path. Fig. 9(b) shows the displacement error of the robot. Here, the error or the deviation of the robot from the desired path is large due to the slippage and the low-level controller could not cope with the changes in the surface conditions.

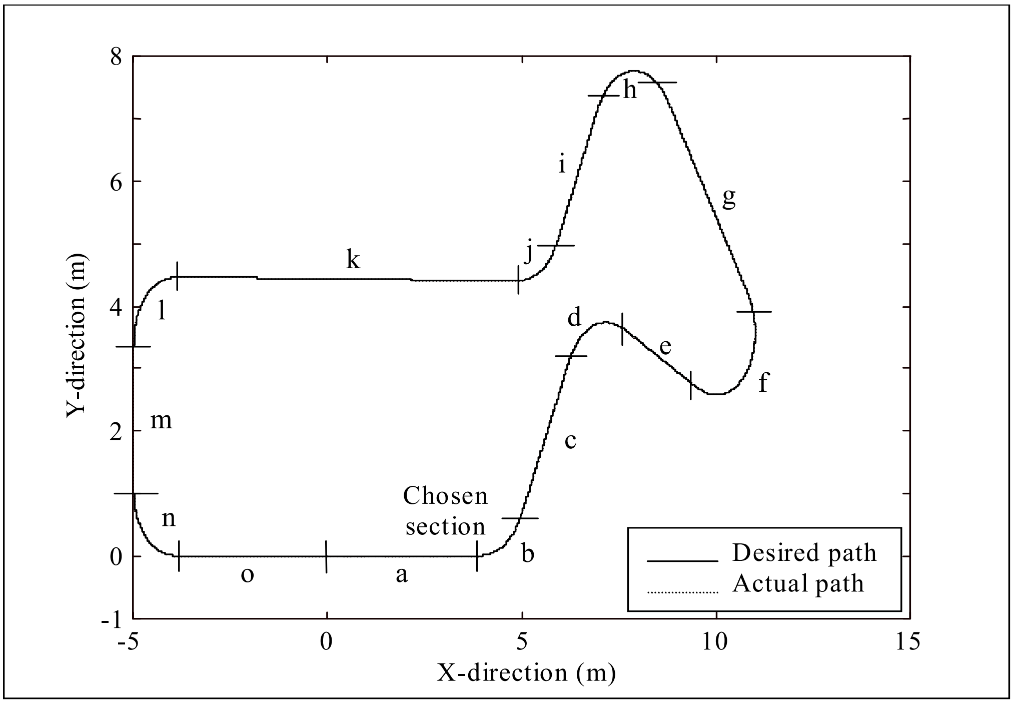


Figure 10(a). Path-tracking after changes in the friction coefficient under the combined controller

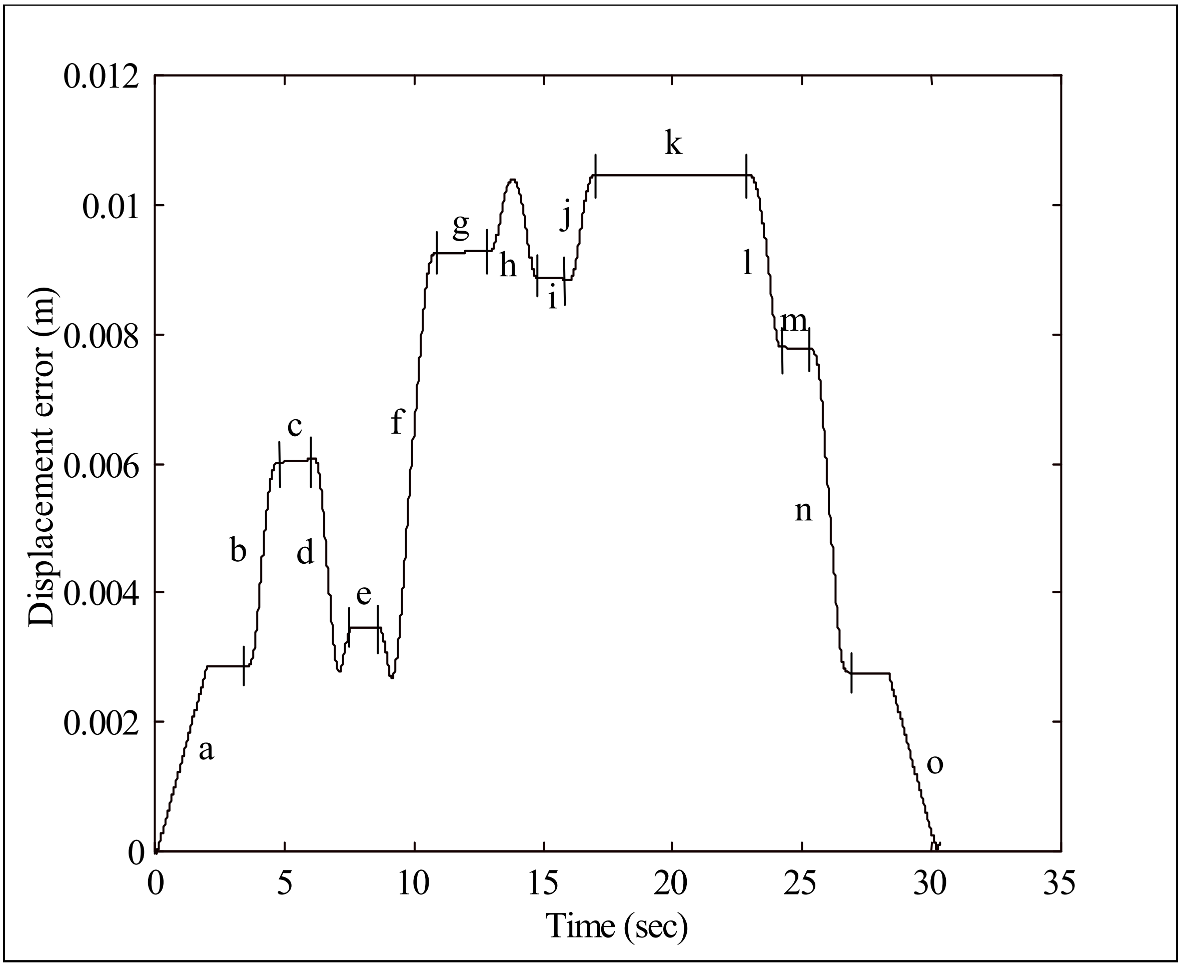


Figure 10(b). Displacement error in path-tracking under the combined controller

Fig. 10(a) shows the path-tracking of the robot in the case of the combined controller. It can be seen that the robot is following the desired path and achieving a good tracking with accuracy. Fig. 10(b) shows the displacement error which is very small and the robot reaches the final position with accuracy.

6. Conclusion

In this paper two control strategies are developed, and tested on the robot. The 'low' level controller performance deteriorated with the changes in the surface condition such as the traction condition (friction coefficient). Meanwhile the combined controller detects the changes and copes with them in an adequate manner, maintaining a largely consistent performance. Some of the issues concerning the environmental structure and the high level control have been presented. Determining the location of the mobile robot plays a vital role in maintaining fast, smooth path-tracking. Measuring the position of the robot in the workspace gives the high level controller an indication of whether the robot is experiencing any slippage or not. All these issues are important for the motion control. The combined control system has been investigated and tested on the differential drive mobile robot. Simulation results show that the performance of the mobile robot under the combined system has improved and the accuracy of path-tracking also improved significantly as it can be seen from the figures.

7. References

Albagul A. (2001), Dynamic Modelling and Control of a Wheeled Mobile Robot, PhD thesis, University of Newcastle upon Tyne, Apr 2000. 2 (Albagul, 2001)

Cox, I. J. (1988). Blanche: An Autonomous Robot Vehicle for Structured Environment, Proceedings of IEEE Intl. Conf. on Robotics and Automation, 1988. 9 (Cox, 1988)

DeSantis, R. M. (1995). Modeling and Path-tracking Control of a mobile Wheeled Robot with a Differential Drive, Robotica, Vol. 13, 1995. 4 (DeSantis, 1995)

Hongo, T.; Arakawa, H.; Sugimoto, G.; Tange K. & Yamaoto, Y. (1987). An Automatic Guidance System of a Self-Controlled Vehicle, IEEE Trans. on Indus. Elect. Vol. IE-34, No. 1, 1987. 8 (Hongo et al., 1987)

Karam, K. Z. & Albagul, A. (1998). Dynamic Modelling and Control Issues for a Three Wheeled Vehicle, Proceedings of 5th Int. Con. on Control, Automation, Robotics and Vision, Singapore, Dec 1998. 1 (Karam & Albagul, 1998)

Nelson, W. L. and Cox, I. J. (1988). Local Path Control for an Autonomous Vehicle, Proceedings of IEEE Intl. Conf. on Robotics and Automation, 1988. 5 (Nelson &Cox, 1988)

Shiller, Z. (1991). Dynamics Motion Planning of Autonomous Vehicles, IEEE Trans. on Robotics and Automation, Vol. 7, No. 2, 1991. 7 (Shiller, 1991)

Wilfong, G. T. (1988). Motion Planning for an Autonomous Vehicle, Proceedings of IEEE Intl. Conf. on Robotics and Automation, 1988. 6 (Wilfong, 1988)

Yutaka, K. & Yuta S. (1988). Vehicle Path Specification by a Sequence of Straight Lines, IEEE Trans. on Robotics and Automation, Vol. 4, No. 3, 1988. 3 (Yutaka & Yuta, 1988)

Rapid Prototyping for Robotics

Imme Ebert-Uphoff, Clement M. Gosselin, David W. Rosen & Thierry Laliberte

1. Introduction

The design of robotic mechanisms is a complex process involving geometric, kinematic, dynamic, tolerance and stress analyses. In the design of a real system, the construction of a physical prototype is often considered. Indeed, a physical prototype helps the designer to identify the fundamental characteristics and the potential pitfalls of the proposed architecture. However, the design and fabrication of a prototype using traditional techniques is rather long, tedious and costly. In this context, the availability of rapid prototyping machines can be exploited in order to allow designers of robotic mechanisms or other systems to build prototypes rapidly and at a low cost.

This chapter summarizes the research experience of two research groups, one at Universit´e Laval, the other at Georgia Tech, concerning the rapid prototyping of mechanisms. The two groups employed two different types of RP technology, Fused Deposition Modeling (FDM) and Stereolithography (SL), respectively, and the use of both is described in this chapter.

The two types of Rapid Prototyping technologies considered here, FDM and SL, are both based on the principle of Additive Fabrication, i.e. parts are built by adding material to the whole, as opposed to subtracting material from the whole (as is done in traditional machining processes). FDM and SL both build three-dimensional parts from a CAD model by building one layer after the other, which facilitates the construction of parts with any desired internal and external geometry in a fraction of the time and cost necessary to build them using a conventional process (Ashley, 1995). Additive Fabrication therefore provides several advantages for the quick and efficient construction of mechanical prototypes:

- Quick turn-around time, thus facilitating many design iterations;
- No limits on part complexity, as parts with complex geometry can be built just as quickly and cheaply as parts with simple geometry;
- Since most components are "self-made", there are no issues of components being available only in certain sizes from a manufacturer;
- Since the designer has access to the interior of the part during the build process, Additive Fabrication provides novel design possibilities (Binnard, 1999). These capabilities can be exploited to yield short-cuts that eliminate the need for fasteners and the need for assembly of the prototypes.

Depending on the stage of development of the system to be prototyped, different strategies may be employed:

(A) At an early stage of experimentation the main goal is to quickly achieve a prototype that provides the basic functionality of the desired system. Implementation details are usually irrelevant at this stage, allowing for use of RP-specific short-cuts in the implementation to yield faster turn-around time.

(B) At the final design stages it is often desired that all parts of the prototype resemble the mechanism to be manufactured in as much detail as possible to test all aspects of the design.

1.1 Components of Robotic Mechanisms and Their Fabrication & Integration in Rapid Prototyping

Prototypes of robotic mechanisms require a variety of different components, which, in the context of this chapter, are categorized in the following three groups:

1. Rigid links;
2. Joints between the rigid links that enable motion;
3. External components that are required for the prototype, but that are not fabricated using Rapid Prototyping. Examples: actuators, sensors, circuits, etc.

This chapter discusses how RP can be used to generate a prototype of a mechanism that contains all of these components in an efficient manner. Different approaches exist for the fabrication and integration of the components as outlined below:

1. Rigid Links: Rigid links of any shape are easy to fabricate in Rapid Prototyping. Since the construction of rigid parts is the classic use of Rapid Prototyping, much research has been reported in this area and this topic is not further discussed in this chapter.
2. Joints Between Rigid Links: Mechanical joints can be fabricated as conventional (kinematic) joints or as compliant joints. Furthermore, conventional joints can be built in several parts, requiring assembly afterwards, or in a single step. All of these possibilities are discussed in this chapter.
3. Integration of external components (actuators, sensors, etc.): These components can be attached to the prototype through screws and fasteners, in order to mimic the design of the the final mechanism to be manufactured in as much detail as possible. As an alternative approach, one can seek to add actuators and sensors at the interior of parts by inserting them while the part is being built. This may eliminate the assembly process, the design of fasteners, etc., and thus speed up the prototyping process considerably at an early stage. Both of these approaches are considered in this chapter.

1.2 Organization of this Chapter

The remainder of this chapter is organized as follows: Section 2. reviews different types of RP technologies with emphasis on FDM and SL technologies. Section 3. reviews related research on the use of RP for automation and robotics.

Section 4. describes the use of FDM technology to build mechanical prototypes of mechanisms. A database of joint models, gears and fasteners, suitable for fabrication in FDM machines is introduced that provides the building blocks for the prototyping of mechanisms. All joints are of conventional type (with assembly) and external components are attached in the traditional way, mimicking the implementation of the final mechanism to be manufactured in as much detail as possible. Examples of several passive and actuated mechanisms are provided.

Section 5. discusses the fabrication of joints using SL technology. Conventional joints are discussed first and it is shown that the joint models developed in Section 4. can easily be

adopted for joint fabrication using SL technology. Then the fabrication of non-assembly conventional joints and non-assembly compliant joints using SL machines is discussed.
Section 6. explores the insertion of external components at the inside of parts during the build process. This research is more exploratory, but may ultimately lead to quicker turn-around times for prototypes in many applications, by eliminating the need for the design of fasteners and the assembly process at early stages of design.
Section 7. finally presents conclusions.

2. Rapid Prototyping Technologies

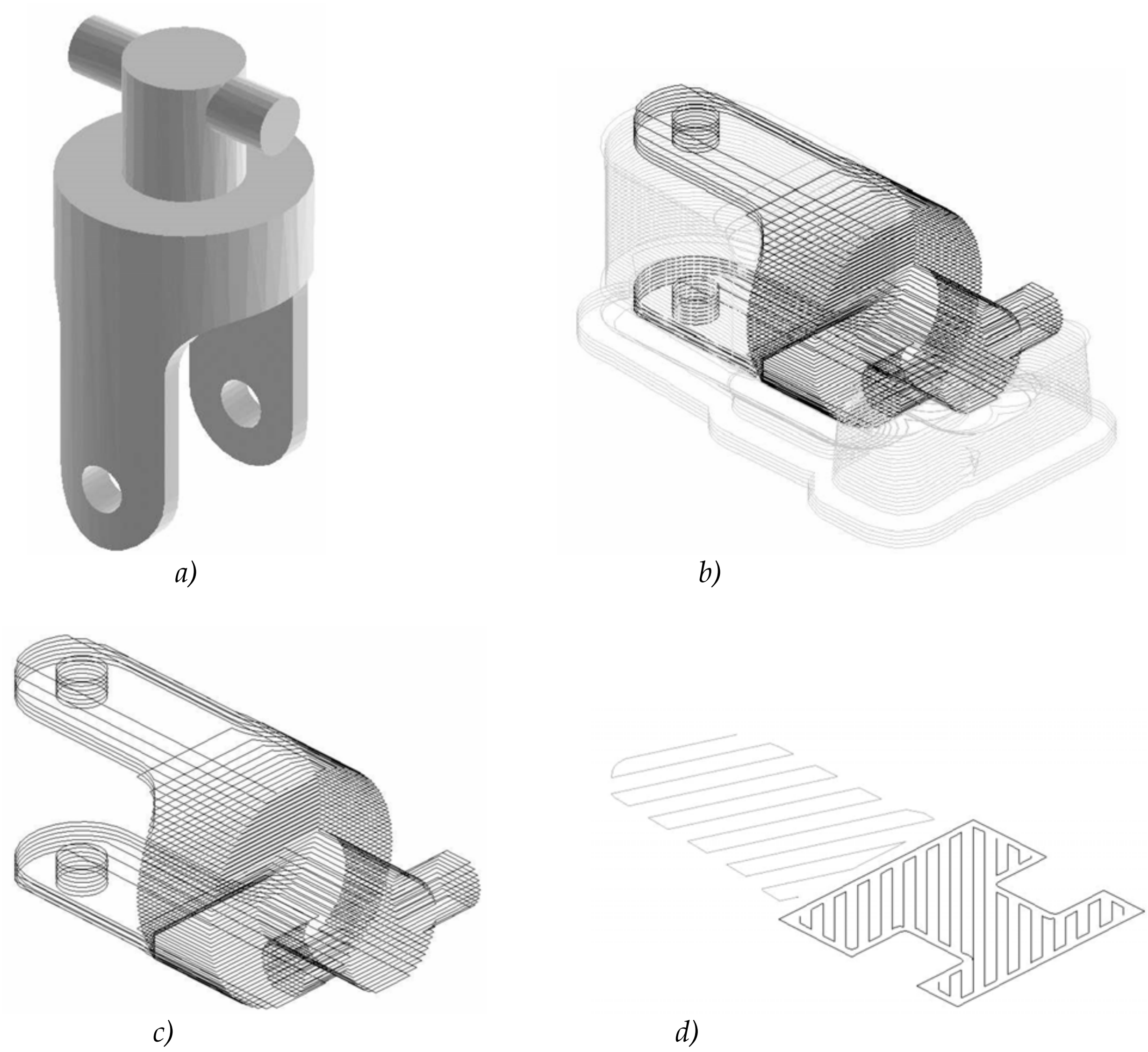
A wide variety of rapid prototyping technologies exists (for an excellent overview, see Kai & Fai (1997)) and many of them can be used for applications in Automation and Robotics (Wohlers, 1993). The principal idea underlying rapid prototyping technologies is that parts and devices are fabricated using a layer-based, additive process. A wide variety of material deposition methods, processing methods, and materials have been explored. For example, in the Selective Laser Sintering (SLS) process, a laser sinters, or melts, polymer powder particles together to form a part cross-section, while in Stereolithography a laser induces a chemical gelling reaction in a liquid photopolymer to fabricate the cross-section. In Fused-Deposition Modeling, a heated extrusion head extrudes polymer filament to trace out a part cross-section. These, and additional technologies, have been used in many applications other than making plastic prototypes, including medical visualization models, patterns for investment casting, molds for injection molding, and even some production uses (Jacobs, 1996).
Two technologies, FDM and SL, are used throughout this chapter to exemplify the capabilities of RP technology, with SL currently being the most widely used method of Rapid Prototyping world-wide. Parts fabricated by SL and FDM are typically of polymer material and ABS polymer, respectively, which may not be rigid enough for some applications. However, most of the discussion is not limited to those two technologies. If desired, other RP technologies, such as Selective Laser Sintering or Laser Engineered Net Shaping (LENS), can be employed to obtain metal parts with improved material properties.

2.1 Fused Deposition Modeling

One of the machines used here is a FDM 2000 rapid prototyping machine from Stratasys Inc., with the Fused Deposition Modeling technology (FDM). In the FDM technology, a thin thread of fused material is deposited layer by layer. The material usually used is ABS polymer. A supporting material is necessary where a new slice does not have any support from previous slices. Once the whole part is made, the supporting material is removed and some finishing is done if necessary. The main advantages of this specific technology are simplicity of use and relatively low cost. There are some limitations regarding the dimensional accuracy and the surface finish. Since the section of the part already built is not refused when the new material is deposited, the bounding between the layers and between the threads is not complete. Therefore, the parts are anisotropic and the material is weaker in the direction of fabrication, which corresponds to the direction of the slicing.
Each rapid prototyping technology has its own software. This software is able to understand a standard type of file named STL (Wohler, 1992). Most of the CAD packages are able to create an STL file of a part. In this work, the STL files are generated from Pro-Engineer. From this STL file, the software processes the model of the part into the

appropriate format. For the FDM technology, the part is first moved in an appropriate orientation. Then, the part is sliced in horizontal layers. The necessary support material is then created to hold the sections of the part which do not lie on lower layers.

Finally, the roads—the paths where the threads of material are deposited—are created for each slice. This process can be performed automatically. However, depending on the desired properties of the part, parameters of the roads (size, spacing and pattern) can be modified manually. In order to clarify the process, an example is presented in Figure 1 (a), (b), (c) and (d).



a) *b)*

c) *d)*

Figure 1. Example of the software process: (a) Part in STL format in its original orientation. (b) Part properly oriented and sliced. (c) Support material added to the sliced part. (d) Roads in one of the slices. Note that the roads of the support material include gaps between them for easy removal

2.2 Background on Stereolithography

Stereolithography (SL) is currently the most widely used RP method and it is used as the second example throughout the chapter. SL is a method of rapidly prototyping parts by using a photo-polymer resin cured by an ultraviolet laser layer by layer. This allows the user to go quickly from a CAD file to a physical three-dimensional manifestation of the part.

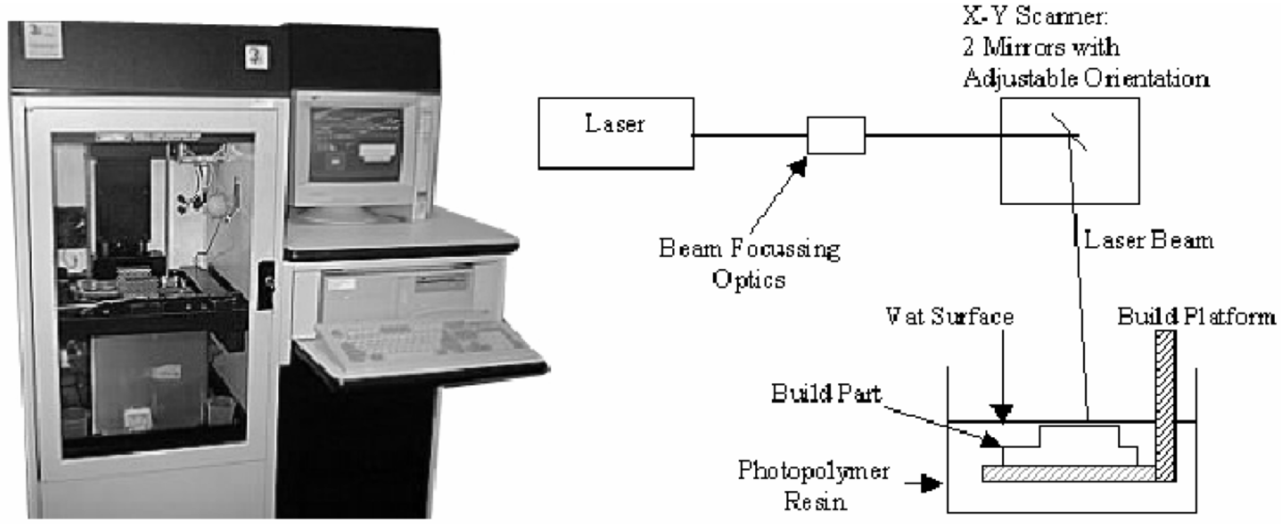


Figure 2. Stereolithography Apparatus (SLA 250) from 3D Systems with a Schematic of the Process

A photograph of a stereolithography apparatus (SLA) along with a schematic of the SL process is shown in Figure 2. A typical SLA consists of an ultraviolet laser, beam focusing optics, two galvanometers that each consist of a mirror oriented by a small motor, a vat of photo-polymer resin, and a build platform that raises and lowers the build part.

A typical build process is completed in the following manner. A laser generates a laser beam that is conditioned and refocused using beam focusing optics. The beam is then reflected by two mirrors that can each rotate about a single fixed axis. Using these two mirrors the beam can be directed to any point (x,y) on the vat surface. The resin at the surface of the vat is cured along the trajectory created by the laser. Thus, by scanning the shape of the part's cross section, one layer of the part is constructed.

The platform holding the part is then lowered by one layer (generally 0.05 - 0.2 mm) into the vat and the process continues with the next layer. Before the next layer is built, a recoating mechanism, in form of a blade, slides over the surface to distribute the liquid resin evenly on the surface of the part to be built.

Section 6.2 describes the construction of a functional model of the SLA 250 shown in Figure 2, including all mechanical and optical components, built in an SLA 250 machine.

2.3 Glossary of Terms

For the convenience of the reader the following summarizes the acronyms used throughout this chapter:

RP Rapid Prototyping – a method of fabricating a physical three-dimensional manifestation of a part from the CAD file of the part;

AF Additive Fabrication – is defined as building products in a process of adding material to the whole, as opposed to subtracting material (as is traditionally done). All of the following are examples of Additive Fabrication.

SL Stereolithography – is a layered manufacturing technique that builds 3D objects layer by layer using an ultraviolet laser and photosensitive polymer resin.

SLA Stereolithography Apparatus – is the acronym for a Stereolithography machine.

SLS Selective Laser Sintering – is a layered manufacturing technique that builds 3D objects layer by layer using a heat generating CO2 laser and powders made of materials including polycarbonate, nylon, and metal.

FDM Fused Deposition Modeling – is a layered manufacturing technique that builds 3D objects layer by layer using two nozzles which place ABS or polyamide (as the building material) and modified ABS (as the support material).

Note that British units of measurement are used frequently in this chapter, since these units are used with RP machines. Metric equivalents follow in parentheses.

3. Related Research on the Use of RP for Automation and Robotics

The traditional use of Rapid Prototyping is the fabrication of rigid parts. However, in recent years several research groups have discovered the potential of producing (robotic) mechanisms using RP, including Lalibert´e et al. (1999), Alam et al. (1999), Diez (2001), Rajagopolan & Cutkosky (1998, 1999) and Binnard (1999).

The fabrication of joints is a crucial first step for the fabrication of mechanisms using RP. Lalibert´e et al. (1999, 2000) present the generic design of joints using the FDM process. Joints for SL and Selective Laser Sintering processes are presented by Alam et al. (1999) and Won et al. (2000). Kataria (2000) discusses the fabrication and required clearances for various non-assembly prismatic joints and revolute joints for both SLA-250 and SLA-3500 machines. Diez (2001) discusses the fabrication of a very different type of joint for SL technology: compliant joints are considered, i.e. joints consisting of a single part that can be bent in some fashion.

Joints presented in Lalibert´e et al. (1999, 2000) require assembly after fabrication, while the joints presented in Alam et al. (1999); Won et al. (2000); Kataria (2000); Diez (2001) are fabricated in already assembled form. Rajagopolan & Cutkosky (1998) present a framework for the analysis of the effect of geometry gaps and clearances on mechanism assemblies in RP, which applies to assembly and non-assembly joint types.

Numerous complex passive and actuated mechanisms using the above joint types were successfully fabricated using RP, including a robotic hand (Won et al., 2000) and several parallel platform mechanisms (Lalibert´e et al., 2000). Additional examples are provided in this chapter.

Lipson & Pollack (2000) investigated the automatic design and manufacture of what they call "robotic lifeforms", where manufacture was by FDM. They programmed software to automatically design robots to move autonomously on a horizontal plane by utilizing evolutionary computation. The output of their software consists of STL files for input directly to an FDM machine. Joints were fabricated in already assembled form, and actuators and sensors were assembled to the robots after fabrication.

Successful insertion of embedded electronic components in RP during the build process was reported for electronic games already in 1992 (Beck et al., 1992). Several research groups are seeking to apply this idea, i.e. the insertion of components during the build process, to the fabrication of mechanisms in RP. By inserting actuators and sensors during the build process and using non-assembly joints, no assembly is required after the build process. A group at Stanford University developed a design framework for this purpose, termed "Design by Composition for Rapid Prototyping", see Binnard (1999) and related research by Binnard & Cutkosky (1998) and Cham et al. (1999). A group at Georgia Tech developed design strategies for the insertion of components for SL Technology, see Kataria (2000) and Kataria & Rosen (2000). Furthermore, the conceptual design of enhanced stereolithography machines, that may better support the insertion of components in the future, was discussed by Geving et al. (2000) and Geving & Ebert-Uphoff (2000).

4. Rapid Prototyping of Mechanisms Using Fused Deposition Modeling

This section presents the rapid prototyping of mechanisms using a commercially available CAD package and a FDM rapid prototyping machine. A database of lower kinematic pairs (joints) is developed using the CAD package and parameters of fabrication are determined experimentally for each of the joints. These joints are then used in the design of the prototypes where the links are developed and adapted to the particular geometries of the mechanisms to be built. Also, a procedure is developed to build gears and Geneva mechanisms. Examples of mechanisms are then studied and their design is presented. For each mechanism, the joints are described and the design of the links is discussed. Some of the physical prototypes built using the FDM rapid prototyping machine are shown.

4.1 Lower Kinematic Pairs and their Fabrication

The main distinction between the rapid prototyping of mechanical parts and the rapid prototyping of robotic mechanisms is the need to include moving joints in the latter. Joints are undoubtedly the most critical components of robotic mechanisms and the effectiveness of prototypes of robotic mechanisms is largely dependent on the ability to produce joints with adequate accuracy. Hence, the prototyping of moving joints is at the core of this work and will be addressed first.

Before mechanisms can be built, a database of lower kinematic pairs (joints) is developed and experimentally tuned. These joints will constitute the building blocks which will then be adapted and scaled to be included in specific mechanisms. These lower kinematic pairs are the revolute, Hooke, spherical, prismatic and cylindrical joints. These joints are the most critical parts of the mechanisms.

Because of the anisotropy of the material deposited layer by layer, the parts must be properly oriented to obtain the desired strength. This is not only true for joints but for all parts. This can be observed in Figures 1 (b) and 3, where the same part is properly oriented in Figure 1 (b) and not properly oriented in Figure 3. Deposition layers oriented perpendicular to a hole axis increase the strength of the hole. For example, the holes in the part of Figure 1 (b) are strong compared to the holes in the part of Figure 3. Deposition layers oriented perpendicular to a pin axis decrease the strength of the pin. For example, the pins at the right end of the part are strong in the part of Figure 1 (b) but fragile in the part of Figure 3. Since the joints involve moving contacts, the dimensional accuracy and the surface

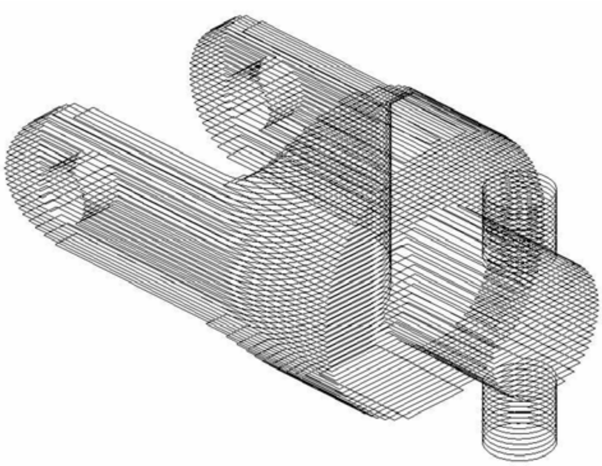


Figure 3. Example of a sliced part not properly oriented

finish are critical. The tolerance of the parts is ±0.005 inches (0.127mm), which is relatively large for moving joints. Also, it can be seen from experimentation that a cylinder built horizontally is oval instead of circular. To obtain proper performances, test parts are built with progressive clearances, assembled and tested. Moreover, because the parts are built in layers of 0.010 (0.254mm) inches, the surfaces that are not perfectly vertical will include "steps" associated with the layers and their surface finish will be rather poor in the directions not aligned with layers. Therefore, it is advantageous to keep the walls perfectly vertical or to keep sliding movements aligned with the layers. For example, as can be seen in Figures 1 (b) and 3, the holes and shafts are smoother when their axis is perpendicular to the orientation of the deposition layers.

As previously mentioned, mechanisms built with the FDM process are built in separate parts and assembled manually. Most of the joints are assembled using the elastic deformation of the material. Therefore, a compromise must be found between the stiffness of the joints and their ability to be assembled without rupture. The maximum elastic deformation has been found to be around 1.5%, but deformations of 3%—with some plastic deformation—are acceptable if necessary. Some features of the geometry of the parts are established using stress-deformation equations.

Finally, some joints may have very deep and narrow holes and it would be very difficult to remove the support material. Fortunately, with the FDM process, it is possible to make what is referred to as bridging. With appropriate settings, the hot thread of material is stretched and does not collapse for a short distance (up to 6 mm) when it is applied without support. Therefore, it is possible to build small ceilings without support. To establish the maximum bridging distance without collapse, a test part with different gap lengths is built without support in the gaps. This is illustrated in Figure 4. Note that the threads of the layer covering the gaps are oriented in the longitudinal direction of the part to make bridging possible.

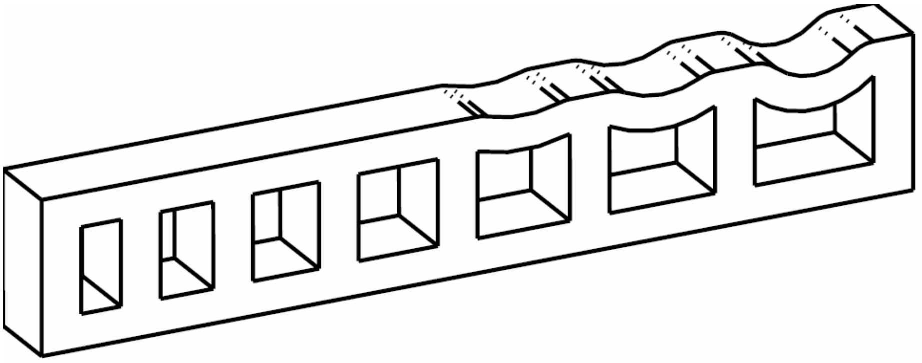


Figure 4. Bridging test : the bridging is correct up to the fourth gap

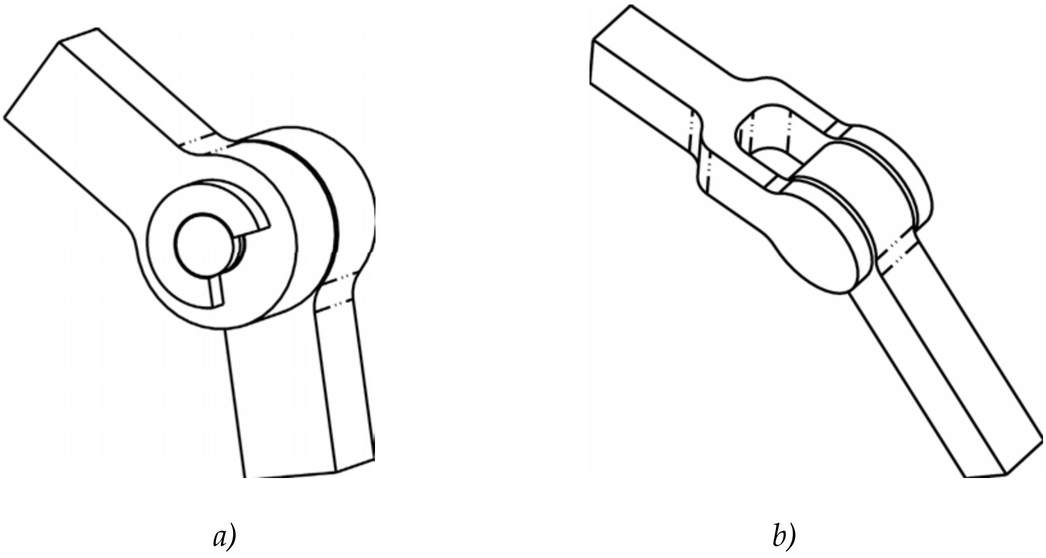
The database of joints is now described:

Revolute Joints

Two types of revolute joints are represented in Figure 5 (a) and (b).

The first type of revolute joint is made of a grooved shaft to be inserted in a hole and held by a snap ring. The snap ring can be bought or made by the rapid prototyping machine. It is preferable to build the part with the hole aligned vertically in order to obtain a stronger part and a rounder and smoother hole. It is better to build the shaft horizontally to improve its strength. However, it is preferable to build it vertically to produce a part with better smoothness and roundness. Most of the time, the shafts are built horizontally since strength is more critical.

The second type of revolute joint is made of a fork-shaped part with two small shafts and a hole. The fork is opened to fit the small shafts at the ends of the hole. Again, it is better to build the part with the hole vertically to obtain a stronger part and a smoother hole. Since the shafts are placed on each side of the part, they work only in shear stress, which is less demanding than the bending stress sustained by the first type of joint. Therefore, the shafts are built vertically. The shape of the fork is an important factor to obtain adequate stiffness while providing sufficient compliance for the assembly. Note that this type of joint is very smooth but cannot be rotated 360 degrees. Upon fabrication, the first type of joint is assembled by inserting the shaft in the hole and snapping the ring in the groove. The second type of joint is assembled by opening the fork and snapping the shafts of the moving part in the holes.



a) *b)*

Figure 5. Revolute joints: (a) type 1 (b) type 2

A Hooke joint is represented in Figure 6. The Hooke joint is composed of a small cross and two fork-shaped parts. Upon fabrication of the three parts, the joint is assembled as follows: the pins of the cross are inserted in the holes of the fork-shaped parts by opening the fork.

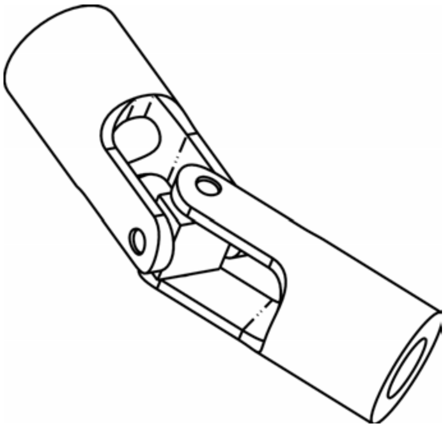


Figure 6. Hooke joint

The forks are built with the holes vertically oriented for strength and roundness. The pins of the cross are built horizontally for strength. Again, the shape of the fork is an important factor to obtain proper stiffness while providing sufficient compliance for the assembly. The Hooke joints can be suitably operated at ±45 degrees.

Spherical Joints

A spherical joint is represented in Figure 7. The spherical joint is made of a sphere at the end of a cylinder and a spherical cap segmented in four sections to allow some expansion during the assembly. The assembly of the joint is performed by pushing the sphere into the spherical cap. The cap is preferably built horizontally to give more strength to the sections. The sphere and its cylinder are preferably made of two parts. The cylinder is built horizontally for strength. The sphere is built perpendicular to the cylinder to keep the layers of the sphere and the layers the cap unaligned in all configurations to avoid any gripping. These parts are then assembled press fit or bound. The spherical joint can be operated in a cone of ±30 degrees.

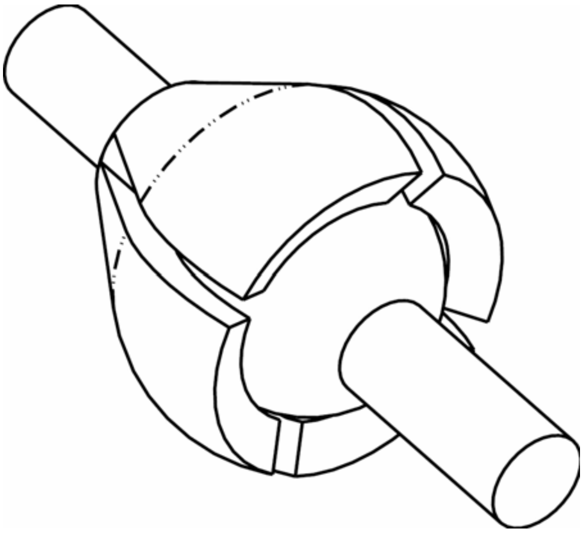


Figure 7. Spherical joint

Prismatic Joints

A prismatic joint is represented in Figure 8 (a). The prismatic joint is made of a square section and its corresponding square tube. In order to obtain proper strength, both parts must be built horizontally. Therefore, the tube should be filled with support material to support the roof of the square tube. Since the tube can be very deep and narrow, it would be very difficult to remove the support material and bridging is used. A groove is made along the bar, except at the ends. A small flexible stopper is made at the end of the tube and inserted into the groove to keep the motion of the bar in a certain range. The two parts are assembled as follows: the stopper is maintained open to allow the insertion of the bar in the tube. For some applications, it is interesting to have a prismatic joint that can keep a certain position even if there is some force applied on it (for instance, to avoid the collapsing of a prototype under gravitational forces). This can be accomplished by creating a wave along the sliding bar and a flexible rod on the tube. The rod will tend to stay in the trough of the waves, allowing to keep givenconfigurations. This is illustrated in Figure 8 (b).

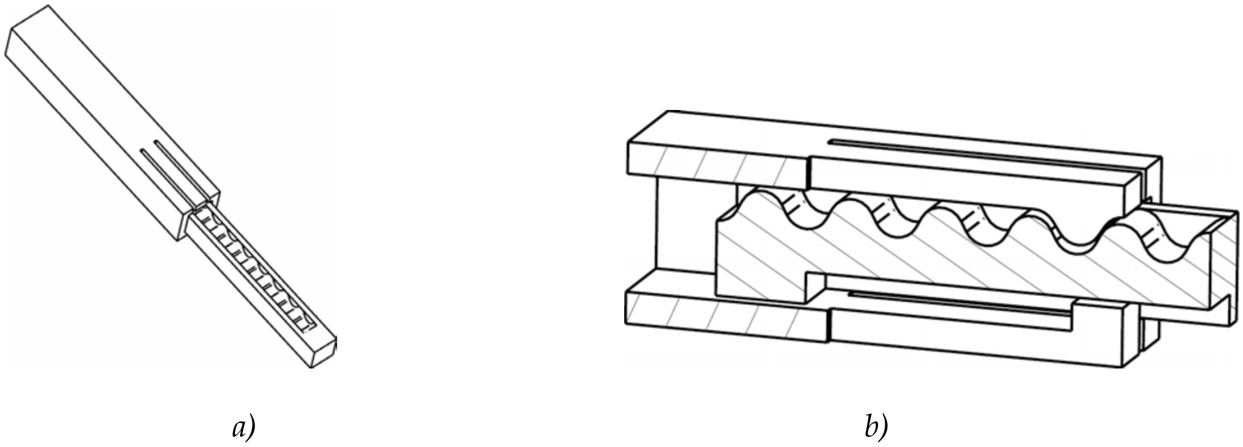


Figure 8. (a) Prismatic joint with wavy moving bar. (b) Zoomed cross section of the prismatic joint with the wave mechanism (top) and the stopper (bottom)

Cylindrical Joints

A cylindrical joint is represented in Figure 9. The cylindrical joint is made of a rod inserted in a tube. As for the prismatic joint, both parts must be built horizontally in order to obtain proper strength. For the tube, bridging is used to avoid use of support material. Since the roof is not planar, the shape of the tube is modified to allow bridging without affecting the desired properties. In order to stop the travel of the joint and allow a smooth rotation, a press-fit sleeve is added to the rod and a press-fit cap is added to the tube.

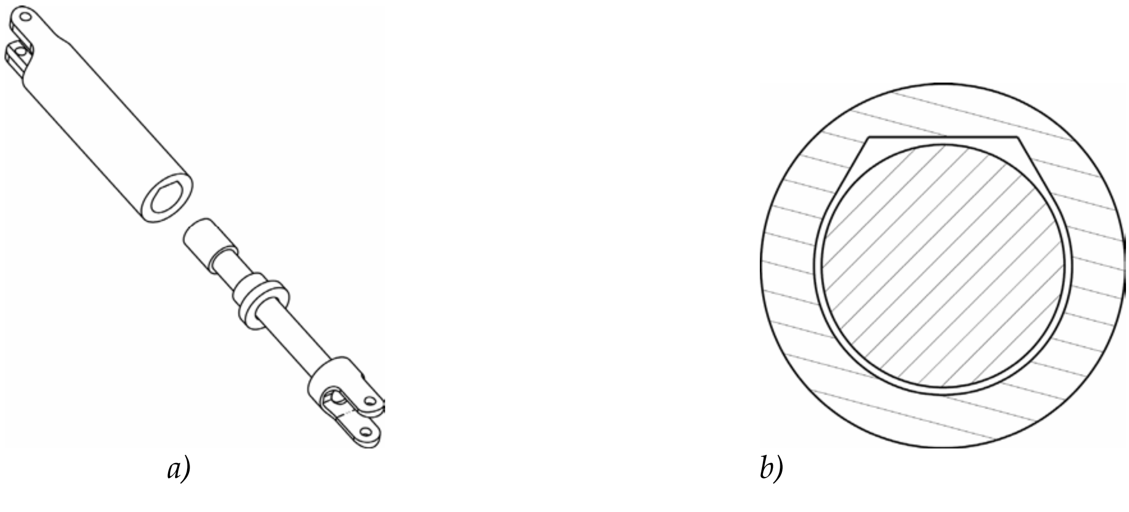


Figure 9. (a) Cylindrical joint with parts of Hooke joints at the ends. (b) A cross section of the cylindrical joint illustrates the shape of the tube to allow bridging

4.2 Rigid Assembly of Parts

In the fabrication of mechanisms, parts must often be rigidly assembled. Indeed, some parts are too large to be built in one piece and different features of a same part must sometimes be oriented in different directions to accommodate the anisotropy of the material. However, some assemblies should not be permanently mounted, in order to allow many versions of a part of an assembly to be exchanged. To assemble the parts permanently, glue or strong press fit has shown to be satisfactory. Some non-permanent assemblies have also been developed, as light press fit. Another example, the twist binder, represented in Figure 10, is useful to easily mount and unmount parts where the translating forces are important but the torque in at least one direction is weak or zero.

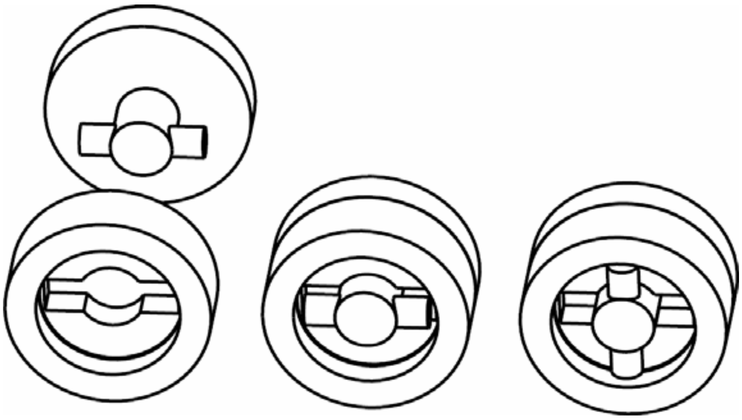


Figure 10. Twist binder : a T-shaped part is inserted in a slotted part, then turned 90 degrees to a press-fit locked position

4.3 Gears

This section discusses the fabrication of spur gears with teeth made using the commonly used involute profile. The theory related to involute gearing will be shortly summarized for the rapid prototyping needs. The following values are known: the diametral pitch Pd, the pressure angle Φ and the number of teeth N or the pitch diameter Dp. These two last values are related by Pd = N/Dp. The following values are then computed : outside diameter is Do = Dp + (2/Pd), the root diameter is Dr = Dp − (2.5/Pd), the base diameter is Db = Dp cos Φ and the thickness of the tooth is t = π/2Pd. Note that while Di refers to diameter values, Ri refers to the corresponding radius value. For more details refer to Oberg et al. (1988). To create the gear teeth in the CAD package, a skeleton curve of the involute profile must first be defined parametrically. Referring to Figure 11, the involute curve can be described by

$$X = -Rb \sin(\alpha - \beta) + Rb\_ \cos(\alpha - \beta) \tag{1}$$

$$= Rb \cos(\alpha - \beta) + Rb\_ \sin(\alpha - \beta) \tag{2}$$

where Rb is the base radius, α is the parametric angle and β is an offset angle related to the thickness t of the tooth. The parametric angle α varies from 0 to αo, which is related to the outside diameter of the gear Do. The values of β and αo are found from the following equations.

$$= \pi/2N + \tan(\Phi) - \Phi - V/Rp \tag{3}$$

$$\alpha o = \tan[\arccos(Db/Do)] \tag{4}$$

where V is an offset to allow sufficient backlash. Good results are obtained with V =0.001 inch (0.0254 mm).

With these equations and some experiments, a procedure has been developed for the CAD package.

1. Extrusion of a circle of diameter Db if Db > Dr or Dr if Dr > Db.
2. Creation of a coordinate system (see Figure 11).

3. Creation of the parametric involute curve found from the preceding equations.
4. Mirror copy of this curve from the plane YOZ.
5. Creation of an outside diameter circle.
6. Extrusion of the tooth from the parametric involute curves and the outside diameter circle.
7. Copy of this tooth each 360/N degrees.
8. If Dr < Db, creation of a cut between the teeth to Dr and copy of this cut.
9. Creation of fillets, holes, hubs, etc.

A similar and adapted procedure can be used to obtain internal spur gears.
Regarding the orientation of the gears in the prototyping machine, it is advantageous to keep their axis vertical for smoothness of operation and strength. With proper adjustment of the parameters of the rapid prototyping machine, gears with diametral pitch as fine as 32 (approximately 0.8 module) can be properly fabricated and used. Future work includes the development of helical, miter and bevel gears.

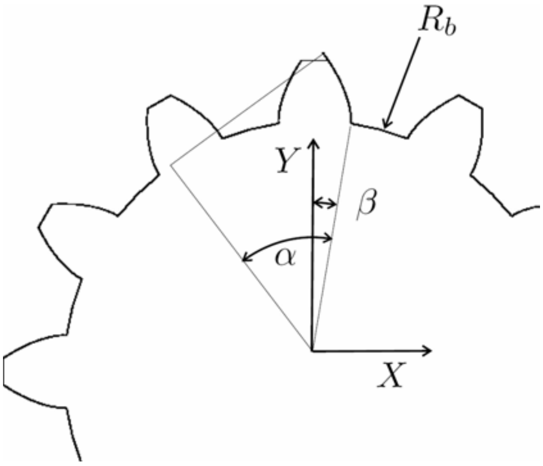


Figure 11. The involute profile of a gear tooth

4.4 Examples of Mechanisms Using FDM

To build a complete mechanism, one has to create the appropriate links and include the joints presented previously. Because of the nature of the rapid prototyping process, these links can be of almost any shape. Here, the anisotropy of the material is the main factor to be considered in order to obtain links of appropriate strength. This section presents some examples of mechanisms built in the Robotics laboratory at Laval University using the joints previously presented. In addition to the mechanisms presented here, the following mechanisms have been built: four-bar and five-bar spherical mechanisms, four-bar spatial mechanisms, serial manipulators, 3-DOF planar parallel mechanism, 3-, 4- and 5-DOF spatial parallel mechanisms and several others.

Simple Mechanisms

Several simple mechanisms have been built. The Bennett linkage is a good example. A prototype of the Bennett mechanism (Phillips, 1990) — a well known spatial four-bar overconstrained linkage — has been built for classroom demonstration purposes. The CAD model and the prototype are presented in Figure 12 (a) and (b).

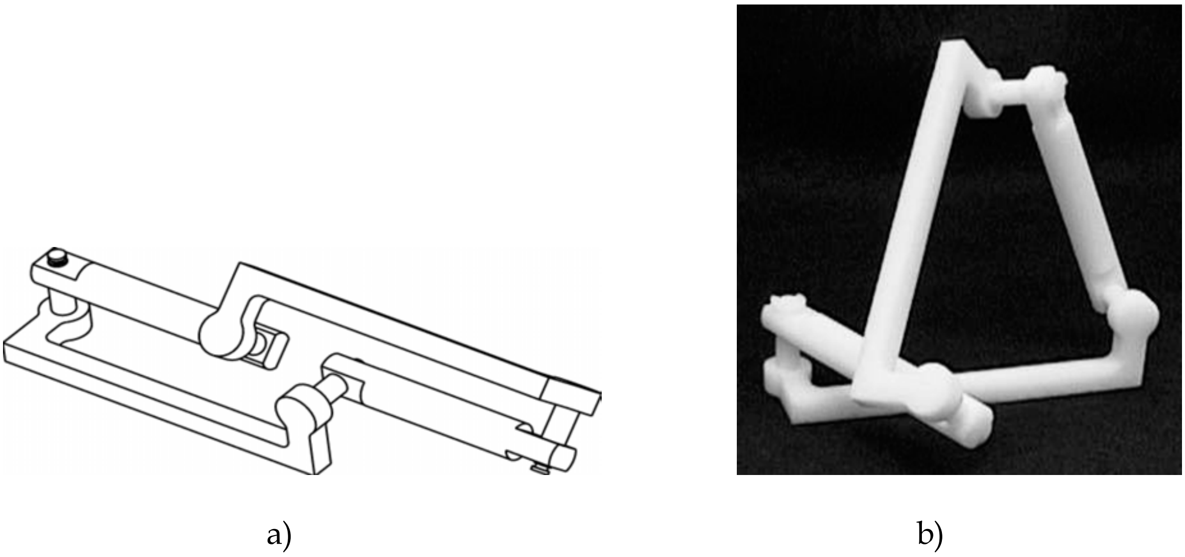


Figure 12. Bennett mechanism: (a) CAD model (b) prototype

Three-DOF Spherical Mechanism: the Agile Eye

The Agile Eye (Gosselin & Hamel, 1994), represented in Figure 13 (a) and (b), is an example of mechanism involving complex geometries that are easily made with rapid prototyping. Each of the three legs has three revolute joints. Note that the bottom link of the legs is made of two parts in order to align the layers with the general orientation of the legs.

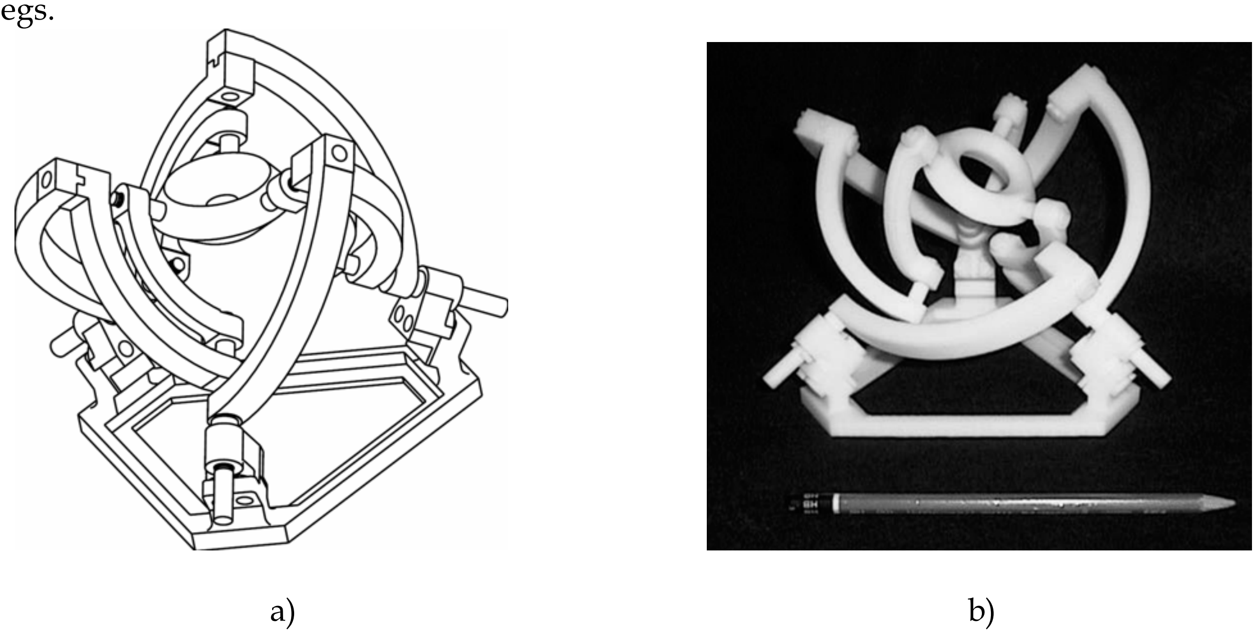


Figure 13. Agile eye: (a) CAD model (b) prototype

Three-DOF, Four-DOF, Five-DOF and Six-DOF Platforms

Several three-, four-, five- and six-DOF parallel platforms have been built. For instance, the six-DOF (Gough-Stewart) (Dasgupta & Mruthyunjaya, 2000) platform illustrates the use of a variety of the joints developed above (Figure 14 (a) and (b)). Each of the legs comprises a Hooke joint, a prismatic joint and a spherical joint, giving 6 DOF. The ends of the legs can be assembled to the upper and lower platforms using a twist binder. With these binders, the legs and the platforms are modular. Therefore, it is easy to try different geometries of platforms or different types of legs without having to rebuild the whole mechanism for each version. The plastic prototypes of parallel robotic mechanisms are very useful to visualize the singularities. In general, singularities can be found from mathematical analysis but are often difficult to physically visualize.

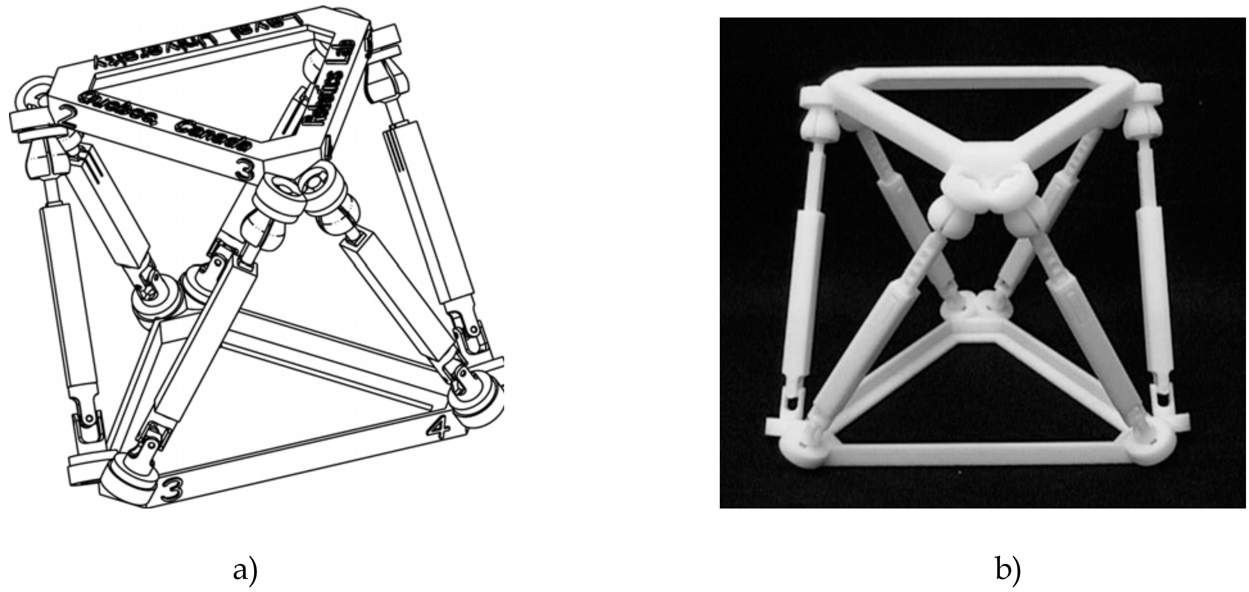


Figure 14. Gough-Stewart platform: (a) CAD model (b) prototype. Note the use of waves in the prismatic joints to avoid collapse of the platform

Example of Geared-Pair Mechanism: Planetary Gear System

The rapid prototyping of gears is useful for the development of complex systems. For example, a planetary gear system is presented in Figure 15. This planetary gear system is made with a 2-inch (50.8 mm) internal gear and 24 diametral pitch (approximately 1.0 module) teeth. The prototype works very smoothly. Gears with 48 diametral pitch (approximately 0.5 module) teeth or finer are difficult to obtain because of the resolution of the machine.

Examples of Actuated Mechanisms

Some of the prototypes built are motorized, as the novel 6-DOF mechanism (Gosselin et al., 1999) represented in Figure 16 (a) and (b). Since larger stresses can be generated at the actuator shafts, an aluminum part is mounted on the shafts and then inserted in the ABS part. All the other joints of the mechanism are in ABS polymer and could

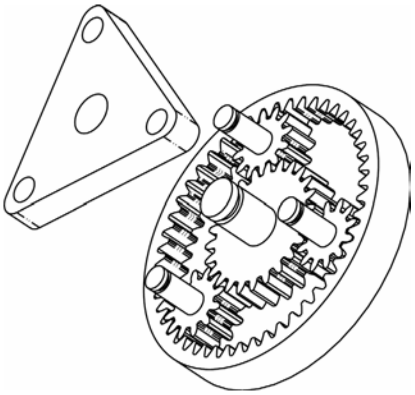


Figure 15. CAD model of a planetary gear system

easily sustain the induced forces even when the end-effector of this mechanism experienced accelerations of more than one g. Another example is a 3-DOF spherical haptic device with only pure rotations of the end effector, represented in Figure 17 (a) and (b). In this prototype, metal parts and bearings are used in combination with plastic parts. The main advantage of rapid prototyping in this example is the complexity of the geometry of the moving parts, some of which would be extremely difficult to machine using conventional processes. Although satisfactory results have been obtained with this spherical haptic device, the compliance of the plastic parts remains an important limitation if high performance is required.

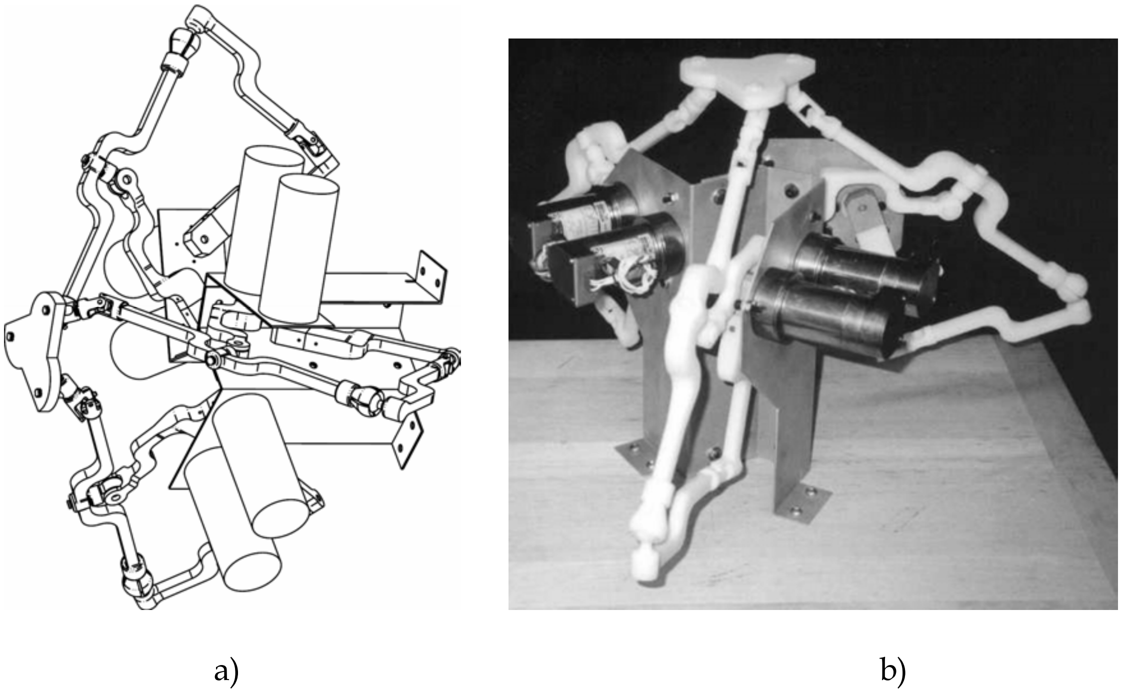


Figure 16. Novel six-DOF mechanism: (a) CAD model (b) prototype

Prototyping of a Robotic Hand

The rapid prototyping technique presented above has been especially useful in the design of a robotic hand (Lalibert´e & Gosselin, 2001, 2003), illustrated in Figure 18.

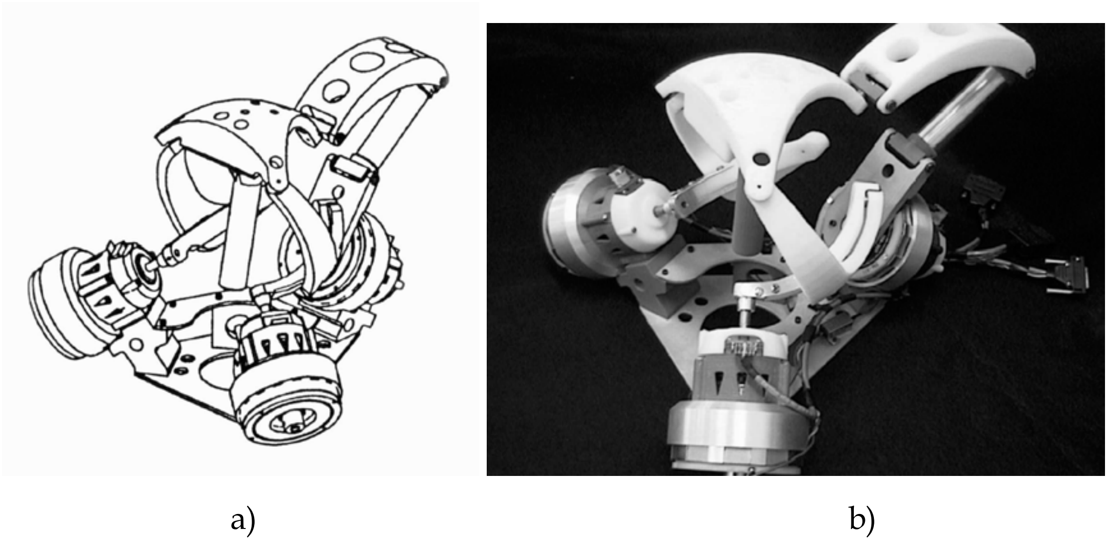


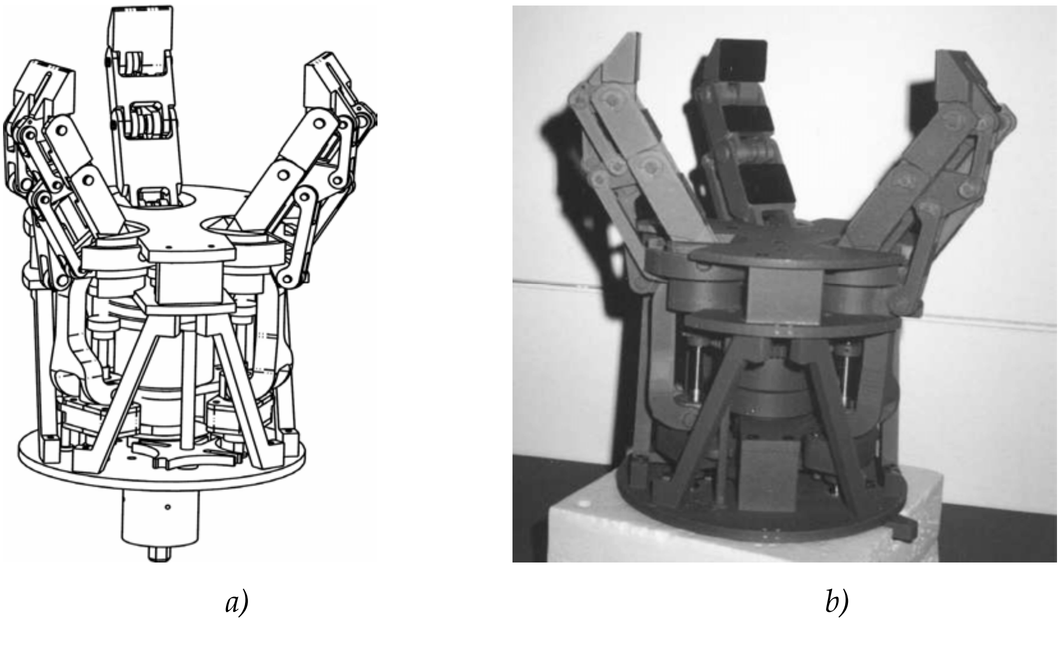
Figure 17. Three-DOF spherical haptic device: (a) CAD model (b) prototype

Several novel features are included in this versatile grasping hand, which has ten degrees of freedom but only two actuators.

Plastic models of the new features have been built during the first steps of the design process in order to validate their functionality or to help choosing among several possible solutions which were difficult to compare by simulation. Among others, prototypes of fingers were very useful to check possible mechanical interferences between the numerous parts of the compact assembly. Also, the prototyping of a new geared differential system has been useful for validation and demonstration.

The prototype of the robotic hand has been built almost entirely of plastic except for off-the-shelf metal screws, nuts and springs. The number of parts, without the screws, retaining rings and bushings, is approximately 200. The prototype has been tested to validate its functionality and real grasping tasks were performed using actuators. If a part of a sub-system was not satisfactory, it was modified on the CAD software, rebuilt and assembled, generally in the same day, making the tuning cycle very short. The mechanical efficiency of the system has been measured and compared with the simulation results in order to validate the theoretical model of the grasping forces. In order to identify the weakest parts of the system, the hand has been overloaded. The resulting broken part could then be modified. As should be expected, the load that could be applied on the plastic prototype was much lower than for the metal version. However, knowing the relative strength of the plastic and metal, test data could be extrapolated.

For instance, one particular part with a very complex shape was tested mechanically using the plastic prototype since its resistance was difficult to obtain through simulation. For the fabrication of the real metal hand, the plastic prototype was a very good complement to the drawings. First, the design of the metal hand was completed with much more confidence from the validation and testing of the plastic prototype. Also, the plastic parts were used to help the machinists understand the most complex geometries before they fabricated the metal parts.



a) *b)*

Figure 18. Underactuated 10-dof robotic hand: (a) CAD model (b) prototype

The prototype has been a remarkably useful tool for explanation and demonstration purposes. Also, the plastic prototype made it possible to apply for a patent with confidence before the construction of the metal version was completed.

Overall, the plastic prototype allowed to validate the new concepts, perform tests (even partially destructive), present the idea to decision makers and submit a patent months before the real metal version was ready.

5. Fabrication of Joints Using SL Technology

Analogously to mechanism prototyping using FDM, the fabrication of joints is also the essential first step for mechanism prototyping using SL and is discussed below.

5.1 Joints Originally Designed for FDM Machine and Built in SL Machine

The group at Universit´e Laval made the CAD files for several of the joints presented in the previous section available to the group at Georgia Tech. The following joints, which had been designed for fabrication in the FDM 2000 machine at Laval, were then built in the SLA-250 machine at Georgia Tech: the revolute joint of type 2 shown on the right of Figure 5, the Hooke joint shown in Figure 6 and the spherical joint shown in Figure 7.
The revolute joint and the spherical joint were fabricated successfully in the SL machine by simply adjusting the tolerances, i.e. the size of the gaps between moving parts, to the SL machine. For the Hooke joint some additional changes were necessary: since the stiffness for the parts made in the SLA-250 (using Resin SOMOS 7110) is much higher than the stiffness of the parts made in the FDM machine (ABS polymer), it was impossible to bend the arms of the Hooke joint enough to assemble the joint, i.e. to slide the center block with the four pins into its proper position. This problem was overcome by changing several dimensions of the parts to make it more flexible and by adding a small indentation at the inside of the joints to guide the pins of the block into its proper location. In any case, only small changes were necessary to use the joints successfully in the SL machine. This demonstrates the universality of the database presented in Section 4. for a wide variety of Rapid Prototyping processes.

5.2 Non-Assembly Joints of Conventional Type

All joints presented in Section 4. consist of several parts that must be assembled after fabrication. The high resolution of SL machines encouraged us to investigate the fabrication of joints that do not require assembly. As a first step we considered joints of the conventional type, i.e. joints consisting of two or more parts moving with respect to each other, but built in already assembled form. An example is the prismatic joint shown in Figure 19, where the sliding bars were built inside the guides in the SL machine in a single step. Clearances must be chosen large enough to ensure that the bars do not adhere to the guides, and any liquid resin trapped in the gaps must be removed by pressured air before post-curing is applied. No support structures were located in the prismatic joint regions.

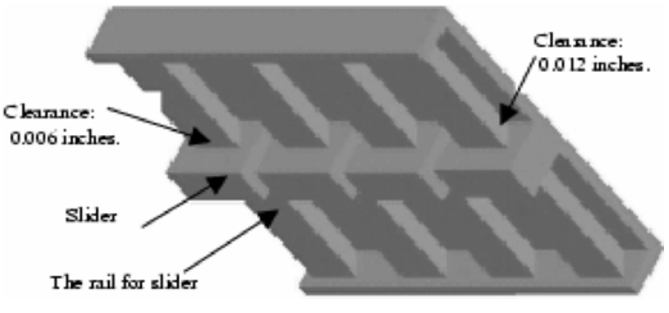
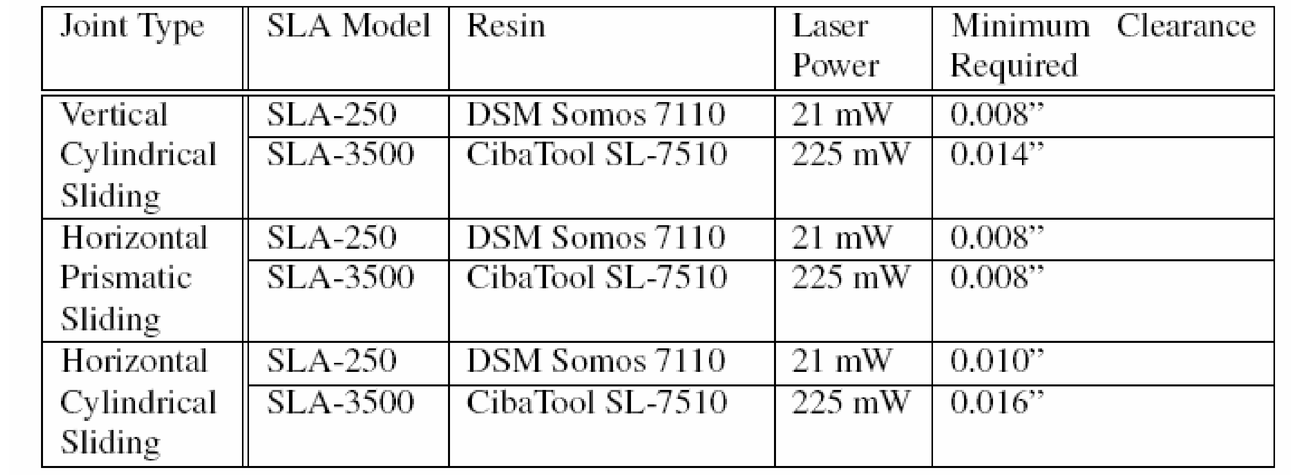


Figure 19. Horizontal Prismatic Joint Experiment

To test SL build limitations for horizontal prismatic joints, the part shown in Figure 19 was utilized. A dovetail cross section joint shape was used. Clearance within the joint was increased from 0.006 (0.152mm) to 0.012 inches (0.305mm) in increments of 0.002 inches (0.051mm). The parts were cleaned with TPM and alcohol before postcuring, and care was taken that the resin in the joints was removed completely. The experiment was conducted in both Georgia Tech SL machines, the SLA-250 and SLA- 3500. The SLA-250 was working at 21 mW laser power while SLA-3500 was working at 230 mW laser power. A similar experiment was performed for vertical cylindrical prismatic joints. Various clearances were used, increasing from 0.006 inches (0.152mm) to 0.016 inches(0.406mm) in increments of 0.002 inches (0.051mm). The results obtained from these experiments are summarized in Table 1.



Joint Type	SLA Model	Resin	Laser Power	Minimum Clearance Required
Vertical Cylindrical Sliding	SLA-250	DSM Somos 7110	21 mW	0.008”
	SLA-3500	CibaTool SL-7510	225 mW	0.014”
Horizontal Prismatic Sliding	SLA-250	DSM Somos 7110	21 mW	0.008”
	SLA-3500	CibaTool SL-7510	225 mW	0.008”
Horizontal Cylindrical Sliding	SLA-250	DSM Somos 7110	21 mW	0.010”
	SLA-3500	CibaTool SL-7510	225 mW	0.016”

Table 1. Clearances for Kinematic Joints

5.3 Non-Assembly Joints of Compliant Type

All the joints discussed so far in this chapter are of a conventional form in the sense that they consist of two or more pieces that are separated by a gap. In contrast, a compliant joint consists of a single piece that can be bent in some fashion.

Compliant joints have been evaluated as a substitution for traditional joints by many researchers, including Howell & Midha (1994, 1996), Goldfarb & Speich (1999), Speich & Goldfarb (2000), Frecker et al. (1997) and Paros & Weisbord (1965). Typical disadvantages of compliant joints include limited range-of-motion and non-ideal kinematic behavior. Advantages include absence of Coulomb friction, no backlash and ease of fabrication (due to monolithic structure).

Goldfarb & Speich (1999) identified the “compound split-tube joint”, shown in Figure 20, as a suitable compliant implementation of a revolute joint. The compound splittube joint exhibits the properties of traditional hinges in many ways: it rotates about a relatively constant axis, allows rotation about this axis easily, and restricts translation and rotation about any other axes. In contrast, most other types of compliant joints are compliant in more axes than the intended axis (Goldfarb & Speich, 1999).

While Goldfarb and Speich considered joints manufactured of steel, we succeeded in reproducing the same basic properties for split-tube joints built using Stereolithography. The Stereolithography fabricated part is shown in Figure 20 along with the CAD model of the compound split tube. These compliant joints are used as the joints in the two mechanisms presented in the following subsection.

5.4 Examples of Mechanisms

Snake-like Mechanism Using Non-Assembly Joints of Conventional Type

Figure 21 shows the first development stage of a miniature snake-like tendon-driven mechanism, inspired by the underactuated snake-like mechanisms introduced by Hirose (1993). The type of joint used in a "point-contact axle" developed specifically for non-assembly fabrication of miniature mechanisms in SL machines (for details, see Miller (2001)). The mechanism is 10 cm long (fully outstretched) and consists of 6 links that are connected by point-contact axles. Pulleys are also already built in. The snake was built in a single step requiring no assembly. After some further refinements, tendons will be added to the mechanism.

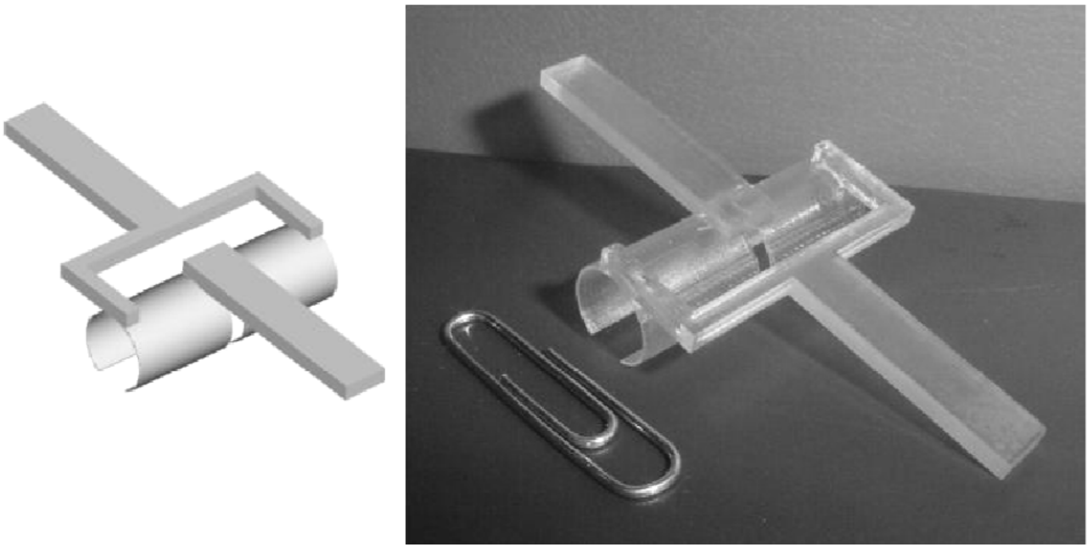


Figure 20. Virtual and Fabricated Compound Split Tube Model

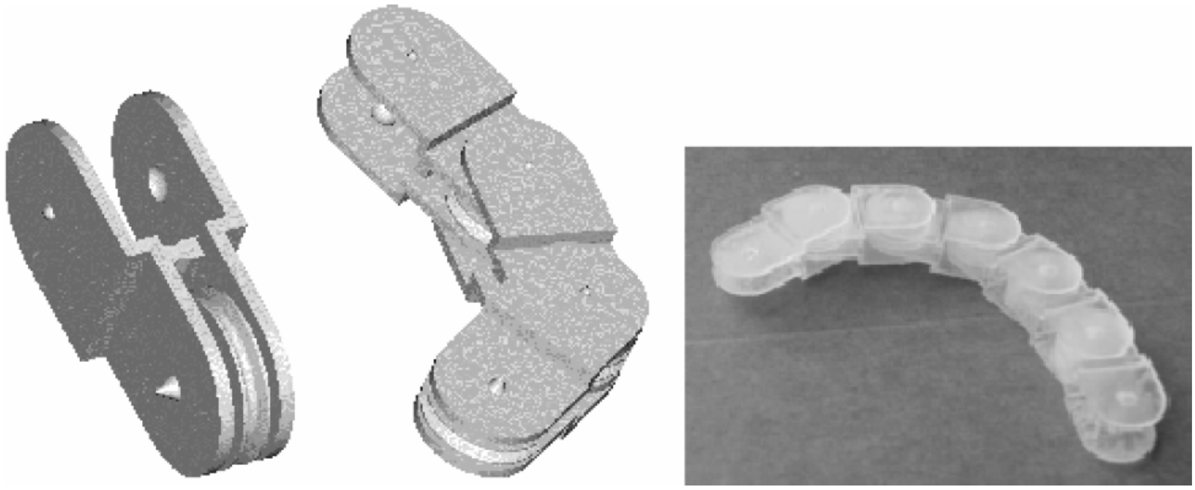


Figure 21. Snake-like mechanism with built-in joints and pulleys. CAD model of a single segment (left) and of two connected segments (center) and prototype (right)

Actuated Mechanisms Using Compliant Joints

Two mechanisms were developed to demonstrate the use of compliant joints to build miniature, actuated, robotic devices: the 2R robot and the robotic hand. Both of these

mechanisms utilized Shape Memory Alloy (SMA) wires for actuation, and show the potential for building non-assembly miniature robotic devices with Additive Fabrication.
The 2R robot, seen in Figure 22, is a small planar robot with two degrees of freedom. It is approximately 3.5 cm tall with a 2 x 1.5 cm base. Two SMA wires produce motion and two compound split tubes act as joints. The attachments for the actuators are already built into the SL part, so that the SMA wires can be attached within minutes after fabrication. The SMA wires can be seen in Figure 22 and are inserted using socalled spring slots to reduce any slack in the wires in its zero position (for details see Diez (2001).)

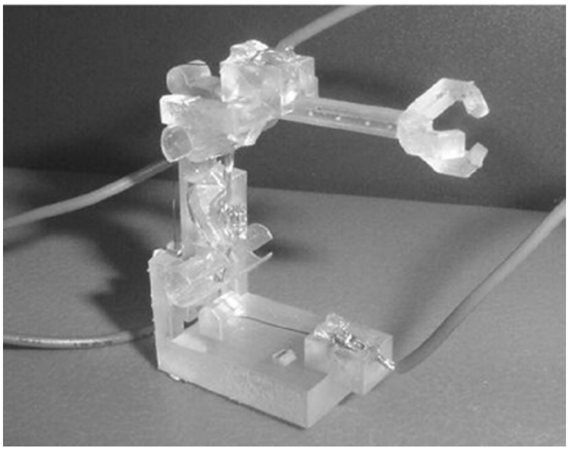


Figure 22. 2R Robot

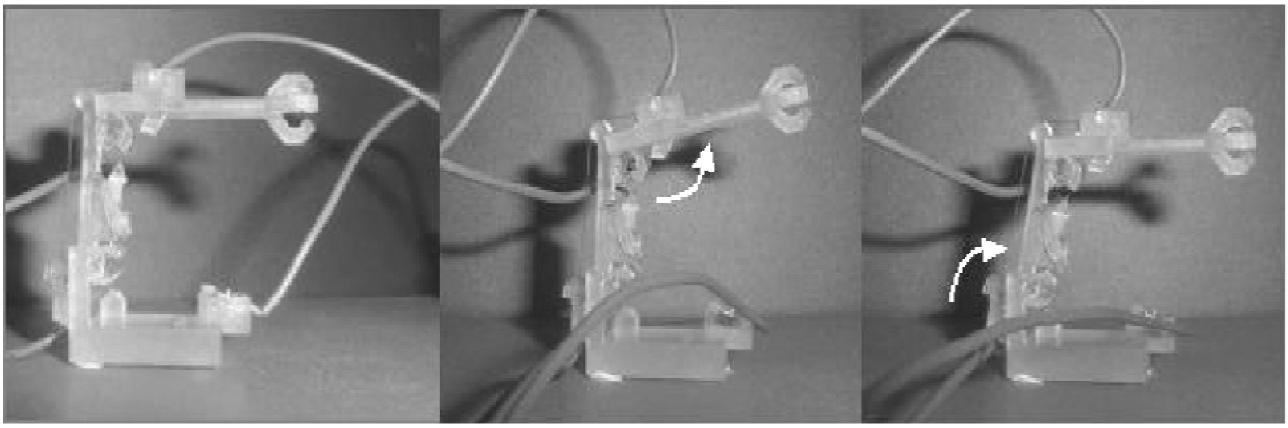


Figure 23. 2R Robot Actuated

The motion achieved by the SMA wires can be seen in Figure 23. In the first frame, no actuation has occurred. In the second frame, the upper link has been actuated as indicated by the arrow. Finally, the third frame shows the lower link being actuated while the upper link is also actuated. Although this motion is small, it shows clear potential for tasks such as micro assembly.

Robotic Hand with Compliant Joints

Won et al. (2000) developed a robotic hand fabricated in SL that uses non-assembly joints of the conventional type. A different design of a robotic hand fabricated in SL is shown in Figure 24 which uses only compliant joints. There are 14 compliant joints in the 5 fingers (each finger has 3 joints and the thumb has 2).

Each finger was built as a single part already including all joints. While it would have been possible to build the whole hand as a single part, we preferred to build it as six separate parts (5 fingers and the palm) for easy maintenance. That way, if a finger breaks, it can easily be replaced. Again Shape-Memory Alloy wires are used as actuators. A total of 9 independent wires are used in the hand: The thumb is actuated with one SMA wire, while the fingers are actuated with two each. The knuckle joints are independently actuated and each finger's top two joints are actuated jointly, i.e. each finger is underactuated by one degree-of-freedom.

The wires are inserted both by spring slots (5 wires) and by sliders (4 wires) that remove all slack from the wires when the wires are inserted (for details see Diez (2001)). Similarly to the 2R robot, the attachment slots for the wires are already built in and the wires can be added within a few minutes after fabrication. Each finger on the hand moves a cumulative 50o. The hand is sized at near human size for demonstration purposes.

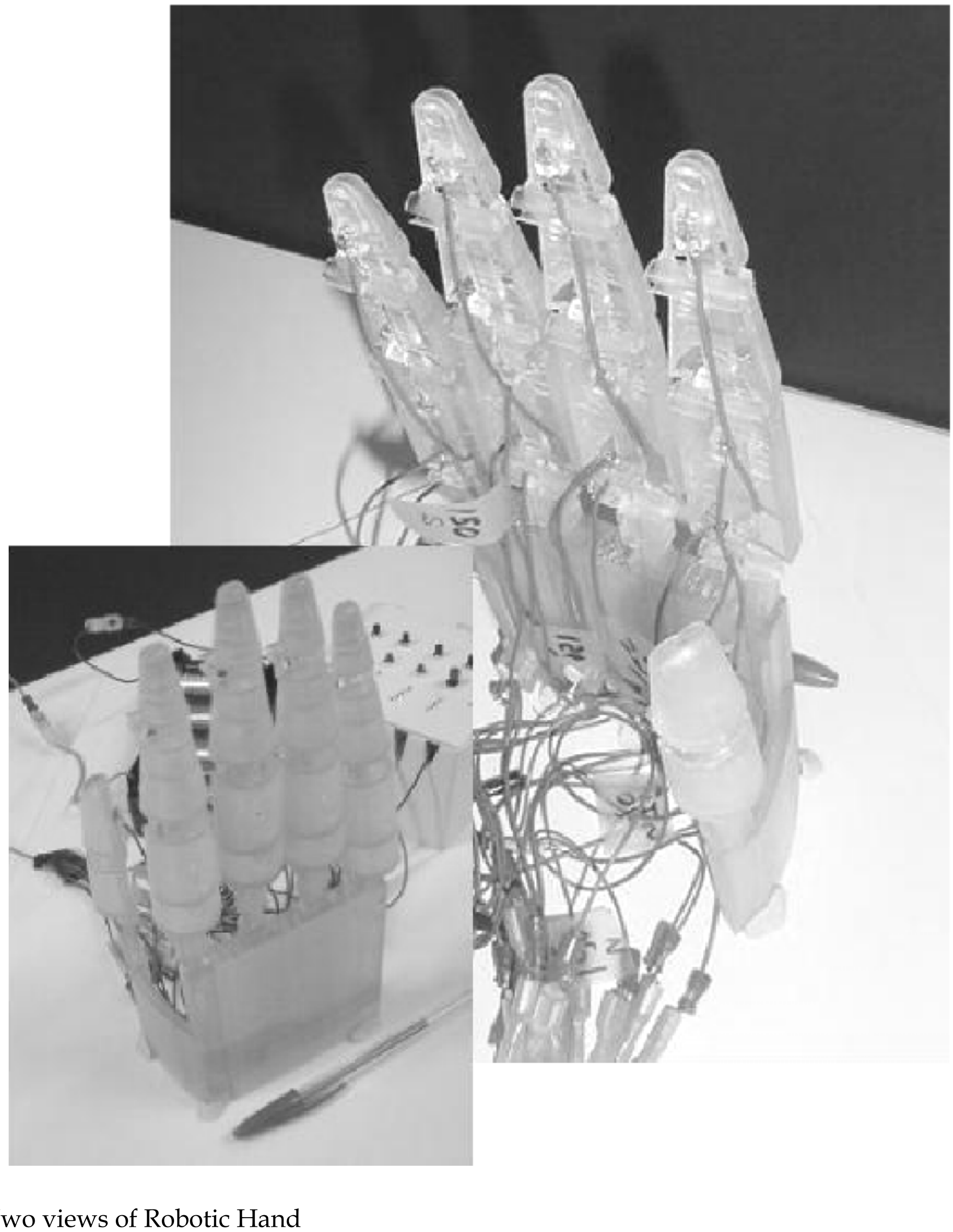


Figure 24. Two views of Robotic Hand

For this type of application the compliant joints have an additional advantage: Since Shape-Memory Alloy Wires only provide force in one direction, a return force is required to return them to their original shape when they cool down. The compliant joints act as joints with built-in springs and thus provide the required return force for the wires. Similarly, the inherent spring character of the compliant joints is also of advantage when prototyping other mechanisms that employ springs to return to a neutral position, such as various tendon-driven robotic mechanisms.

6. Novel Design Possibilities of Additive Fabrication for the Prototyping of Mechanisms

As discussed earlier, the main goal of prototyping at an early stage of experimentation is to quickly achieve a prototype that provides the basic functionality of the desired system. Implementation details are usually irrelevant at this stage, allowing for use of RP-specific short-cuts in the implementation to yield faster turn-around time. This section explores one of the capabilities unique to Additive Fabrication that may be useful for this purpose, namely the possibility to include inserts during the build process, thus eliminating the design of fasteners and the assembly process. SL technology is used to exemplify these unique capabilities.

6.1 Strategies for Insertion of Actuators and Sensors in SL

In the construction of functional prototypes, it is often advantageous to embed components into parts while building the parts in SL machines. This avoids post-fabrication assembly needs and can greatly reduce the number of separate parts that have to be fabricated and assembled. Furthermore, SL resins tend to adhere well to embedded components, reducing the need for fasteners. However, these advantages are not without limitations. Some part shapes must be modified to accommodate the embedding of components during builds, and the SL build process must also be modified. The advantages and limitations are explored in this subsection.

We have fabricated many devices with a wide range of embedded components, including small metal parts (bolts, nuts, bushing), electric motors, gears, silicon wafers, printed circuit boards, and strip sensors. These components are representative of those necessary for many types of prototype robotic mechanisms. Some care and preparation is often needed to embed some components, particularly if resin could contaminate the component or make the component inoperable when solidified. Device complexity is greatly facilitated when the capability to fabricate kinematic joints is coupled with embedded inserts since functional mechanisms can be fabricated entirely within the SL vat, greatly simplifying the prototyping process.

Figure 25 shows one example of a mechanism with four embedded metal components: two bushings, one leadscrew, and a nut that is embedded with the elevator (to mate with the leadscrew). Additionally, the ″leadscrew-top″ fits into the top of the leadscrew, but was assembled after removal from the SL machine.

A second example is a model of the recoating system in a SL machine, shown in Figure 26. As discussed in Section 2.2, the recoating mechanism of a SL machine consists of a blade that slides over the surface of the vat after a layer has been completed, to distribute the liquid resin evenly on the surface of the part to be built. In this case, two components were embedded during the build, a rack gear and a motor and speed-reducer assembly. The recoating blade slides relative to the stationary structure via two prismatic joints, fabricated as assembled.

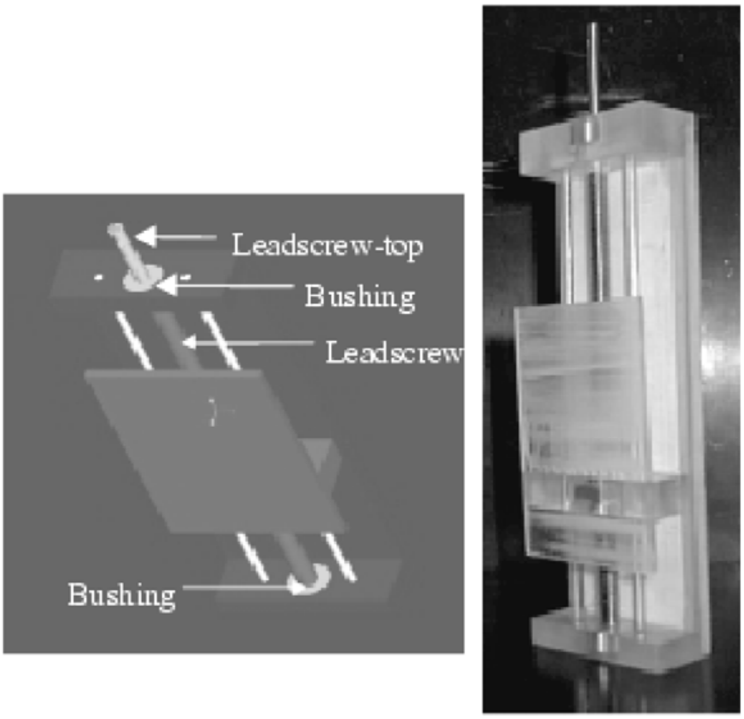


Figure 25. Elevator Model with 4 Embedded Components

Generally electric motors should be coated with wax or similar substance so that the liquid SL resin does not penetrate themotor housing and cause the motor to seize. Otherwise, no modifications to any of these components is required.

The tolerances for the inserts vary depending on size, load capacity, length of contact surface, type of joint, orientation, layer thickness etc. Though more study is needed in this area, the basic tolerances that we have identified are indicated along with the experiments below.

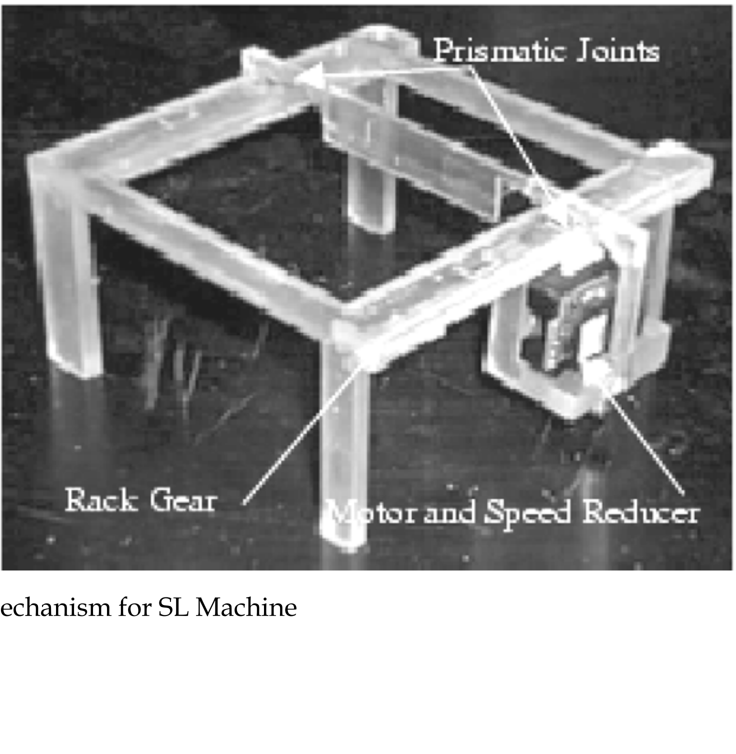


Figure 26. Recoating Mechanism for SL Machine

The fits for SL depend heavily on the surface roughness of the insert surface. Table 2 indicates the dimensions for clearance, transition and interference fits between inserts and SL parts. It should be noted that it is possible to insert components during the build for clearance and transition fits, but it is extremely difficult to insert a component having an interference fit since it requires considerable force that may damage the part or break the support structures skewing the existing build.

Figure 27 explains the critical dimensions specified in Table 2 that summarize thefits, based on our experiments. However, it is possible that tolerances and fits will vary on different machines, different laser powers, and for different resins.

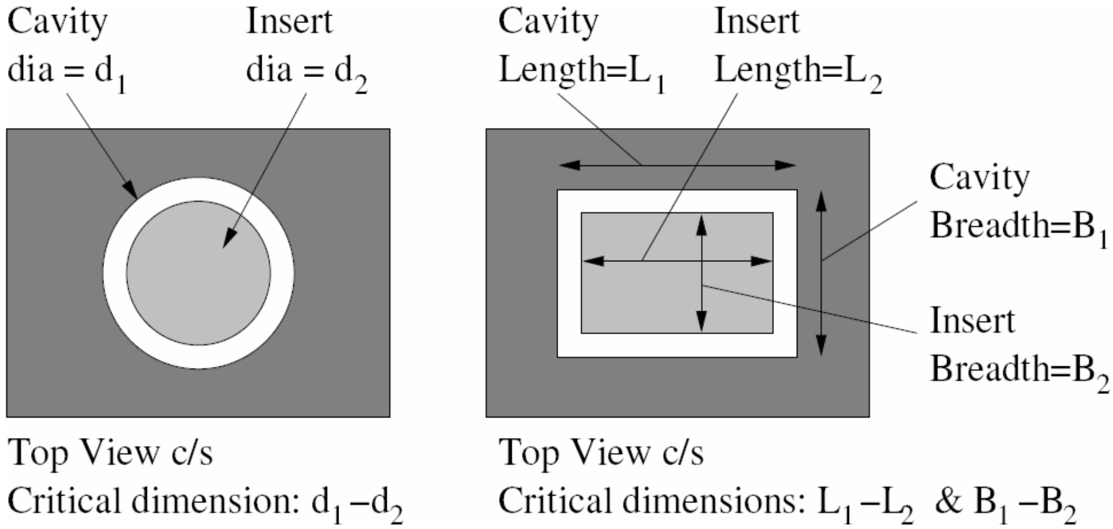
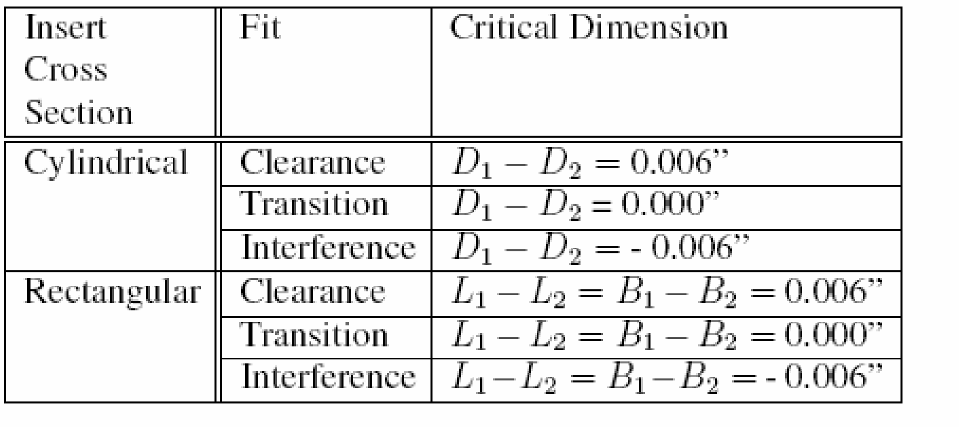


Figure 27. Critical Dimensions for Fits between Inserts and SL Parts



Insert Cross Section	Fit	Critical Dimension
Cylindrical	Clearance	$D_1 - D_2 = 0.006$”
	Transition	$D_1 - D_2 = 0.000$”
	Interference	$D_1 - D_2 =$ - 0.006”
Rectangular	Clearance	$L_1 - L_2 = B_1 - B_2 = 0.006$”
	Transition	$L_1 - L_2 = B_1 - B_2 = 0.000$”
	Interference	$L_1 - L_2 = B_1 - B_2 =$ - 0.006”

Table 2. Fits Between Inserts and SL Parts

6.2 Functional Model of an SLA-250 Built in an SLA-250

The most complex experiment performed at Georgia Tech was the construction of a model of the SLA-250 machine shown in Figure 28. The mechanical, optical and recoating subsystems of the original SLA 250 (discussed in Section 2.2 and shown in Figure 2) were modeled and fabricated in the SLA 250 at 1 : ¼ scale. The model includes all the components of the original machine:

- The mechanical platform with the vat that can be raised and lowered, using a leadscrew and a motor;

- The recoating blade that moves horizontally, using a second motor and a pair of sliding contacts;
- A small laser pointer that emulates the laser of the original machine;
- Two galvanometers, i.e. mirrors mounted onto motors, that redirect the laser onto any point on the vat surface.

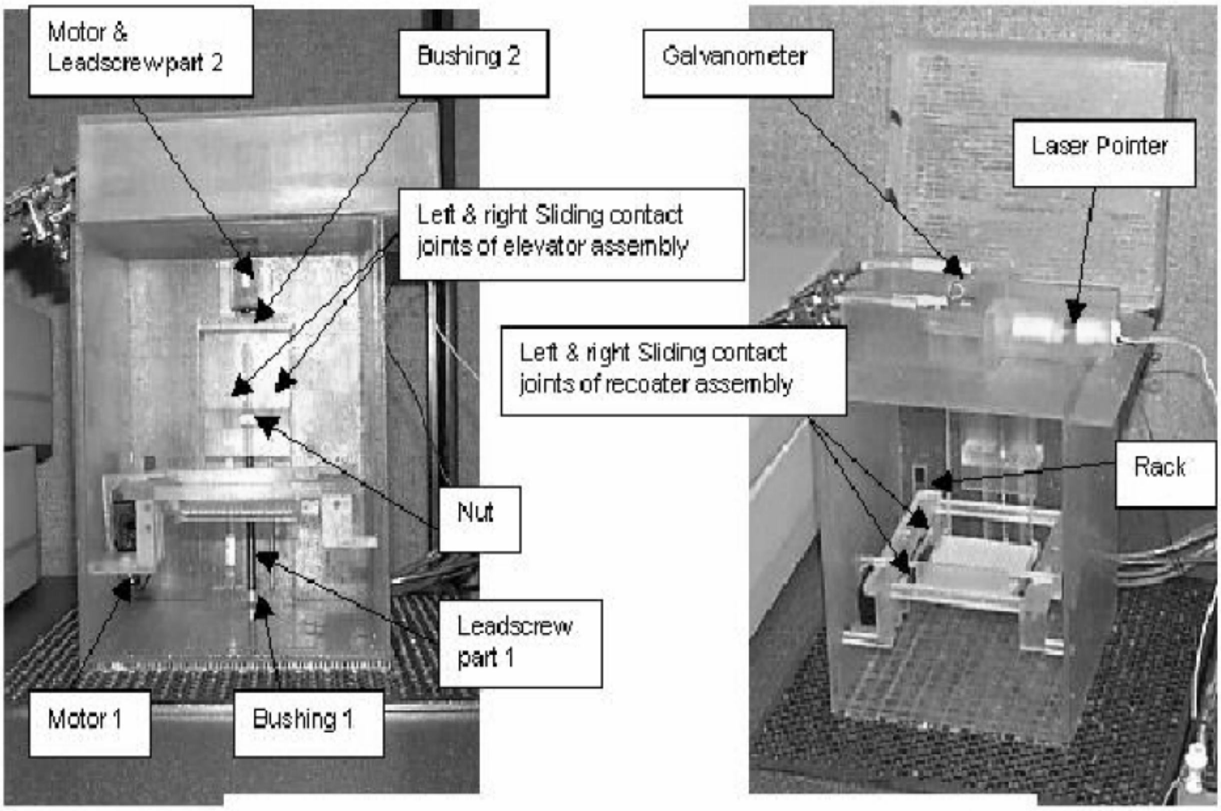

Motor & Leadscrewpart 2
Bushing 2
Left & right Sliding contact joints of elevator assembly
Nut
Leadscrew part 1
Motor 1
Bushing 1
Galvanometer
Laser Pointer
Left & right Sliding contact joints of recoater assembly
Rack

Figure 28. Functional prototype of an SLA-250

The model measures 152x152x257 mm. Enclosures were built in the RP machine for several of the components before insertion, see Kataria & Rosen (2000). Then the model was built in a single step in the SLA 250 machine, including 11 inserts, 4 sliding contact joints and one rotating contact joint. The model is fully functional.

For completeness, the following provides a list of all components. The inserts included:

1. Mechanical Components: Two bushings, lead-screw (2 parts), a nut, and a rack gear.
2. Electrical Components: Two gear-motors (one inserted with the lead-screw part 2)
3. Electronic and other Components: A set of galvanometers and a laser pointer.

The sliding contact joints included the bottom and top parts of the joint for the recoater guide (planar surfaces) and the left and right elevator guides (cylindrical shafts andholes). The rotating contact joint was for the hinge at the top of the chamber thatencases the galvanometer and laser.

This prototype proves the feasibility of building around inserts and illustrates the ultimate potential of the concept. Limitations observed include the need to redesign parts of the structure to accommodate some of the inserts and the need to redesign some parts to facilitate the removal of support structures. In fact, if we would build this model again, we would build it in modules, rather than as a single build. For example, thelead-screw SLA platform mechanism built well as a stand-alone experiment, but was difficult to clean up after this build. The recoating mechanism was similarly difficult toclean up.

6.3 Discussion - Challenges and Development of RP Technology for the Insertion of Components

An important aspect of common stereolithography machines is that they employ ultraviolet lasers to solidify the resin. Since most mechanical and electronic componentsare not sensitive to ultra-violet light and no significant heat is produced in the process,the inserted components are not damaged in the process. On the other hand, the liquidresin used in stereolithography may easily be contaminated by substances on theinserted components. For some inserts it is sufficient to clean them with alcohol before insertion, while others are coated manually with a small layer of resin and cured in the UV oven before insertion. Additionally, liquid resin may infiltrate inserts, such as electric motors, rendering them inoperable. Care must be taken to protect both the resin and components when inserting components in the resin vat.

The fact that inserted components may temporarily stick out of the part to be builtcan cause interference with the machine. For SL machines, interference can occurbetween the insert and the re-coating system that applies new layers of resin on top of the build part. Lastly, the laser beam of the SL machine can be blocked by an insert,resulting in a laser shadow which prevents material to be cured. Other RP processes share some of these and other limitations (e.g. excessive heat).

These limitations have spurred the development of a generalized Stereolithographymachine to overcome these problems, see Geving et al. (2000); Geving & Ebert-Uphoff(2000). 3D Systems, the primary manufacturer of SL machines, and other membercompanies of the Rapid Prototyping and Manufacturing Institute (RPMI) actively supportthese efforts that will make it easier for the user to include inserts in SL machines.

7. Conclusions

The rapid prototyping framework presented in this chapter provides fast, simple and inexpensivemethods for the design and fabrication of prototypes of robotic mechanisms.As evidenced by the examples presented above, the prototypes can be of great help togain more insight into the functionality of the mechanisms, as well as to convey theconcepts to others, especially to non-technical people. Furthermore, physical prototypescan be used to validate geometric and kinematic properties such as mechanicalinterferences, transmission characteristics, singularities and workspace.

Actuated prototypes have also been successfully built and controlled. Actuated mechanisms can be used in lightweight applications or for demonstration purposes. The main limitation in such cases is the compliance and limited strength of the plasticparts, which limits the forces and torques that can be produced.

Finally, several comprehensive examples have been given to illustrate how the rapidprototyping framework presented here can be used throughout the design process. Two robotic hands and a SLA machine model demonstrate a wide variety of link and jointfabrication methods, as well as the possibility of embedding sensors and actuators directlyinto mechanisms. In these examples, rapid prototyping has been used to demonstrate,validate, experimentally test (including destructive tests), modify, redesign and,in one case, support the machining of a metal prototype.

The use of Rapid Prototyping for the prototyping of robotic mechanisms is in its early stages. Much research remains to be done in order to (1) explore the full potential of RP for robotic mechanisms and identify the most promising research directions; (2)develop RP machines with additional functionality that are targeted to this new use.Given the rapid and inexpensive nature of the processes described here, it is believed that the framework presented in this paper can be a significant advantage in the designof robotic mechanisms.

Acknowledgements

Clement Gosselin and Thierry Lalibert´e would like to thank the Natural Sciences and Engineering Research Council of Canada (NSERC) for supporting this work through research and equipment grants. The help of Gabriel Cˆot´e in the preparation of the CADmodels is also acknowledged.
Imme Ebert-Uphoff and David Rosen gratefully acknowledge the financial support from the Rapid Prototyping and Manufacturing Institute (RPMI) member companies. In addition, they would like to acknowledge the contributions of colleague, Dr. TomKurfess, and of the following graduate students, Alok Kataria, Jacob Diez, Brad Geving,Chad Moore, and former undergraduate students, Kris Kozak and Jeff Miller. Thehelp of former RPMI lab manager, Giorgos Hatzilias, in conducting the experiments isacknowledged.

8. References

Alam, M.; Mavroidis, C.; Langrana, N. & Bidaud, P. (1999). Mechanism design using rapid prototyping, In Proceedings of the 10th World Congress on the Theory ofMachines and Mechanisms, Vol. 3, pp. 930–938, Oulu, Finland, June 1999.

Ashley, S. (1995). Rapid prototyping is coming of age. Mechanical Engineering, Vol.117, No. 7, July 1995, pp. 62–68.

Beck, J.E.; Prinz, F.B.; Siewiorek, D.P. & Weiss, L.E. (1992). Manufacturing mechatronicsusing thermal spray shape deposition, Solid Freeform Fabrication Symposium, Austin, TX, August 1992.

Binnard, M. (1999). Design by Composition for Rapid Prototyping. Kluwer Academic Publishing.

Binnard, M. & Cutkosky, M.R. (1998). Building block design for layered shape manufacturing, In Proceedings of 1998 ASME Design Engineering Technical Conferences, number DETC98/DFM-12, Atlanta, GA, Sept 1998.

Cham, J.G.; Pruitt, B.L.; Cutkosky, M.R.; Binnard, M.; Weiss, L.E. & Neplotnik, G. (1999). Layered manufacturing with empedded components: Process planningconsiderations, In Proceedings of 1999 ASME Design Engineering Technical Conferences, number DETC99/DFM-8910, Las Vegas, NV, Sept 1999.

Dasgupta, B. & Mruthyunjaya, T.S. (2000). The Stewart platform manipulator: a review.Mechanism and Machine Theory, Vol. 35, No. 1, January 2000, pp. 15–40.

Diez, J.A. (2001). Design for additive fabrication – building miniature robotic mechanisms.Master's thesis, Georgia Institute of Technology, School of MechanicalEngineering, Atlanta, Georgia, March 2001.

Frecker, M.I.; Anathasuresh, G.K.; Nishiwaki, S.; Kikuchi, N. & Kota, S. (1997). Topological synthesis of compliant mechanisms using multi-criteria optimization.Transactions of the ASME, Journal of Mechanical Design, Vol. 119, June1997, pp. 238–245.

Geving, B. & Ebert-Uphoff, I. (2000). Enhancement of stereo-lithography technology to support building around inserts. In Proceedings of 2000 ASME Design Engineering Technical Conferences, number DETC00/MECH-14207, Baltimore, MD, Sept 2000.

Geving, B.; Kataria, A.; Moore, C.; Ebert-Uphoff, I.; Kurfess, T.R. & Rosen, D.W. (2000). Conceptual design of a generalized stereolithography machine, In Proceedings of 2000 Japan-USA Symposium on Flexible Automation, number 2000JUSFA-13172, Ann Arbor, MI, July 2000.

Goldfarb, M. & Speich, J.E. (1999). A well-behaved revolute flexure joint for compliant mechanism design. Transactions of the ASME, Journal of Mechanical Design, Vol. 121, September 1999, pp. 424–429.

Gosselin, C.M. & Hamel, J.-F. (1994). The agile eye: a high-performance three-degreeof-freedom camera-orienting device. In Proceedings of the IEEE InternationalConference on Robotics and Automation, pp. 781–786, San Diego, CA, 1994.

Gosselin, C.M.; Allan, J.-F. & Lalibert´e, T. (1999). A new architecture for a highperformance 6-dof parallel mechanism, In Proceedings of the Tenth World Congress of the Theory of Machines and Mechanisms, Vol. 3, pp. 1140–1145, Oulu, Finland, 1999.

Hirose, S. (1993). Biologically Inspired Robots. Oxford University Press, Oxford, 1993.

Howell, L.L & Midha, A. (1994). A method for the design of compliant mechanisms with small-length flexural pivots. Transactions of the ASME, Journal of Mechanical Design, Vol. 116, No. 1, March 1994, pp. 280–290.

Howell, L.L & Midha, A. (1996). A loop-closure theory for the analysis and synthesis of compliant mechanisms. Transactions of the ASME, Journal of Mechanical Design, Vol. 118, No. 1, March 1996, pp. 121–125.

Jacobs, P.F. (1996). Stereolithography and other RP&M Technologies. ASME Press, New York.

Kai, C.C. & Fai, L.K. (1997). Rapid Prototyping: Principles and Applications in Manufacturing. John Wiley & Sons, Inc.

Kataria, A. (2000). Standardization and process planning for building around inserts in stereolithography apparatus. Master's thesis, Georgia Institute of Technology, School of Mechanical Engineering, Atlanta, Georgia, July 2000.

Kataria, A. & Rosen, D.W. (2000). Building around inserts: Methods for fabricating complex devices in stereolithography. In Proceedings of 2000 ASME Design Engineering Technical Conferences, number DETC00/MECH-14206, Baltimore, MD, September 2000.

Lalibert´e, T.; Gosselin, C.M. & Cˆot´e, G. (1999). Rapid prototyping of mechanisms. In Proceedings of the 10th World Congress on the Theory of Machines and Mechanisms, Vol. 3, pp. 959–964, Oulu, Finland, June 1999.

Lalibert´e, T.; Gosselin, C.M. & Cˆot´e, G. (2000). Rapid prototyping of lower-pair, geared-pair and cam mechanisms. In Proceedings of 2000 ASME Design Engineering Technical Conferences, number DETC00/MECH-14202, Baltimore, MD, Sept 2000.

Lalibert´e, T. & Gosselin, C.M. (2001). Underactuation in space robotic hands, In Proceedings of Sixth ISAIRAS: A New Space Odyssey, Montr´eal, Canada, June 2001.

Lalibert´e, T. & Gosselin, C.M. (2003). Actuation system for highly underactuated gripping mechanism. United States Patent No. 6505870.

Lipson, H. & Pollack, J.B. (2000). Automatic design and manufacture of robotic lifeforms. Nature, Vol. 406, August 2000, pp. 974–978.

Miller, J. (2001). Undergraduate research report - spring semester 2001. Technical report, Georgia Tech, May 2001.

Oberg, E.; Jones, F.D. & Horton, J.H. (1988). Machinery's Handbook, 23 edition. Industrial Press, New York.

Paros, J.M. & Weisbord, L. (1965). How to design flexure hinges. Machine Design, Vol. 37, No. 27, November 1965, pp. 151–156.

Phillips, J. (1990). Freedom in Machinery. Cambridge University Press, Melbourne.

Rajagopolan, S. & Cutkosky, M.R. (1998). Tolerance representation for mechanism assemblies in layered manufacturing. In Proceedings of 1998 ASME Design Engineering Technical Conferences, number DETC98/DFM-5726, Atlanta, GA, Sept 1998.

Rajagopolan, S. & Cutkosky, M.R. (1999). Optimal pose selection for in-situ fabrication of planar mechanisms. In Proceedings of 1999 ASME Design Engineering Technical Conferences, number DETC99/DFM-8958, Las Vegas, NV, Sept 1999.

Speich, J.E. & Goldfarb, M. (2000). A compliant-mechanism-based three degree-offreedom manipulator for small-scale manipulation. Robotica, Vol. 18, 2000, pp. 95–104.

Wohler, T. (1992). CAD meets rapid prototyping. Computer-Aided Engineering, Vol. 11, No. 4, April 1992.

Wohlers, T. (1993). Rapid prototyping systems. In Proceedings of the First Rapid Prototyping Convention, Paris, France, June 1993.

Won, J.; DeLaurentis, K.J. & Mavroidis, C. (2000). Fabrication of a robotic hand using rapid prototyping. In Proceedings of 2000 ASME Design Engineering Technical Conferences, number DETC00/MECH-14203, Baltimore, MD, Sept 2000.

The Role of 3D Simulation in the Advanced Robotic Design, Test and Control

Laszlo Vajta & Tamas Juhasz

1. Introduction

The intelligent mobile robots are widely used in applications such as general indoor and outdoor operations, emergency rescue operations, underwater and space exploration, pipe- and duct inspection in nuclear power plants, construction environments and so on.

Mobile robots use locomotion that generates traction, negotiates terrain and carries payload. Some robotic locomotion also stabilizes the robot's frame, smoothes the motion of sensors and accommodates the deployment and manipulation of work tools. The locomotion system is the physical interface between the robot and its environment. It is the means by it reacts to gravitational, inertial and work loads.

Generally the mobile robots have some amount of autonomy. This means that they can do programmed tasks while they are responding to the occurring environmental effects automatically. A large proportion of the mobile robots are remotely operated platforms that usually have local autonomy as well. The interactive human control of these telerobots needs advanced 3D visualization using novel graphics techniques.

Typically the remote controlling of a robot requires a virtual model of the given platform that describes its behavior correctly. We can state that telerobotics and robot simulation are usually interconnected research areas. Concerning the current field of telerobotics and robot-simulation in general, one key factor in the human-robot interface is realistic visualization. Although the traditional computer graphics algorithms produce fairly good quality virtual environments, there is no doubt about the need for more realistic ("near true") 3D impressions. Of course this can lead to interactive, entertaining applications (Vajta & Juhasz, 2004) as well. In this paper we will present an overview about the advanced graphics techniques in semi- or true 3D, which could be applied in a modern mobile robot simulation solution.

Although our final aim is real robotics, it is often very useful to perform simulations prior to investigations with real robots. Designing mobile robots (either teleoperated or fully-autonomous platforms) requires a step-by-step approach including the accompanying testing processes as well. The sum of the development, testing and running charges or any other expenses demands utilizing a simulator to cut down overall costs. Simulations are easier to setup, less expensive, faster and more convenient to use. Building up new robot models and setting up experiments usually takes only a few hours. Simulated robotics setups are less expensive than real robots and real world setups, thus allowing a better design exploration. Virtual prototyping through behavioral simulation enables continuous iterative design, allowing the developer of a robotics system to find inconsistencies earlier in the design phase. If we have validated simulation results, we can transfer our procedures onto the real robots. In the second part of this paper we will present our robot simulator project from the visualization side and the design, test and control side as well.

Finally it should be noted, that people in the field of interactive computer graphics use the terms "3-D" in the meaning of a "realistic"-looking image which may be perceived with one eye. This is a confusing nomenclature because most people use the term "3-D," when they mean a *stereoscopic* image. Such an image requires the beholder to use both eyes. We will use the term "3-D" with both meanings, too.

2. Mobile Robot Simulation Applications

In general the ideas and techniques developed during the simulation process yield to ideal conditions to raise synergy: a catalytic effect for discovering new and simpler solutions to traditionally complex problems. Using virtual prototyping by means of behavioral simulation reduces the time-to-market, as it shows the inconsistencies early in design phase. Abstract modeling with CAD tools, enhanced conceptual design and moving up life-cycle assessments by virtual prototypes allow devising the optimal architecture of the system.

Today there is a wide variety of simulators on the market, but most of them are trade, model- or application specific ones. A summary was created by Yiannis Gatsoulis (University of Leeds) who was claimed by the CLAWAR community to analyze the state-of-the-art of this field a few years ago. It contains a snapshot of the available softwares (environment editors, image processing- and control libraries, system simulators, etc.) that are in connection with this research area. It assesses the cost, usability, expandability and rapid development abilities of the given applications. Unfortunately since the creation of this document a lot of projects were suspended or cancelled – as their authors became to deal with other tasks. There are many "bid fair" applications with tall feature lists, almost without any usable, implemented functionality. Considering the amount of available system simulators today – that are really worth trying to use – let us emphasize only the Webots® software system that is still developing continually (as a commercial product of the Cyberbotics Ltd.). Reviewing this software can describe well the state-of-the-art in mobile robot simulation in these days.

2.1 Cyberbotics Ltd.

The Cyberbotics Ltd. is one of the leading companies in the mobile robot prototyping and simulation software area. The company was founded in 1998 by Olivier Michel, as a spin off company from the MicroComputing and Interface Lab (LAMI) of the Swiss Federal Institute of Technology, Lausanne (EPFL). Cyberbotics develops custom simulation tools like the Sony Aibo robot simulator which was developed for Sony Corporation, but it also develops and markets Webots, the award winning fast prototyping and simulation software for mobile robotics. Developed since 1996, Webots became a reference software used by over 200 universities and research centers worldwide.

2.2 The Webots® Simulator – Overview

The Webots simulator package is now the main commercial product available from Cyberbotics Ltd (Cyberbotics). The provided robot libraries enable you to transfer your control programs to several commercially available real mobile robots. Webots lets you define and modify a complete mobile robotics setup, even several different robots sharing the same environment. For each object, you can define a number of properties: such as shape, color, texture, mass, friction, etc. Each robot can be equipped with a large number of available sensors and actuators. You can program these robots in many languages using your favorite development environment, simulate them and optionally transfer the resulting programs onto your real robots. Webots has been developed in collaboration with the Swiss Federal Institute of Technology in Lausanne, thoroughly tested, well documented and continuously maintained for over 7 years.

2.3 The Webots® Simulator – Features

Webots has a number of essential features intended to make this simulation tool both easy to use and powerful:

- Models and simulates many kinds of mobile robots, including wheeled, legged and flying ones.
- Includes a complete library of sensors and actuators.
- Lets you program the robots in C, C++ and Java, or from third party software through TCP/IP.
- Transfers controllers to real mobile robots (including Aibo®, Lego® Mindstorms®, Khepera®, Koala® and Hemisson®).
- Uses the ODE (Open Dynamics Engine) library for accurate physics simulation.
- Creates AVI or MPEG simulation movies for web and public presentations.
- Includes many examples with controller source code and models of commercially available robots.
- Lets you simulate multi-agent systems, with global and local communication facilities.

2.4 The Webots® Simulator – Robot and World Editor

A library of sensors is provided so that you can plug a sensor in your robot model and tune it individually (range, noise, response, field of view, etc.). This sensor library includes distance sensors (infra-red and ultrasonic), range finders, light sensors, touch sensors, global positioning sensor (GPS), inclinometers, compass, cameras (1D, 2D, color, black and white), receivers (radio and infra-red), position sensors for servos, incremental encoders for wheels. Similarly, an actuator library is provided. It includes differential wheel motor unit, independent wheel motors, servos (for legs, arms, etc.), LEDs, emitters (radio and infra-red) and grippers.

With Webots, you can create complex environments for your mobile robot simulations, using advanced hardware accelerated OpenGL technologies, including lighting, smooth shading, texture mapping, fog, etc. Moreover, Webots allows you to import 3D models in its scene tree from most 3D modeling software through the VRML97 standard. You can create worlds as large as you need and Webots will optimize them to enable fast simulations.

Complex robots can be built by assembling chains of servo nodes. This allows you to easily create legged robots with several joints per leg, robot arms, pan / tilt camera systems, and so on. For example, you can place several cameras on the same robot to perform binocular stereo vision, or 360 degree vision systems.

2.5 The Webots® Simulator – Realistic Simulation

The simulation system used in Webots uses virtual time, making it possible to run simulations much faster than it would take on a real robot. Depending on the complexity of the setup and the power of your computer, simulations can run up to 300 times faster than the real robot when using the fast simulation mode. The basic simulation time step can be adjusted to suit your needs (precision versus speed). A step-by-step mode is available to study in detail how your robots behave.

Simulating complex robotic devices including articulated mechanical parts requires precise physics simulation. Webots relies on ODE (Open Dynamics Engine) to perform accurate physics simulation wherever it is necessary. For each component of a robot, you can specify a mass distribution matrix (or use primitives for simple geometries), static and kinematical friction coefficients, bounciness, etc. Moreover each component is associated with a bounding object used for collision detection.

Servo devices can be controlled by your program in torque, position or velocity. The control parameters for the servo can be individually adjusted from your controller program.

The graphical user interface of Webots allows you to easily interact with the simulation while it is running. By dragging the mouse, you can change the viewpoint position, orientation and zoom using the mouse wheel. Pressing the shift key while dragging the mouse allows you to move or rotate objects. This feature facilitates interactive testing.

2.6 The Webots® Simulator – Programming Interface

Programming your robot using C or Java language is quite simple. Any Webots controller can be connected to a third party software program, such as Matlab®, LabView®, Lisp®, etc. through a TCP/IP interface. Research experiments often need to interact automatically with the simulation. The supervisor capability allows you to write a program responsible for supervising the experiment. Such a program can dynamically move objects, send messages to robots, record robot trajectories, add new objects or robots, etc. The supervisor capability can be used in computationally expensive simulations where a large number of robot configurations and control parameters have to be evaluated, as in genetic evolution, neural networking, machine learning, etc.

2.7 The Webots® Simulator – Transfer to Real Robots

Once tested in simulation your robot controllers can be transferred to real robots:

- Khepera® and Koala®: cross-compilation of C Webots controllers and remote control with any programming language
- Hemisson®: Finite state automaton graphical programming with remote control and autonomous execution modes.
- LEGO® Mindstorms®: cross-compilation for RCX of Java Webots controllers based on LeJOS.
- Aibo®: cross-compilation of C/C++ Webots controller programs based on the Open-R SDK
- Your own robot: The Webots user guide explains how to build your own Webots cross-compilation system for your very own robot.

2.8 The Webots® Simulator – Summary

Webots has been used by more than one hundred universities and research centers worldwide since 1998 for both education and research purposes. Education applications allow the students to get started with robotics, 3D modeling, programming, artificial intelligence, computer vision, artificial life, etc. with an integrated tool. It is easy to implement a virtual robot contest, like a soccer contest or a humanoid locomotion contest, based on Webots, which is highly motivating for the students. Applications using Webots can be classified as follows:

- The multi-agent simulations category includes research experiments where several robots cooperate to reach a global goal.
- The artificial intelligence category attempts to validate psychology hypothesis, mainly learning, by simulating intelligent mobile robot behaviors.
- The control research category involves developing efficient control algorithms to perform complex mobile robot motion.
- The robot design category aims at shaping mobile robots defining the position and properties of its sensors and actuators. This is especially useful to investigate new wheeled, legged or flying robots.

3. Methods for 3D Visualization

The human sensing of the depth is a complex sensation. It has at least three different components, the so called extraretinal (not discussed here), the monocular and the binocular ones.

The monocular impression is the mixture of more, well-known phenomena as the perspective, shadowing, atmospheric distortion, a priori expected size of the objects, texture distortion, etc. Even in case of one-eye (monocular) sensing, the human brain interprets some kind of depth information. Although the capability of the monocular sense of depth is limited and it is sometimes inaccurate, it has essential role in the global sensing procedure.

Binocular sensation (frequently called as stereo vision) is the most trivial component of the depth sensing. Due to the 4-6 cm separation between the eyes, each eye has a slightly different viewpoint. The images from the two different viewpoints are sent to the brain and their difference – which is termed parallax – is interpreted as depth.

3.1 Advanced Visualization Techniques

As we mentioned before, the artificially generated visual impression is an inherent part of the man-machine interface. The use of visualization methods in this field, especially in case of remote controlling of mobile systems (teleoperation) needs the possible maximum level of reality. There are plenty of displaying methods under development and in use for this purpose. In the following we give a short overview on this topic.

Stereoscopic systems need two different images from the same scene. The human eyes must sense the two images totally separated. We can classify the stereo displays based on the method of the separation:

- passive separation
- active separation

Stereoscopic systems, which use *passive spectacles*, separate the two – in space overlapped – images by some kind of filters. Anaglyphs use color code separation and the used spectacles are having red and cyan optical filters in fact. Other solutions are using polarized lights, where the two images have perpendicular or circular polarization, which is aligned with the direction to the polarization filters of the glasses. A new method, called Infitec™ (acronym for interfering filter technique, that is a trademark from Daimler-Chrysler Research and Technology – Ulm, and is licensed to Barco) has superior stereo separation, which is perfectly suited for applications with high-contrast, non-saturated images. Two optical filters split the color spectrum in 2 parts: one for the left and one for the right eye information. It delivers superior stereo separation without ghosting, with full freedom of motion, independent of head tilt. Based on (Barco) – compared to polarization-based passive stereo – it achieves better separation, is independent of screen technology and therefore results in better uniformity.

Stereoscopic systems with *active glasses* usually use time-overlapped images. In fact the active spectacles are programmed shutters, which direct the images intermittently to the right and left eyes. Active stereo method is a flicker-sensitive solution – the smallest time-shift between the image display and the glass control "kills" the stereo feeling.

There is no doubt about the need of glasses is the more difficult aspect of the implementation of stereoscopic systems. Other methods, which eliminate the need of glasses, have growing importance.

LCD displays with lenticular lens in front are projecting the adjacent columns of the screen in two slightly different directions – hopefully into the left and right eye of the viewer respectively.

The screen displays two, vertically interlaced images in the same time, which construct a stereo pair. The solution is sensitive on the movement of the viewer. There are displays on the market, which are able to track the movement of the user's eye, and modify the displayed images accordingly. This solution produces true depth sense, and allows relative free movement of the user – without the need for glasses. Auto-stereoscopic 3D imaging becomes popular for example on laptop computers.

We may not forget that monocular feeling plays an important role in our visual sensation. Our experiments proved that by using monocular sensing only – for example the transmitted images of an onboard camera – we can even drive a mobile platform remotely with high safety. How is this possible?

The stereo depth sense is produced through evaluation of the horizontal shift of corresponding points on two images. Even in case of random dot structure the human brain can find the corresponding pairs and represent them as in depth distributed information.

But there are other image features, which are heavily influenced by the depth, too. In our understanding, the key issue for the monocular depth sensing phenomenon is the depth sense from the motion.

The projected image of a true 3D scene, which contains objects in different distances, will change the arrangement of the objects on the image in case of any change in the viewer's position. Objects on the image are covering each other, whereas this coverage varies according to the viewing angle. Continuous movement of the viewer causes continuous change in this coverage. By displaying the image of the moving camera on a monocular screen impressive depths feeling can be achieved. The solution is popular in some telerobotic application, where the movement is an inherent part of the task in any case.

On the software side there are many stereo solutions for CAD systems. Quad-buffered stereo is a routine within OpenGL that can be employed by 3D applications in order to show stereo images. For CAD applications such as SolidWorks and Solid Edge, there are plugins to generate stereo pairs. There are solutions which allow images and movies formatted in common stereo formats – such as above and below, side by side and interlaced – to be displayed in any available viewing format including anaglyph, page-flipping, sync-doubling stereo display modes, or auto-stereoscopic display. Supported file formats include .jpg, .jps, .bmp, .tga, .gif, .wma, .wmv, .asf, .avi and .mpg. In addition some software allows a stereo 3D effect to be applied to non-stereo movies and images allowing them to be viewed stereoscopically.

Unfortunately the usage of stereo visualization of robot simulators is not a widespread solution. We will introduce two different approaches on this field: the use of anaglyphs and the 3D visualization by motion.

4. The RobotMAX Mobile Robot Simulator

Now there is an ongoing mobile robot simulator project (called: RobotMAX) at the Mobile- and Microrobotics Laboratory of the Department of Control Engineering and Information Technology in Budapest University of Technology and Economics.

The predecessors of this application are discussed in (Juhasz, 2003a; Juhasz, 2003b; Juhasz, 2004). RobotMAX is written totally from scratch (with code translating / reusing from the previous versions) using the new C# language (O'Reilly, 2003) that was introduced in the Microsoft .NET Framework. The framework is meant to be platform independent (Windows and Linux versions are available) and has a native support to advanced software technologies such as XML, SOAP or COM+.

4.1 Objectives in the Development of RobotMAX

In these days the key objectives for developing large software systems are modularity (concerning all devices and aspects) and interoperability (using the output from- or serve input to other related applications). We are trying to give general solutions for the upcoming robot simulation problems staying apart from model- or application-specific constraints.

One of our most important objectives is to develop a simulator that supports the testing of individual robotics components (vision system, navigation system, etc.) by means of hardware devices as well (so to support hardware-in-the-loop testing). For this purpose the main components of the simulated system are coupled through a standard interface and the simulation core and they are designed in a way that they could be replaced by an analogous physical hardware, transparently. Previously verified hardware devices can be useful if you want to focus on testing only one separate software component's behavior.

Another important objective is incorporating the X3D (Web3D standard) format to describe the virtual environment. This is a powerful and extensible open file format for 3D visual effects, behavioral modeling and interaction. It can be considered as a successor of the well-known VRML format. By providing an XML-encoded scene graph and a language-neutral Scene Authorizing Interface, it makes scene verification much easier and allows 3D content to be easily integrated into a broad range of applications. Its base XML language lets incorporating 3D into distributed environments, and facilitates moving 3D data between X3D-aware softwares (a.k.a. cross-platform or inter-application data transfer).

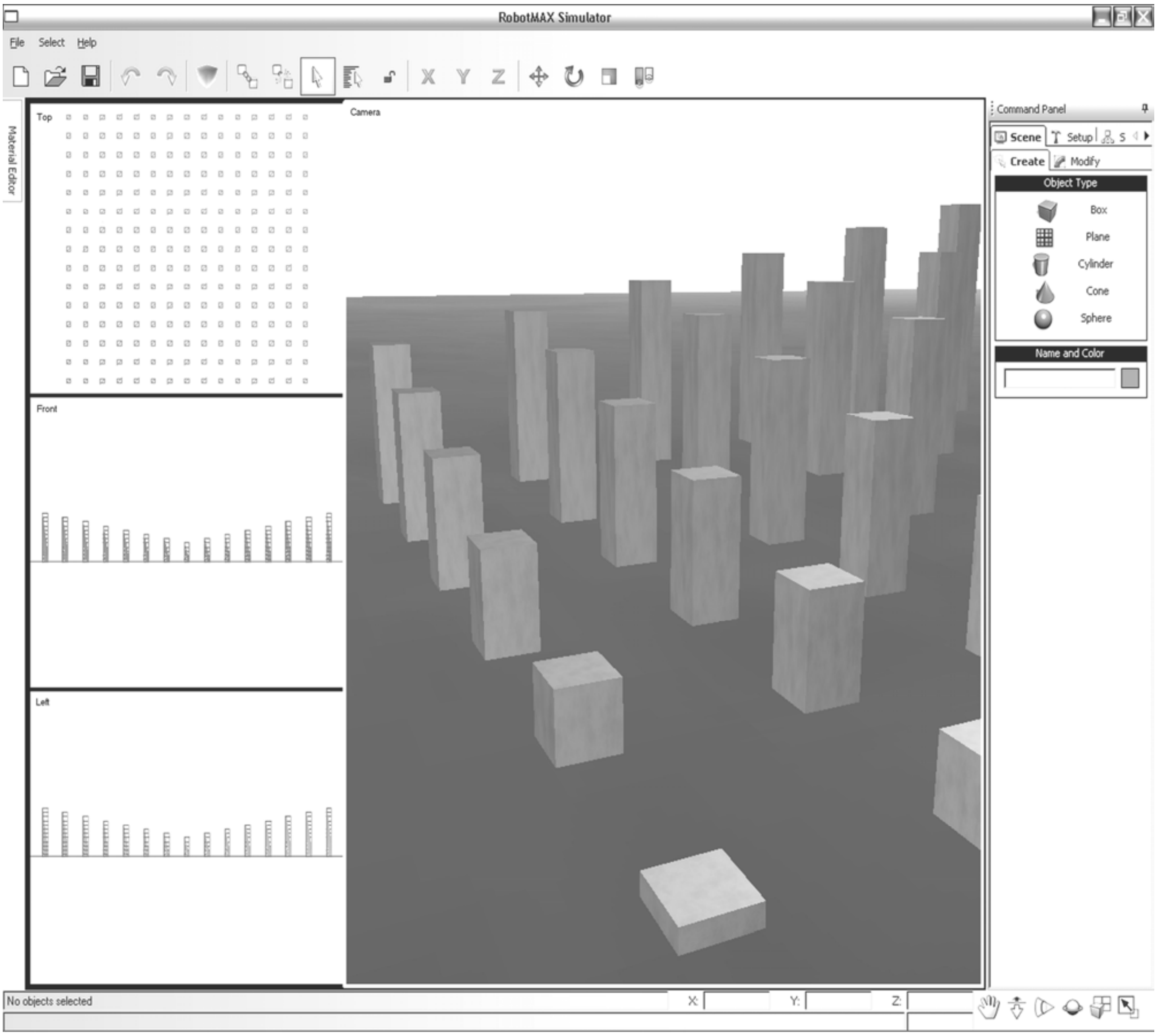
4.2 Visualization in RobotMAX

Our 3D visualization engine is based on the open-source Axiom 3D engine, which is used as our rendering "middleware". Axiom is a fully object oriented 3D graphics engine developed using C# and the Microsoft.NET v1.1 platform to create an easy to use, flexible, extendable, and powerful engine that allows for rapid development of games and other graphical applications (Axiom3D). By using the .NET framework (Thai & Lam, 2001) as the target platform, developers can focus more on core functionality and logic, rather than dealing with the complexities of languages like C++. The core of Axiom is a port of the very popular OGRE graphics engine, which was chosen based on its clean object-oriented design, powerful features, and flexibility. The main features of our extended 3D engine that is used in the robotics simulation are:

- Extensible Hierarchical Scene Graph with support for serializing standard X3D (successor of VRML standard) format scenes.
- Up to 4 reconfigurable rendering views (orthogonal or perspective cameras) with wireframe, solid or anaglyph rendering modes. We are using Managed DirectX 9 target render system (the OpenGL implementation using Tao.OpenGL wrapper is not working correctly, yet).
- Support for Ogre .material files, allowing the flexibility for controlling fixed function or programmed render state on object basis. For realistic dynamic behavior the objects' materials are containing their individual physical parameters such as density, friction, bounciness and so on. The dynamic physics simulation will be carried out via using a C# wrapper package for the also open source Open Dynamics Engine (ODE) using these parameters.
- Vertex and fragment programs can be the part of the material files. There is a full support for low level shaders written in assembler, as well as all current high level shader language implementations (Cg/DirectX HLSL/GLSL) for stunning visualization effects. Supported profiles are: arbvp1, vp_1_1…vp30 for vertex shaders; and arbfp1, ps_1_1…ps_2_0, fp20, fp30 for pixel shaders.
- Easy to use Render to Texture functionality (used in anaglyph render mode: described later)
- Keyframe animation support. Currently allows animations to be assigned to nodes in the scene graph, allowing objects to move along predefined spline paths. Can be used to define moving obstacles in the virtual environment.
- Extendable controller support, allowing a wide variety of automated effects at runtime. Can be used to do time-specific motions and parameter changes during the simulation.
- Skeletal animation with an Ogre .skeleton file loader. Features include multiple bone assignments per vertex, smooth frame rate scaled blending, and multiple animations can be blended together to allow for seamless animation transitions. Skeletons are the best technique for describing the complex movement of the vertices of a walking humanoid platform.

4.3 User Interface in RobotMAX

Our mobile robot simulator solution uses a CAD interface that is similar to the well-known 3DStudio MAX® software (proprietary of Discreet™) to let you easily design the desired virtual environment and mobile platforms. Many primitive object types are available (boxes, planes, cylinders, cones and spheres) and general 3D mesh objects can be imported from standard X3D or Ogre .mesh files. All 3D objects (even cameras and lights) are nodes in a tree structure that are handled and stored hierarchically in the X3D scene description file. Thus one object can be a parent of another, where each object passes its transformation to its children. By construction all objects are the child of an unyielding root object (the one that has no parent at all), and they can be re-linked to their new parent (with the Link tool) as desired. The nodes that are representing the individual robot platforms themselves do not differ much from other general nodes. You can mark "anything" (3D objects, joints, cameras, etc.) as a part of an individual mobile platform during the assembly phase (in the Robot Construction Dialog). Generally there is a base chassis that holds (i.e. it is the parent of) the wheels, onboard cameras, sonars, etc. in each platform. These parts can be referenced and manipulated independently within the user-defined high-level simulation program.



No objects selected

Figure 1. The user interface of the RobotMAX simulator

After finishing up with the environment construction, you can switch to simulation mode where you can load, edit and execute your own high-level program (i.e. a precompiled .NET library) that will control the behavior of your robots. Of course the data of the onboard sensors (CCD cameras, sonars, tilt sensors, etc.) are at your disposal during the simulation.

4.4 The Architecture of the RobotMAX Simulator

We can say that mobile robot systems (both real and simulated ones) have three separate main components – sensing, processing and locomotion – that are communicating with each other. If we are in the software world, it is an understandable imagination that these components are treated as separate modules in the application. These modules are connected through a standard communication interface. The modular architecture of the simulator and this interface make possible the simulation of the behavior of a remotely controlled platform where the sensing and locomotion components are spatially separated from the intelligent control component (running on different computers that are connected via the internet or similar communication channels). Thus the communication model of a remote controlled mobile robot looks like the following:

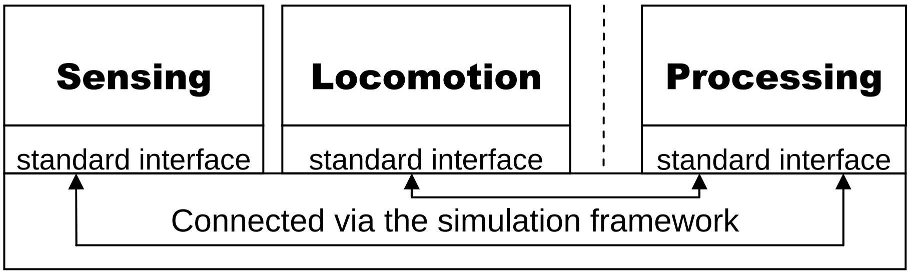


Figure 2. The modular architecture of a remote controlled robot (processing is on the local side)

The simulator will offer the following kinds of locomotion models:

- Differential drive with caster
- Ackermann steering
- Synchronous drive
- Tricycle
- Omni-directional (a.k.a. Swedish wheel)

The user can manually select and configure the desired driving subsystem in the previously mentioned Robot Construction Dialog. The simulator incorporates the dynamic model of these platforms and it uses the open source ODE (Open Dynamics Engine) physics engine to approximate the behavior of the physical platform that is simulated.

If we use the previously discussed standard communication interfaces between the implemented modules then the core simulator framework can treat virtual and physical components the same way. By using hardware implementation in a given module, we can talk about hardware-in-the-loop testing. Using analogous hardware devices in a test loop lets you focus only on a specific part of the system that will be under investigation. This methodology plays an important role in virtual prototyping scenarios. Let's assume the following: we have a virtual representation of the real scene (with acceptable accuracy) where our real robot will be operating. We place this physical robot into an empty room with phantom obstacles (e.g.: sketch drawings on the floor: to avoid any unpredicted collisions) and control it according to the virtual environment's visual information. Thus we can verify our dynamic model being used whether the physical platform reacts the same way as the virtual one upon the same command sequence (for example a common navigation algorithm). If we got satisfactory results we can take our platform out of the room and put it to the real environment.

4.5. Advanced 3D Visualization in RobotMAX

The simulator supports advanced 3D visualization using the anaglyph technique. You can see good quality colored anaglyphs at (Anachrome 3D). In the followings we will discuss the used algorithm in our application, and compare it against the motion-based monocular 3D technique.

Generally when we are looking at an object in the distance we perceive distinct images with our left and right eyes. The closer the object is the bigger is the deviation (the so called parallax) between these images. Our brain "calculates" the depth from this difference: this is the basis of stereo imaging that can lead to true 3D perception.

We have implemented anaglyph rendering mode in the RobotMAX simulator. Anaglyph imaging produces a stereoscopic motion- or still picture in which the left component of a composite image is usually red in color. This is superposed on the right component in a contrasting color (usually cyan or blue) to produce a three-dimensional effect when viewed through correspondingly colored filters in the form of spectacles.

In our application this technique can be used to enhance 3D perception of the user in the virtual reality. Let's assume we have a monocular perspective camera (named "CameraM"): the parameters of which (position, orientation, field-of-view, etc.) are all known. We clone this camera twice: getting "CameraL" and "CameraR" as results that are matching their parent exactly. The previous one will present our left eye's view and the subsequent one is for the right eye's view.

According to the anatomy of an adult human being, the center-to-center distance between the eyes of the viewer is assumed to be 6 cm, thus we shift the left ("CameraL") and right ("CameraR") cameras by 3-3 cm along their local X axes (horizontal) in opposite ways.

We can "mark" the information of the given images using RGB filters: the left camera should have a red filter (RGB: [1,0,0]) whilst the right one should have a cyan filter (RGB: [0,1,1]). The anaglyph image contains the combined visual information that can be separated if we are using a red-cyan anaglyph spectacle. Fortunately red and cyan are complementary colors (their sum produces pure white or gray if they are mixed in the right proportions), so we don't lose brightness (luminance) information in anaglyph images. If we combine these images by simply adding them together, we get the desired anaglyph image.

All of these operations can be carried out in our rendering engine. The engine is using texture render targets for "CameraL" and "CameraR", and is multi-texturing the given images in each frame over the same invisible 2D plane using texture-blending operations, then this plane is made visible to the user. This can be done many times per seconds, so this rendering mode can be used in anaglyph motion pictures as well.

Although anaglyph images look black and white viewed through the spectacles, this feature isn't disadvantageous in case of RobotMAX, because we don't need color rendering in this mode of the simulator. Color information in fact is not really necessary for designing and testing. Nevertheless in case of telerobotic applications there is a limited need for color information as well (depth information is more important).

The previous technique is able to create and enhance the depth cue of stereoscopic vision at a price. It requires specialized viewing equipment, and therefore is not a realistic solution for three-dimensional viewing for the masses. It also is limited to a single viewing perspective. Therefore, it is unable to provide the important depth cue of motion parallax.

On the other hand the generation of anaglyph images is more effective than the calculation of continuous camera motion, which is necessary for both the motion-based monocular stereo effect and motion parallax. Even in case of well-configured motion stereo sequences without binocular sensation the resulting depth feeling is strongly limited. However we implemented motion stereo reproduction of still images in RobotMAX (Figure 3 shows an example) as a compromise to eliminate the need for additional special equipment (glasses).

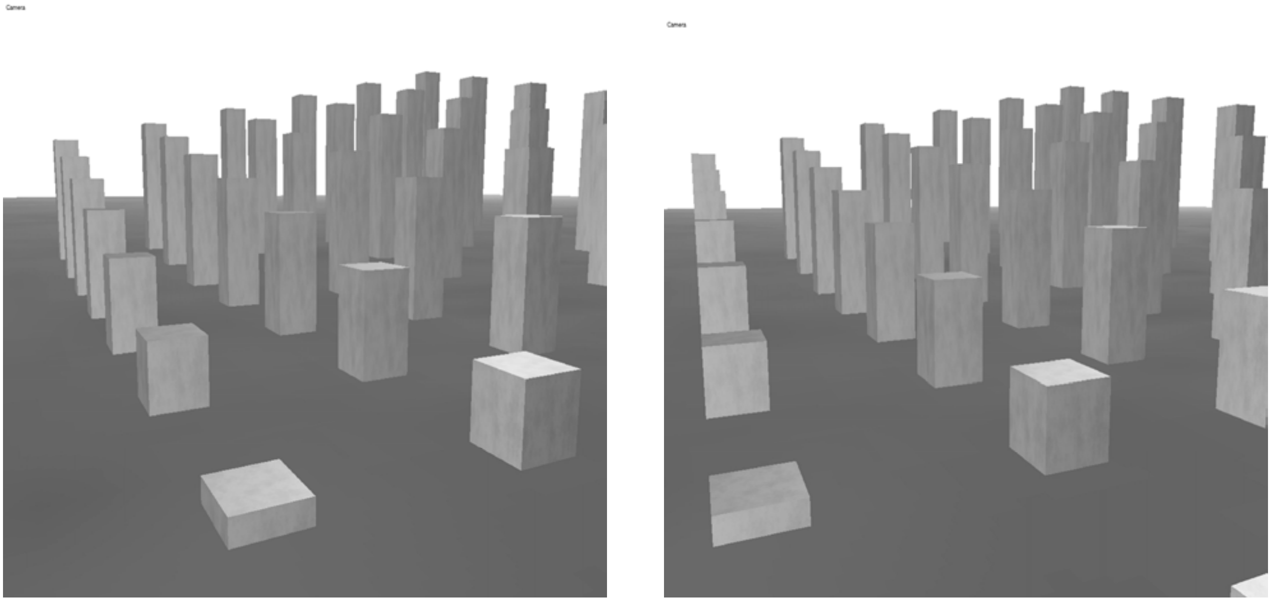


Figure 3. The first and last frames of a motion-stereo image sequence

On Figure 3 you can see two frames of a motion-stereo image sequence rendered using RobotMAX. The X coordinate (in world coordinate system) of the viewpoint of the monocular camera was slowly changing during the time. You can see that the corresponding perspective projected objects are overlapping differently on these frames. Those objects that are in the distance are almost static whilst the closer ones are seemingly moving a lot more. Our brain is using this changing occlusion information during interpreting the different 2D images from time-to-time, perceiving a depth sensation. This is an example how monocular, 2D perception can produce 3D information.

It is incontrovertible, that the use of a special auto-stereoscopic display is the most advanced solution (to set aside its price), because lenticular displays are able to solve both of the problems and provide three-dimensional stereo images over a range of viewing angles without the need for special viewing glasses (or a head mounted unit).

A lenticular display consists of a horizontal array of cylindrical lenses placed in front of interleaved pictures of a given object from different viewing angles. The device is designed so that at a given viewing angle, only one of the set of interleaved images can be seen. Therefore, as the viewer moves, the lenticuar sheet picks out a different image, and the percieved image rotates to provide the proper perspective for the viewer. This creates the experience of motion parallax. Figure 4 illustrates the method with which a lenticular display creates this effect (Okoshi, 1976). Since there is a difference between the eyes in both distance and viewing angle, each eye will see a different image, and stereo vision is achieved as well (see figure 5).

This technique is able to provide stereoscopic vision and motion parallax without the need for specialized viewing equipment. However, the achievable resolution of such a device is limited by large files sizes, and the available monitor resolution due to the need for many interleaved images. Another drawback is that despite its incorporation of many depth cues, it is still a two dimensional representation, and it is incapable of exploiting accomodation. Only truly spatial three-dimensional displays will be able to provide this.

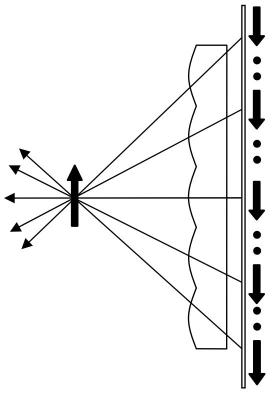


Figure 4. Lenticular display

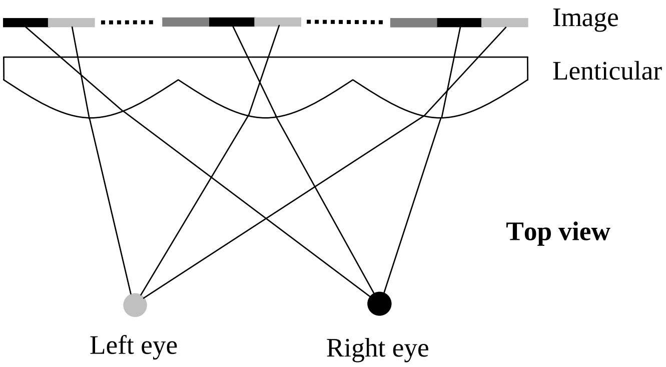


Figure 5. Stereo sense with motion parallax

In fact all of the three-dimensional imaging techniques described up to this point has been two-dimensional representations with enhanced depth cues to create the illusion of a third spatial dimension. However, no matter how complex they are, they are still two-dimensional and ultimately flawed. This is because they lack the ability to appease the eyes need for ocular accommodation in experiencing a three-dimensional world. Spatial three-dimensional displays will allow the eye to focus at different depths throughout the display and truly experience all three dimensions. The volumetric display technique is still in its infancy, but it is worth further investigation because it is rich not only in fundamental science from different fields but also in skilful, detailed engineering. Already their application in the robotics lies far away.

The auto-stereoscopic plug-in module for RobotMAX is currently under development.

5. Conclusions and Future Plans

In this paper we presented an overview about modern visualization approaches in the field of telerobotics and interactive robotics simulation. We have made a comparison presenting the advantages and disadvantages between the (binocular) anaglyph and the (monocular) motion stereo methods in our robot simulator application.

At this time, our team in the Mobile- and Microrobotics Laboratory is in the very center of experimenting with some of the great amount of available devices supporting advanced monocular and binocular 3D techniques to find one that is the nearest to our requirements. From this aspect we presented the RobotMAX simulator, which is currently under development at our laboratory: it will integrate robotics design (advanced visualization, driving mechanisms- and kinematical structure planning, sensor layout optimalization, etc.) and interactive control with the effective hardware-in-the-loop testing methodology into a modern simulation framework. Our mobile robot simulation project has another objective also: RobotMAX is aimed be a new research and education platform for the next generation of students in the field of mobile robotics and 3D imaging in our department at BUTE.

Acknowledgements

We would like to acknowledge the support of the Hungarian Scientific Research Fund (OTKA) grant No.: T-042634-OTKA, and the CLAWAR (Climbing And Walking Robots) Research Training Network – Mobile Robotic Demonstrators and Applications: G1RT-CT-2002-05080.

6. References

Anachrome 3D: http://www.anachrome.com/index.htm

Axiom3D: http://www.axiom3d.org

Barco: http://www.barco.com/VirtualReality/en/stereoscopic/infitec.asp

Cyberbotics: http://www.cyberbotics.com

Discreet-3D Studio MAX: http://www4.discreet.com/3dsmax/

Juhasz, T. (2003a): "OpenGL powered mobile robot simulator supporting research on landmark-based positioning", *Proceedings of MicroCAD'03 Conference*, 2003, Vol. N., pp. 85-91, ISBN 9-636-61560-8

Juhasz, T. (2003b): "Graphics acceleration techniques for a mobile robot simulator",*Proceedings of CESCG'03: Central European Seminar on Computer Graphics*, Section 5: Computer Vision, 22-24th April, 2003, Budmerice, Slovakia

Juhasz, T. (2004): "Surveying telerobotics and identification of dynamic model parameters", *Proceedings of MicroCAD'04 Conference*, 2004, Vol. K., pp. 211-216 (in Hungarian), ISBN 9-636-61619-1

Okoshi, T. (1976): "Three-Dimensional Imaging Techniques", *Academic Press*, New York

O'Reilly & Associates: "Programming C#", May, 2003 ISBN 0-596-00489-3

Thai T., Lam H. Q. (2001): ".NET framework essentials", 1st edition, ISBN 0-596-00165-7

Vajta, L.; Juhasz, T. (2004): "New challenges in mobile robot simulation systems", *Proceedings of 1st CLAWAR/EURON Workshop on Robots in Entertainment, Leisure and Hobby*, 2-4. December 2004, Vienna, Austria

Web3D consortium: "X3D overview", http://www.web3d.org/x3d/overview.html

Mechatronics Design of a Mecanum Wheeled Mobile Robot

Peter Xu

1. Introduction

The Mecanum wheel was designed in Sweden in 1975. Using four of these wheels provides omni-directional movement for a vehicle without needing a conventional steering system (Muir & Neumann, 1990; Dickerson & Lapin, 1991; Braunl, 1999; Navy, USA, 2002 and Lunze & Schmid, 2002). The wheel itself consists of a hub carrying a number of free moving rollers angled at 45° about the hub's circumference. The rollers are shaped such that the overall side profile of the wheel is circular. However, wheel slip is a common problem with the Mecanum wheel, particularly when the robot moves sidewise, as it has only one roller with a single point of ground contact at any one time. This severe slippage prevents the most popular dead-reckoning method, using rotary shaft encoders (Everett, 1995 and Borenstein et al, 1996), from being performed well on the Mecanum robot. To cope with the problem, visual dead-reckoning was used as a slip-resilient sensor (Giachetti et al, 1998; Nagatani et al, 2000 and Kraut, 2002). This technique, also used in optical mice, makes use of an on-board video-camera continuously capturing frames of the ground beneath and image processing hardware on the robot determining the speed and direction in which the current frame has moved relative to the previous frame thus allowing the speed and direction of that point of reference to be calculated. However, visual dead-reckoning using a single camera or optical mouse can not provide all three-degree-of-freedom positional information for robot navigation and motion control. Fixed line following is the simplest and most reliable solution, yet is also the most limiting. A physical line is marked on the ground along the path which the robot is to follow (Everett, 1995 and Borenstein et al, 1996). For a robot that is set up in a fixed location for a set task this system is effective but for a research robot with omni-directional capability this approach is seen to be a primitive, though still viable, option.

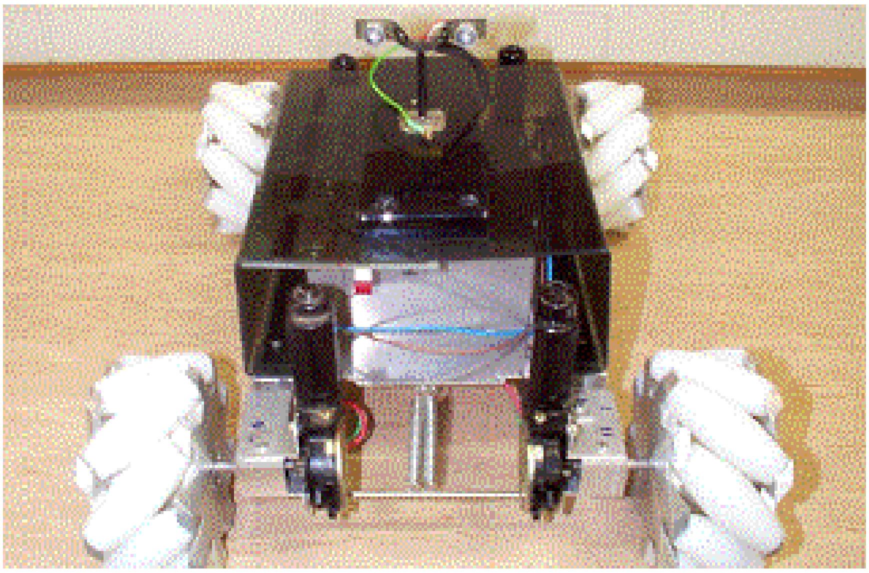
This chapter presents a research project recently completed at Massey University, New Zealand. The research started upon an existing omni-directional platform built of a box-like aluminium chassis, four electric window winder motors and four Mecanum wheels (Phillips, 2000). The aim of this project was to provide the platform with motion control that could be programmed to accommodate various robotic behaviours specified. With respect to the path following behaviour, two optical mice were attached to give positional feedback for closed-loop control and dead-reckoning for navigation and a Mitsubishi M16C/62 microcontroller was interfaced and programmed to implement robotic behaviours. A closed-loop control in Cartesian space was proposed to control x and y-movement and rotation motions of the robot.

As this was a project incorporating mechanical, electrical and software development, a mechatronics design principle was applied. The different areas were developed synergistically thus allowing interactions between the disciplines to be viewed and managed. It also meant that all three core disciplines needed to be developed to a certain stage before any one area could be further worked on. Although it was physically possible to use other means to develop the core areas independently, a synergistic approach tends to be more efficient. Even though this parallel design approach was used, the areas of development shall be discussed in sections assuming that other sections have already been completed to a certain level and are referenced where necessary.

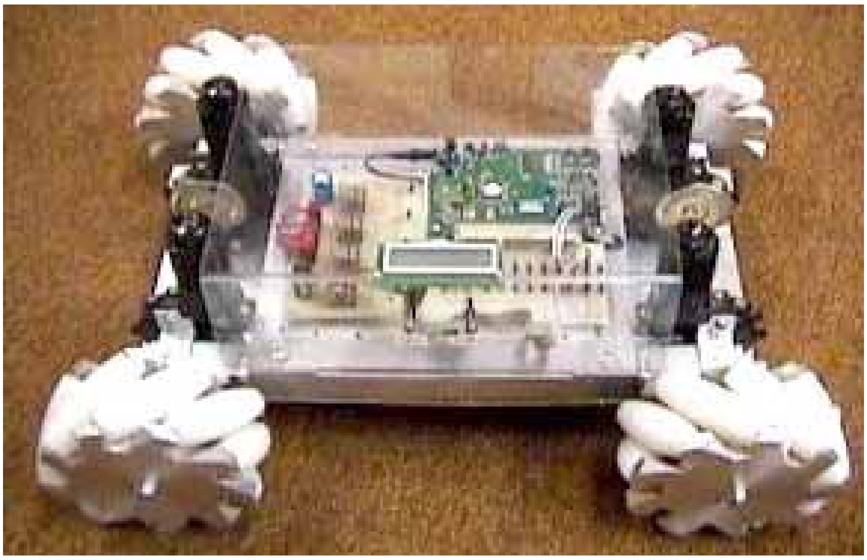
2. Robot Chassis

The original construction of the robot chassis (Phillips, 2000) included a form of "suspension". The motor and wheel assembly at each corner of the chassis was mounted on a single shaft that allowed one degree-of-freedom between it and the chassis about the longitudinal axis. As shown in Figure 1(a) there was no mechanism designed to limit or control this degree-of-freedom and, once constructed, left a free moving joint. The two side assemblies at either end were subsequently linked via a piece of thin tubing in an attempt to limit and effectively prevent movement of the joints. A pair of springs had been added in an attempt to stiffen the setup further.

Some investigations were made into the system and it quickly became apparent that movement about the free axis was completely undesirable as it would allow the sides of the hub to foul the ground. Movement of this form would also affect the motion of the Mecanum wheel, as the sideways force vector generated would work directly against the "suspension". When the wheel pivoted it would alter the current contact patch, thus altering the dynamics of the system. Early testing of the platform using the prototype driver board found this temporary solution unsuitable as it collapsed after only a short period of operation. The robot was to be used on level floors indoors, such as carpeted surfaces, so suspension considerations were therefore not warranted (Braunl, 1999). The motor and wheel assemblies were subsequently welded square to the chassis to provide a solid, robust, cheap and quick solution. Figure 1(b) shows the modified chassis.



(a) Original chassis with pivot system (Philipps, 2000)



(b) The final robot after modifications

Figure 1. Mobile Platform with Four Mecanum Wheels

The other hardware modifications were as follows: the electronics were mounted on top of the chassis on stand-offs; a clear polycarbonate cover was formed to enclose the electronics and provide additional mounting space but still allow access to the circuitry and plugs via the open ends; a Liquid Crystal Display (LCD) was screwed up to the inside surface of the cover and power and program switches were positioned on the side of the cover; and a 18Ahr Sealed Lead-Acid was mounted beneath the robot platform. Major physical properties were measured and are given in Table 1.

Wheels		**Chassis**		**Overall**	
Diameter	150 mm	Length	463 mm	Length	543 mm
Width	75 mm	Width	260 mm	Width	460 mm
Mass	1.94 kg	Height	203 mm	Height	218 mm
Moment of Inertia	0.00546 kgm^2	Mass	12.15 kg	Mass	19.91 kg
Number of Rollers	9	Moment of Inertia	0.223 kgm^2		
Motor Torque	8.16 Nm				

Table 1. Physical Properties of Robot Platform

3. Motor Drive

The robot uses four used automotive electric window winder motors to drive the Mecanum wheels. These units are powered by a 12V DC motor which in turn is connected to a self-contained worm drive reduction. The output gear is connected via a rubber cushion drive. The reduction ensures that the motors provide considerable torque for driving the system. One disadvantage with these inexpensive used motors is that the high current demand of these motors requires a substantial portable power source.

The specifications developed for the motor driver board were:

(1) The circuit should be compatible with a single logic-level PWM input signal for the speed control of each wheel and a single logic-level input line for the direction of motor rotation for each wheel.

(2) The circuit should be able to operate with a high PWM carrier frequency (16 kHz or greater) to provide inaudible operation.
(3) The circuit would require four independent H-Bridge drivers.
(4) Each H-Bridge driver circuit must be capable of providing three amps continuous current at 12V DC.

The desired approach for the circuit design was to use dedicated motor driver Integrated Circuits (ICs) that were suitably rated. This would provide benefits of being compact, simple and being able to directly interface to a microcontroller. Investigation into the availability and pricing for dedicated ICs revealed that the options were too expensive to buy in the limited quantities required for this project. Instead the decision was made to construct a prototype quad MOSFET and relay H-Bridge driver circuit on Vero board to test the concept and enable further progression of the project. Each H-Bridge circuit consisted of one logic-level N-Channel 45Amp power MOSFET (PHB 45N03LT) and a Double-Pole Double-Throw (DPDT) relay. A signal transistor was used to interface the relay with logic-level signals. A Schottky diode was placed across the motor outputs to catch transient voltage spikes generated during the high speed switching of the MOSFET as a result of the inductive nature of motor coils. Figure 2 shows the schematic for a single H-Bridge. Four replicas of this circuit were built on the prototype board.

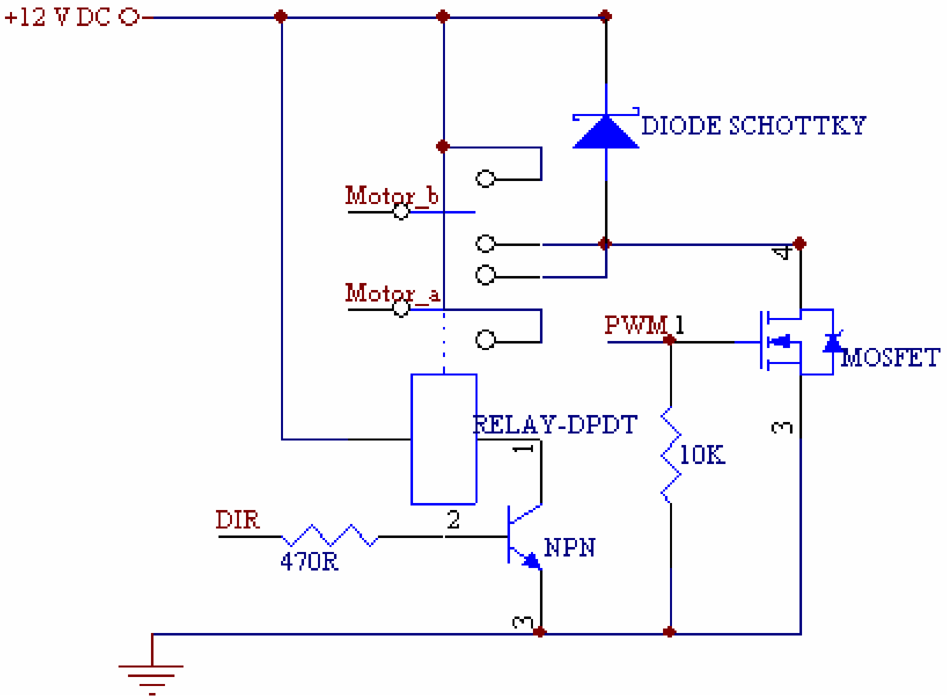


Figure 2. Single Prototype H-Bridge Schematic

Testing proved that this circuit design performed well and, with switching speeds of up to 20 kHz and full load applied to the motors, the MOSFETS did not overheat beyond their specification. However, one area of concern with the design was that relays, being a mechanical part, are subject to wear. This wear is greatly increased if they are switched at high current and high speed (in excess of 10Hz). This application differed though because the relay should only be switched when the motor is passing through zero speed and changing direction. Hence there should be no current flowing at the time the relay is switched. The circuit design was altered to include a charging jack and a voltage divider to provide battery status for use later in the project. The design was then laid out to create a Printed Circuit Board (PCB) for the circuit. The PCB was manufactured using available in-house equipment and implemented as a permanent solution.

4. Microcontroller

In order to give the existing robot any intelligent functionality some form of on-board processor was essential. Microcontrollers are ideally suited for such an application as they are compact, have many built-in hardware features such as timers and UARTS, have a significant number of digital I/O lines and have low power requirements. The essential microcontroller specification for this project was its ability to generate four independent PWM signals at frequencies greater than 15 kHz and with at least 8-bit resolution at these high frequencies. Other general requirements were; high speed operation to ensure environmental data could be processed at real-time, large RAM and ROM for complex algorithms, at least four 10-bit Analogue-To-Digital Converters (ADCs) to interface a sensory array and more than 30 digital I/O lines to interface peripherals such as an LCD display. A Mitsubishi M16C/62 was employed as it was available free of charge in the lab and also met the aforementioned requirements.

Connector	Pin Number	Function
1B/1C	VCC	Mouse 1, 2 & LCD +5V DC
2A	97	Battery voltage signal
5A	88	LCD data - DB0 (pin 7)
6C	87	LCD data - DB1 (pin 8)
6B	86	LCD data - DB2 (pin 9)
6A	85	LCD data - DB3 (pin 10)
7C	84	LCD data - DB4 (pin 11)
7B	83	LCD data - DB5 (pin 12)
7A	82	LCD data - DB6 (pin 13)
8C	81	LCD data - DB7 (pin 14)
8B	80	LCD function – RS (pin 4)
8A	79	LCD function – R/W (pin 5)
9C	78	LCD function – E (pin 6)
9A	76	Mouse 1 PS/2 bus – clock
10C	75	Mouse 2 PS/2 bus – clock
10B	74	Mouse 1 PS/2 bus – data
10A	73	Mouse 2 PS/2 bus – data
25C	28	Motor driver signal – PWM_1
25B	27	Motor driver signal – DIR_1
25A	26	Motor driver signal – PWM_2
26C	25	Motor driver signal – DIR_2
26B	24	Motor driver signal – PWM_3
26A	23	Motor driver signal – DIR_3
27C	22	Motor driver signal – PWM_4
27B	21	Motor driver signal – DIR_4
32B/C	GND	Mouse 1, 2 & LCD OV DC

Table 2. Pin Allocation for M16C/62 Microcontroller

The M16 is a 16 bit microcontroller and can be programmed using the high level 'C' language. The board contains many functions such as built-in timers, both hardware and software interrupts, A-D converters and 87 digital I/O lines. Its other major features are: 256KB ROM, 20KB RAM, two 7-segment LEDs, 16MHz main clock & 32KHz subclock, reset IC, switches, 5V regulator, RS232 driver chip, 96-pin DIN connector for application board interface. A pin allocation of the microcontroller interfacing for the robot is given in Table 2.

5. LCD Display for Robot Status

A 16 Character x 2 line LCD (Dick Smith Electronics – Z 4170), which is directly microcontroller compatible, was interfaced using an 8-bit bi-directional data bus. Three other Digital I/O lines were required for interfacing the Data / Instruction Select, Read / Write Select and Signal Enable lines of the display module. It was necessary to write a software driver for the display. The code was written as a module that could be easily used in any other programs. The software was derived from the data sheet provided with the display and example code found for similar displays.

The programs developed for this project required real-time communications and the use of an onboard UART and RS232 connector. As a result the PC software debugging suite could no longer be used. Instead the LCD was primarily used throughout the software development stage as a debugging tool. The final programs used the display to provide visual feedback to the user about the current status of the robot's operation, including position and battery status.

6. Optical Mice for Dead-Reckoning

Both navigation and path following require the robot to know its location. It was considered that a system using optical mice would provide greater flexibility when compared to other options (e.g., shaft encoder based dead reckoning, image based line following, beacon reference system).

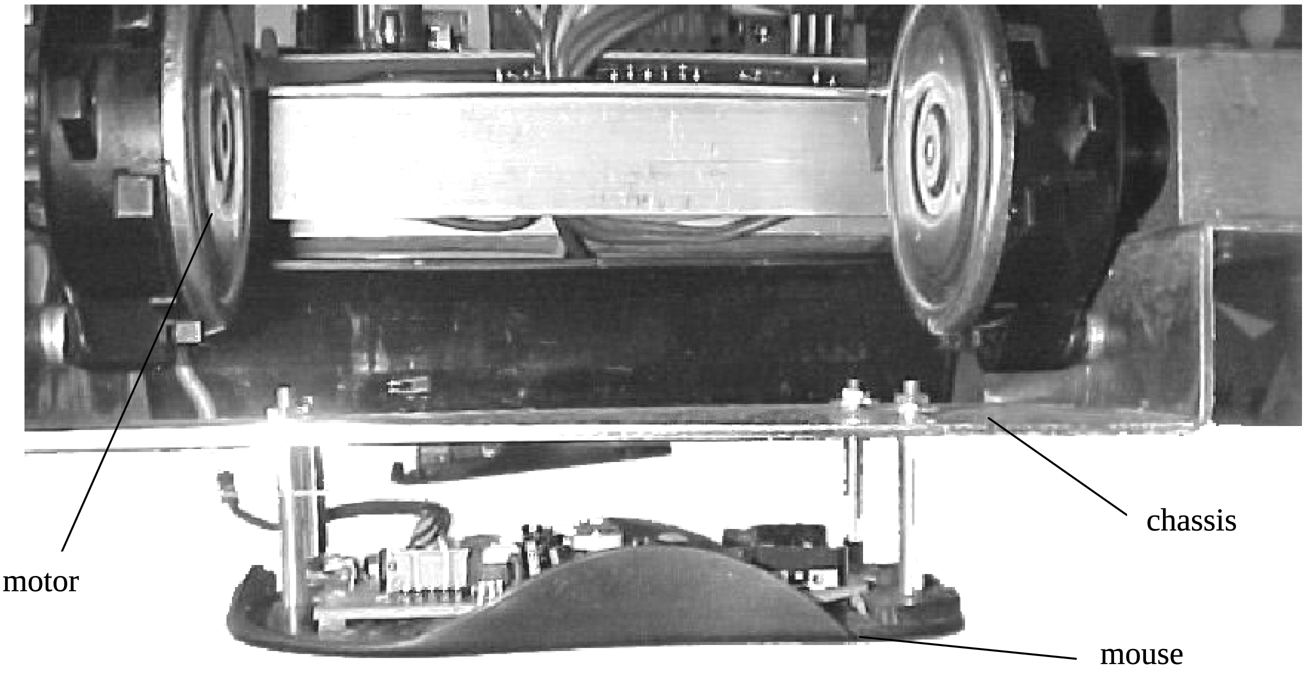


Figure 3. The Mounting of an Optical Mouse

An optical mouse has a tiny camera, a Digital Signal Processor (DSP) and microcontroller on-board which can provide a continuous data stream of x and y movements. Since an optical mouse can only measure translational movement not rotational movement, two optical mice were needed to sense motion in the robot's three degrees of freedom. By placing a mouse at either end of the robot looking down to the floor, as shown in Figure 3, the difference between the front and rear x-displacements is proportional to the angular rotation of the robot.

6.1 PS / 2 Protocol for Optical Mouse

Optical mice that were readily available could be interfaced via either the USB or PS/2 protocol and two A4Tech SWOP-35 optical mice were bought for NZ$38 each. However, since the M16C/62 microcontroller does not have hardware USB or PS/2 capabilities, a software PS/2 driver was derived and written from the specifications for the PS/2 mouse protocol.

The PS/2 protocol uses a two wire bi-directional serial bus. Both wires are open collector and can be pulled low by the host (microcontroller) or device (mouse). One wire is the clock signal which is provided by the device at anywhere between 10 and 20 kHz. The clock line can be held low by the host to inhibit the PS/2 bus. In this scenario the device will accumulate displacement until it overflows at which point data is lost. The second wire is the data signal which uses standard serial framing, one start bit, eight data bits, an odd parity bit and one stop bit (8-O-1). The Least Significant Bit (LSB) of the data is always sent first. The Clock line of the PS/2 bus is connected to a general purpose I/O line and the Data line is connected to one of the hardware interrupt lines of the M16C/62 microcontroller. A basic summary of PS/2 communications is as follows,

<u>Device to Host communications (Figure 4)</u>

- The device pulses the Clock line a total of 11 times, while transmitting the start bit, data bits, parity bit and stop bit on the Data line.
- The Host is expected to sample the Data line each time the Clock is low and the Device changes the state of the Data line while the Clock is high.

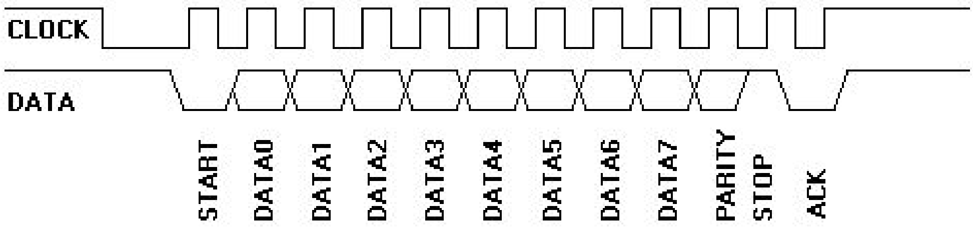


Figure 4. Device to Host Communications (Chapweske, 2002)

<u>Host to Device communications (Figure 5)</u>

- The Host signals its intent to transmit a command or argument byte by holding the Clock line low for at least 100µs, before pulling the Data line low and releasing the Clock line.
- The Device pulses the Clock line a total of 11 times to receive a byte of data. The Host is expected to change the state of the Data line while the Clock is low and the Device samples the Data line while the Clock is high.

- After the tenth Clock pulse, the Device checks for a valid stop bit (Data line high), and responds by pulling the Data line low and clocking one more time.

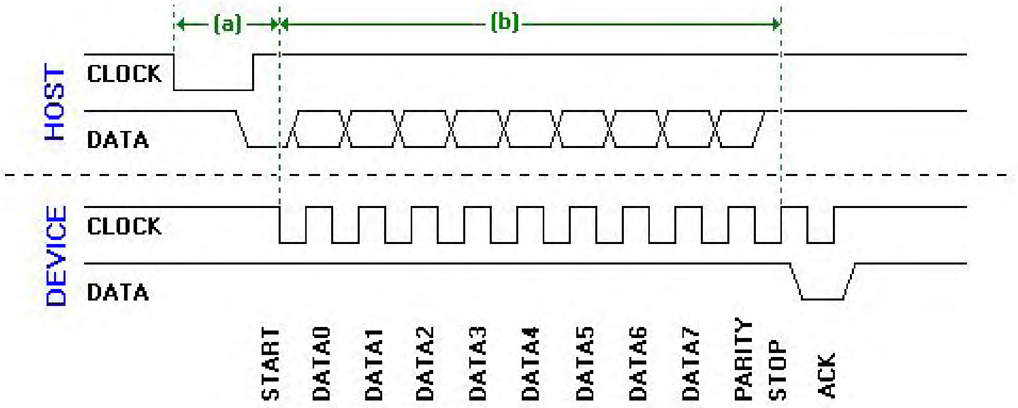


Figure 5. Host to Device Communications (Chapweske, 2002)

6.2 M16C/62 Software Driver for Optical Mouse

The software driver written for the M16C/62 waits for the mouse to complete its power up self test and then issues a command to the mouse to enable data reporting (0xF4). By default, the mouse starts up in stream mode in order to continually generate x and y displacement data. This data is relative to the previous set of data whenever there is motion. However, the data is only transmitted on the PS/2 bus if data reporting is enabled. The acknowledge command (0xFA) is then expected to be returned, if it is not, the user is requested to reset the system. Once the mouse is initialised, the Clock line is held low by the microcontroller to inhibit the bus. At this point the main program is able to allow the Clock line to float high and read the data in when an interrupt is caused on the Data line. In default start up, the mouse operates in a standard PS/2 compatible mode. The data sent in this mode consists of a three byte packet. The first byte is a header, the second is the x data and the third the y data. Table 3 shows the detail of the standard PS/2 mouse data packet.

The optical mouse provides approximately 400 CPI (counts per inch) but testing proved that a value of 430 CPI, or 17 counts per mm was needed to properly calibrate the displacement data given by the mouse. The movement data is in 9-bit two's complement, with the ninth bit located in the packet header.

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Byte 1	Y overflow	X overflow	Y sign bit	X sign bit	Always 1	Middle Btn	Right Btn	Left Btn
Byte 2	X Movement							
Byte 3	Y Movement							

Table 3. Formatting of the PS/2 Mouse Data Packet

The mouse can also utilise the corresponding overflow bit in the header to increase its displacement accumulation buffer to 10-bit two's complement (± 512 counts).
This limits the displacement of the mouse between readings to *512/17 = 30mm per reading*. To enable the mouse to travel at 1m/sec it would need to be serviced every *30/1000 = 0.03 seconds*.

6.3 Testing

Preliminary testing was performed by using one mouse as an electronic ruler. The raw data received from the mouse was formatted according to the packet header. This data was accumulated for both the x and y-direction without scaling to maintain accuracy. The x and y-displacement accumulator values were subsequently scaled to millimetres to be displayed on the LCD panel. The parity of the data was also checked and an error was displayed on the LCD panel if a parity error occurred, to give an indication of the frequency of errors.
A mouse was temporarily attached to the front of the robot to test whether the performance of the measurement provided would be sufficient in the application. The tests showed that the optical mice provided better than 1% accuracy when the displaced measurement was compared to that given by a tape measure. Parity errors did occur but were very infrequent. The errors were most likely to occur if the mouse data cable was placed in close proximity to any one of the DC motors. To minimise the noise to which the data cables were subjected they were routed centrally down the chassis, while the DC motor power cables were routed through the side rails.
The final stage in fully implementing the optical mice on the robot was to create mounting brackets to hold the mice securely in the x and y-direction. Because the mice needed to be sitting flush with the ground it was necessary to implement some system to allow vertical movement. The mice were stripped down to the base and four shafts were mounted, one in each corner. These shafts were free to slide up and down in holes in the mounting bracket. Initially springs were used to provide some form of assisted return for the mice but testing proved that this added too much drag on the ground. Instead gravity proved to be sufficient to hold the mice to the ground.

7. Planned Path Motion

7.1 Open Loop Testing

With the mice installed it was possible to assess the open loop performance of the robot quantitatively, thus creating a benchmark for any further development. A standard planned path was designed that would provide the best assessment of the specifications. The path consisted of straight forwards, reverse, sideways and 45° diagonal lines. The 45° diagonal path was considered to be the most demanding of the robot as it is only driving with two wheels to achieve this direction. The standard testing path is shown in Figure 6. Due to the limited space available for testing, the maximum displacement from the centre of the course was 800mm. Table 4 gives the Cartesian coordinates for the path as used in the software.
The speed for forward and reverse movement was 152.88 mm/sec which gives 5.25 sec for 800 mm, the speed for sidewise movement was 152.28 mm/sec which gives 5.25 sec for 800 mm and the speed for diagonals was 167.19 mm/sec which gives 6.77 sec for 1130 mm.

	0	1	2	3	4	5	6	7	8	9
X	0	0	800	0	-800	-800	0	800	800	0
Y	0	800	800	0	-800	0	0	0	-800	0

Table 4. Planned Path Cartesian Coordinates

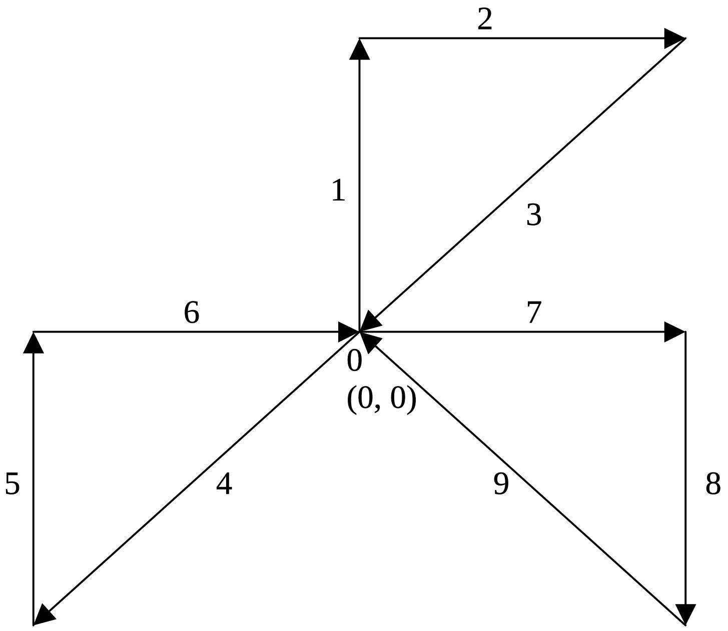


Figure 6. Planned Path for Testing

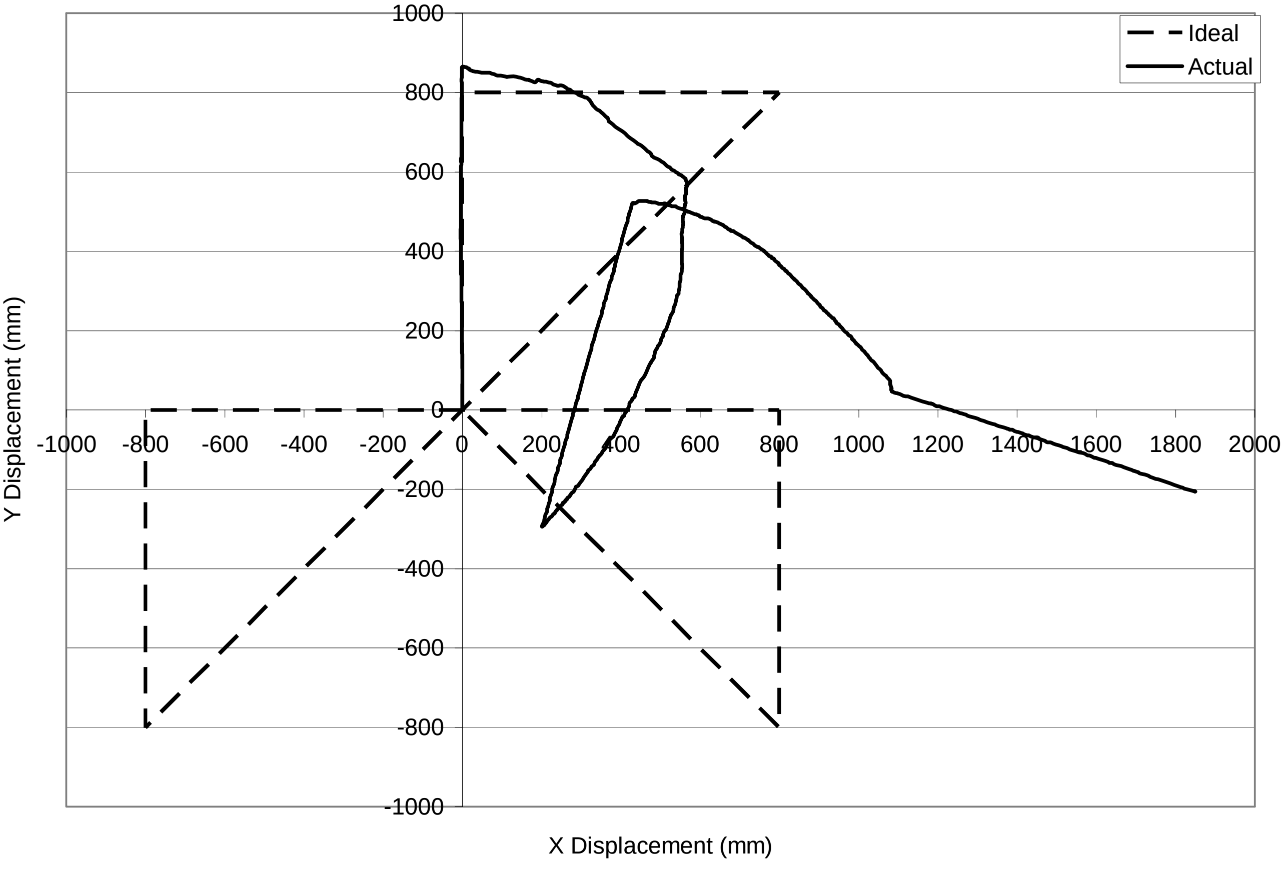


Figure 7. Open Loop Path Following Performance

7.2 Closed-Loop Control

In order to let the robot follow a planned path a closed loop control in Cartesian space was required. The simplest form of control for this multi-input, multi-output system is an array of Proportional Integral Derivative (PID) controllers as it requires no modelling of the system and is efficient to calculate for use in a fast acting real time application. The speed of the robot in the x and y-direction and rotation is controlled to bring the robot to the desired position otherwise known as a velocity servo controller. The problem was simplified by controlling the rotational position of the platform to a set-point of 0° and so simplifying the task of the microcontroller as the need to calculate trigonometry functions at real time, which is highly demanding on the processor, was alleviated. The set-points for the x, y and rotational speed controllers are proportional to the error between the desired x, y and rotational position and the actual positions. The desired value for the x and y position is calculated from the equation for a straight line ($y = mx + c$), between the previous and the current destination.

The three PID controllers were tuned using the Ziegler-Nichols Ultimate Gain method (Jobling, 2002) via three potentiometer voltage dividers connected to ADCs on the microcontroller. These were continuously scanned and scaled appropriately. Refer to Table 5 for the actual PID tuning constants. The software also streamed positional data through the UART to be logged and analysed by a PC. The path followed by the robot using closed loop control can be seen in Figure 8. A dramatic improvement can be seen over the performance of the open loop system.

	Proportional (P)	Integral (I)	Derivative (D)
X-Controller	3.38	0.154	1.02
Y-Controller	1.96	0.205	1.04
Rotational-Controller	2.72	0.343	1.21

Table 5. PID Tuning Constants

7.3 Results from Testing

Qualitative observations were made throughout the initial hardware development and modification stage of the project. These observations confirmed that modifications did increase the quality of the motion of the robot. The modifications to the pivot system had a successful result as they returned the chassis to a usable state that allowed further work and development to be done on the project.

Quantitative testing and analysis of the system was performed on the planned path motion application. The actual path travelled by the robot was compared to the desired path specified. Given in Table 6 are the results for this test calculated for both open loop control and closed loop control.

The specifications for the planned path motion were that the robot should not deviate more than ±1mm from the desired path for any forward or reverse movement, or ±5mm for any arbitrary angled movement within a two meter square area. The test data shows that neither of these specifications was met with open loop control. The implementation of closed loop control enabled the forward and reverse specification to be met. The diagonal path did prove to be difficult to control and although its performance lay outside the defined specification, it was a significant improvement over the open loop control.

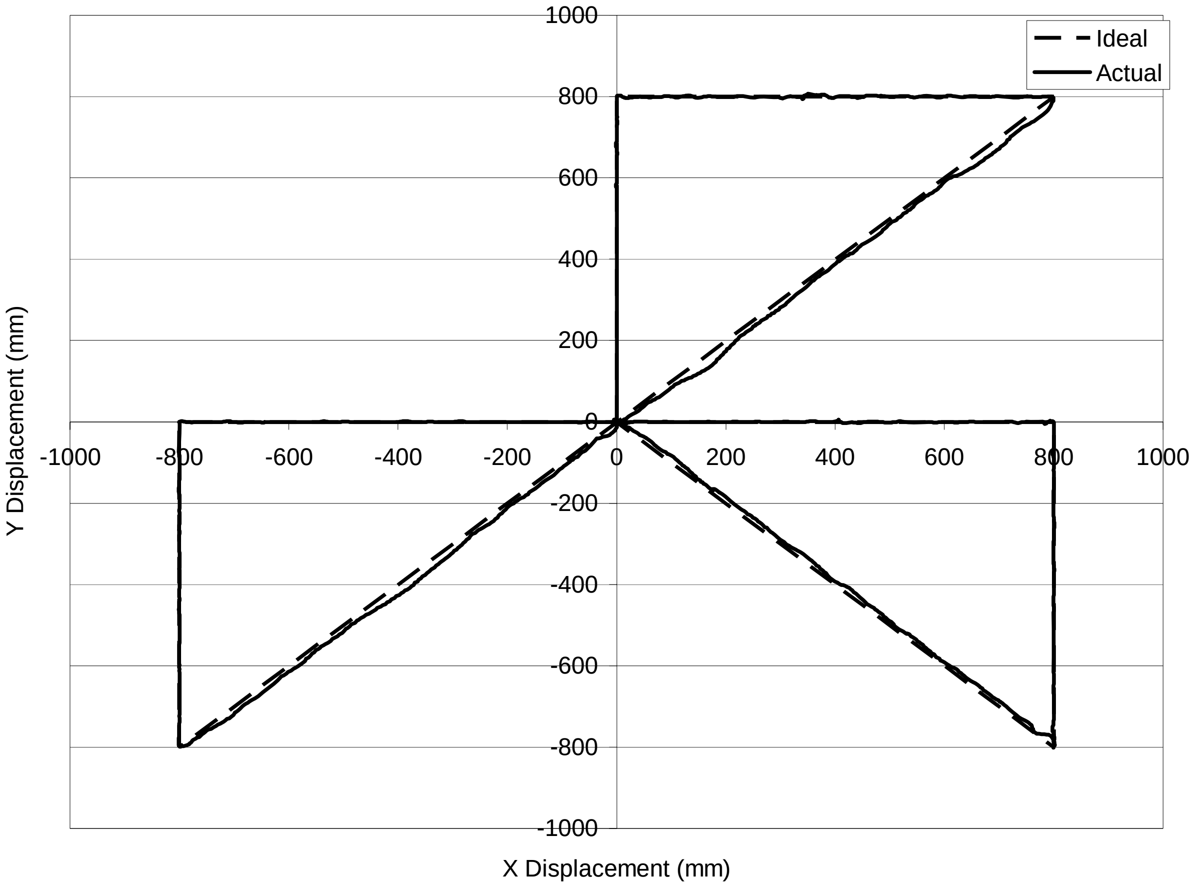


Figure 8. Closed Loop Path Following Performance

Open Loop Control				
	Forwards/Reverse	Diagonal (mm)	Left/Right	Rotation
Mean	-2.29	-47.72	415.74	-39.92
Standard Deviation	1.56	98.43	136.27	64.13
Minimum	-5.00	-239.00	6.00	-137.26
Maximum	0.00	65.00	496.00	27.69
Closed Loop Control				
Mean	0.04	9.56	-0.08	-0.07
Standard Deviation	0.65	8.49	1.43	0.72
Minimum	-2.00	-6.00	-6.00	-3.01
Maximum	1.00	36.00	7.00	7.53
Improvement	242%	1160%	9559%	8906%

Table 6. Results of Planned Path Testing

8. Conclusions

The motor driver circuit board designed for the drive of the Mecanum wheels met all the specifications given in Section 3. The limitation to its current handling capacity was the relays which are rated at five amps. The board has provided a cheap and compact interface between the microcontroller and the DC motors, fitting without trouble in the available space on the robot chassis. It is considered adequate to support further work done on the robot. Optical mice were successfully implemented on the system to provide positional measurement data to the microcontroller. They provided a cheap and accurate method of dead-reckoning for robot navigation, while making use of current technology in an application that is not currently common practice.

Wheel slip continued to be an issue with the Mecanum design. This problem was significantly improved by the use of closed loop control. PID controllers created a sturdy control design to investigate the platforms closed loop performance. Although the specifications set for the control task where not fully met, the performance of the platform was greatly improved.

The platform has been completed to a standard whereby further projects can use the robot to conduct advanced multivariable control investigations. The robot has been presented as a complete package, including a battery charger, in order to increase the ease of further research done on the project. If optimal system performance is required it is recommended that some modifications to the existing Mecanum wheels be made. This may necessitate new rollers being machined from a more appropriate material and, most likely, involve some form of rubber coating. To develop more sophisticated control of and telemetry from the platform, a wireless connection should be included. This would allow trigonometric, dynamic path and status calculations to be off-loaded from the microcontroller.

Acknowledgement

The work presented in this chapter is based on the research projects completed by J.G. Phillips and J.A. Cooney, Massey University, New Zealand, in 2000 and 2002, respectively.

9. References

Borenstein, J.; Everett, H.R.; Feng, L. (1996). Navigating Mobile Robots: Sensors and Techniques. A K Peters, Ltd, MA, USA.

Bräunl, T. (1999). Eye-Bot: a family of autonomous mobile robots. In: Proceedings of 6th International Conference on Neural Information Processing, pp. 645-649, November, 1999, Perth, Australia.

Chapweske, A. (2002). PS/2 mouse and keyboard protocol. Available from: http://panda.cs.ndsu.nodak.edu/~achapwes/ PICmicro/PS2/ps2.htm. accessed: October 2002.

Dickerson, S.L.; Lapin, B.D. (1991). Control of an omni-directional robotic vehicle with Mecanum wheels. In: National Telesystems Conference Proceedings, pp.323-328, March 26-27, 1991, Atlanta, USA.

Everett, H.R. (1995). Sensors for Mobile Robots: Theory and Application. A K Peters, Ltd, MA, USA.

Giachetti, A.; Campani, M.; Torre, V. (1998). The use of optical flow for road navigation. IEEE Transactions on Robotics and Automation. Vol.,14, No.,1, pp.34-48.

Jobling, C.P. (2002). Zeigler-Nichols tuning of PID compensators. Available from: http://eespectre.swan.ac.uk/~eechris/ ee306/docs/zeigler.doc. Accessed: October 2002.

Kraut, J. (2002). An Autonomous Navigational Mobile Robot. Bachelor of Science Thesis, Department of Electrical and Computer Engineering, University of Manitoba, Manitoba, Canada.

Lunze, J.; Schmid, C. (2002). Regelung einer mobilen Plattform via Internet. available from: http://www.esr.ruhr-uni-bochum.de/lehre/P1/P1_Fahrzeug_V6.pdf. accessed: October 2002.

Muir, P.F.; Neumann, C.P. (1990). Kinematic modeling for feedback control of an omnidirectional wheeled mobile robot. In: Autonomous Robot Vehicles, Editors I.J. Coxy and G.T. Wilfong, pp.25-31, Springer-Verlag, Berlin.

Nagatani, K.; Tachibana, S. et al. (2000). Improvement of odometry for omnidirectional vehicle using optical flow information. In: Proceedings of IEEE/RSJ Int. Conference on Intelligent Robots and Systems, pp.468-473, Oct 30-Nov 5, 2000, Takamatsu, Japan.

Navy, USA. (2002). Omni-directional vehicles, Available from: http://robots.net/robomenu/996604291.html, Accessed: October 2002.

Phillips, J. G. (2000). Mechatronic Design and Construction of an Intelligent Mobile Robot for Educational Purposes. Master of Technology Thesis, Massey University, Palmerston North, New Zealand.

-II-

Perception

Tracking Skin-Colored Objects in Real-time

Antonis A. Argyros & Manolis I.A. Lourakis

1. Introduction

Locating and tracking objects of interest in a temporal sequence of images constitutes an essential building block of many vision systems. In particular, techniques for effectively and efficiently tracking the human body, either in part or as a whole, have received considerable attention in the context of applications such as face, gesture and gait recognition, markerless human motion capture, behavior and action interpretation, perceptual user interfaces, intelligent surveillance, etc. In such settings, vision-based tracking needs to provide answers to the following fundamental questions. First, how is a human modeled and how are instances of the employed model detected in an image? Second, how are instances of the detected model associated temporally in sequences of images?

Being a complex, non-rigid structure with many degrees of freedom, the human body is intricate to model. This is reflected on the models that have been employed in the literature for human tracking, whose type and complexity vary dramatically (Gavrila, 1999; DeCarlo & Metaxas, 2000; Delamarre & Faugeras, 2001; Plänkers & Fua, 2001), depending heavily on the requirements of the application domain under consideration. For example, tracking people in an indoors environment in the context of a surveillance application has completely different modeling requirements compared to tracking the fingers of a hand for sign language interpretation.

Many visual cues like color, shading, edges, texture, motion, depth and their combinations have been employed as the basis for modeling of human body parts. Among those, skin color is very effective towards detecting the presence of humans in a scene. Color offers significant advantages over geometric models, such as robustness under occlusions, resolution changes and geometric transformations. Additionally, color is a natural cue for focusing attention to salient regions in an image and the computational requirements for processing it are considerably lower compared to those associated with the processing of complex geometric models. In the remainder of this section, we briefly review existing approaches based on the answers they provide to the two fundamental questions stated above.

1.1 Modeling and Detection of Color

A recent survey (Yang et al, 2002) provides an interesting overview concerning the use of color for face (and, therefore, skin-color) detection. A major decision towards deriving a model of skin color relates to the selection of the color space to be employed. Several color spaces have been proposed including RGB (Jebara & Pentland, 1997), normalized RGB (Kim et al. 1998; Jones & Rehg, 1999), HSV (Saxe & Foulds, 1996), YCrCb (Chai & Ngan, 1998), YUV (Yang & Ahuja 1998), etc. Color spaces efficiently separating the chrominance from the luminance components of color are typically considered preferable. This is due to the fact that by employing the chrominance-dependent components of color only, some degree of robustness to illumination changes can be achieved. A review of different skin chrominance models and a comparative evaluation of their performance can be found in (Terrillon et al., 2000).

Once a suitable color space has been selected, the simplest approach for defining what constitutes skin color is to employ bounds on the coordinates of the selected space (Chai & Ngan, 1998). These bounds are typically selected empirically, i.e. by examining the distribution of skin colors in a pre-selected set of images. A more elaborate approach is to assume that the probabilities of skin colors follow a distribution that can be learnt either off-line or by employing an on-line iterative method (Saxe & Foulds, 1996). Depending on whether this distribution is represented analytically or not, existing approaches can be classified as parametric or non-parametric.

Non-parametric approaches represent the learnt distribution by means of a histogram of color probabilities. Parametric approaches are based either on a unimodal Gaussian probability density function (Kimet et al., 1998, Yang & Ahuja, 1998) or on multimodal Gaussian mixtures (Jebara et al., 1998; Raja et al., 1999) that model the probability distribution of skin color. The parameters of a unimodal Gaussian density function are estimated using maximum likelihood estimation techniques. Multi-modal Gaussian mixtures typically require the Expectation-Maximization (EM) algorithm (Dempster et al., 1977) to be employed. According to Yang et al (Yang & Ahuja, 2001), a mixture of Gaussians is preferable compared to a single Gaussian distribution. Still, Jones and Regh (Jones & Regh, 1999) argue that histogram models provide better accuracy and incur lower computational costs compared to mixture models for the detection of skin-colored areas in an image. A few of the proposed methods perform some sort of adaptation to become insensitive to changes in the illumination conditions. For instance, (Raja et al., 1999) suggest adapting a Gaussian mixture model that approximates the multi-modal distribution of the object's colors, based on a recent history of detected skin-colored regions. Vezhnevets et al (Vezhnevets et al., 2003) provide a survey of published pixel-based skin detection methods.

1.2 Tracking

Assuming that skin-colored regions have been appropriately modeled and can be reliably detected in an image, another major issue relates to the temporal association of these observations in an image sequence. The traditional approach to solving this problem has been based on the original work of Kalman (Kalman, 1960) and its extensions. If the observations and object dynamics are of a Gaussian nature, Kalman filtering suffices to optimally solve the tracking problem. However, in many practical cases the involved distributions are non-Gaussian and, therefore, the underlying assumptions of Kalman filtering are violated.

As suggested in (Spengler & Schiele, 2003), recent research efforts that deal with object tracking can be classified into two categories, namely those that solve the tracking problem in a non-Bayesian framework (e.g. Javed & Shah 2002; Siebel & Maybank, 2002; Triesch & von de Malsburg, 2001) and those that tackle it in a Bayesian one (e.g. Isard & Blake 1998; Koller-Meier & Ade, 2001; Hue et al., 2002). In most of the cases (Isard & Blake, 1998), the problem of single-object tracking is investigated. These single-object approaches usually rely upon sophisticated, powerful object models. Other studies such as (Koller-

Meier & Ade, 2001; Hue et al., 2002) address the more general problem of tracking several objects in parallel. Some of these methods employ configurations of several individual objects, thus reducing the multi-object tracking problem to a set of instances of the less difficult single-object tracking problem. Other methods employ algorithms making use of particle filtering (Arulampalam et al., 2002), i.e. sequential Monte-Carlo generalizations of Kalman filtering that are based on sampled representations of probability densities. Despite the considerable amount of research that has been devoted to tracking, an efficient and robust solution to the general formulation of the problem is still lacking, especially for the case of simultaneous tracking of multiple objects.

The rest of the paper is organized as follows. Section 2 provides an overview of the proposed skin color tracker. Sections 3 and 4 present the operation of the proposed tracker in more detail. Section 5 provides sample results from the application of the tracker to long image sequences and discusses issues related to its computational performance. Finally, section 6 provides the main conclusions of this work along with an outline of possible extensions to it.

2. Overview of the Proposed Approach

With respect to the two fundamental questions that have been posed in the introductory section, the proposed approach relies on a non-parametric method for skin-color detection and performs tracking in a non-Bayesian framework. A high level description of the operation of the proposed method for tracking multiple skin-colored objects is as follows. At each time instant, the camera acquires an image on which skin-colored blobs (i.e. connected sets of skin-colored pixels) are detected. The method also maintains a set of object hypotheses that have been tracked up to the current instant in time. The detected blobs, together with the object hypotheses are then associated in time. The goal of this association is twofold, namely (a) to assign a new, unique label to each new object that enters the camera's field of view for the first time, and (b) to propagate in time the labels of already detected objects that continue to be visible.

Compared to existing approaches, the proposed method has a number of attractive properties. Specifically, the employed skin-color representation does not make use of prohibitively complex physics-based models and is learned through an off-line procedure. Moreover, a technique is proposed that permits the avoidance of much of the burden involved in the process of manually generating training data. Being non-parametric, the proposed approach is independent of the shape of skin color distribution. Also, it adapts the employed skin-color model based on the recent history of tracked skin-colored objects. Thus, without relying on complex models, it is able to robustly and efficiently detect skin-colored objects even in the case of changing illumination conditions. Tracking over time is performed by employing a novel technique that can cope with multiple skin-colored objects, moving in complex patterns in the field of view of a possibly moving camera. Furthermore, the employed method is very efficient computationally. A prototype implementation of the proposed tracker operates on live video at a rate of 28 Hz on a Pentium IV processor running under MS Windows, without resorting to assembly optimizations or special hardware instructions such as MMX or SSE.

A more detailed description of the approach adopted by the proposed method for solving the two fundamental sub-problems identified in the introduction is supplied in the subsequent sections. An earlier description of the proposed method appears in (Argyros & Lourakis, 2004).

3. Detecting Skin Colored Blobs

Skin color detection in the framework of the proposed method consists of four steps: (a) estimation of the probability of a pixel being skin-colored, (b) hysteresis thresholding on the derived probabilities map, (c) connected components labeling to yield skin-colored blobs and, (d) computation of statistical information for each blob. Skin color detection adopts a Bayesian approach, involving an iterative training phase and an adaptive detection phase. Section 3.1 describes the employed skin color detection mechanisms and sections 3.2 and 3.3 deal, respectively, with simplifying the process of off-line training and introducing adaptiveness to the skin detection procedure.

3.1 Basic Training and Skin Detection Schemes

During an off-line phase, a small set of training input images is selected on which a human operator manually delineates skin-colored regions. The color representation used in this process is YUV 4:2:2 (Jack, 2004). However, the Y-component of this representation is not employed for two reasons. First, the Y-component corresponds to the illumination of an image pixel and therefore, by omitting it, the developed classifier becomes less sensitive to illumination changes. Second, compared to a 3D color representation (i.e. YUV), a 2D one (i.e. UV) is of lower dimensionality and is, therefore, less demanding in terms of memory storage and processing costs.

Assuming that image pixels with coordinates (x, y) have color values $c = c(x, y)$, training data are used to compute (a) the prior probability $(\ )$ of skin color, (b) the prior probability (c) of the occurrence of each color c and (c) the prior probability $(c|\ )$ of a color c being a skin color. Based on this information, the probability $(\ |c)$ of a color c being a skin color can be computed by employing the Bayes rule:

$$(\ |c) = \frac{(c|\ )\ (\ )}{(c)}. \tag{1}$$

Equation (1) permits the determination of the probability of a certain image pixel being skin-colored using a lookup table indexed with the pixel's color. All pixels with probability $(\ |c) > T_{\max}$ are considered as being skin-colored. These pixels constitute seeds of potential skin-colored blobs. More specifically, image pixels with probabilities $(\ |c) > T_{\min}$ where $T_{\min} < T_{\max}$ that are immediate neighbors of skin-colored image pixels are recursively added to each blob. The rationale behind this region growing operation is that an image pixel with relatively low probability of being skin-colored should be considered as such in the case that it is a neighbor of an image pixel with high probability of being skin-colored. A similar type of hysteresis thresholding operation has been proven extremely useful to edge detection (Canny, 1986). Indicative values for the thresholds $T_{\max}$ and $T_{\min}$ are 0.5 and 0.15, respectively. A standard connected components labeling algorithm is then responsible for assigning different labels to the image pixels of different blobs. Size filtering on the derived connected components is also performed to eliminate small, isolated blobs that are attributed to noise and do not correspond to interesting skin-colored regions. Each of the remaining connected components corresponds to a skin-colored blob. The final step in skin color detection involves the computation of central moments up to second order for each blob that, as will be explained shortly, will be needed during the tracking process.

3.2 Simplifying Off-Line Training

Training is an off-line procedure that does not affect the on-line performance of the tracker. Nevertheless, the compilation of a sufficiently representative training set is a time-consuming and labor-intensive process. To cope with this problem, an adaptive training procedure has been developed. Training is performed on a small set of seed images for which a human provides ground truth by defining skin-colored regions. Alternatively, already existing, publicly available training sets such as the "Compaq" skin database of (Jones & Rehg, 1999) can be employed. Following this, detection together with hysteresis thresholding is used to continuously update the prior probabilities $(\ )$, (c) and $(c|\ )$ based on a larger image data set. The updated prior probabilities are used to classify pixels of these images into skin-colored and non-skin-colored ones. In cases where the classifier produces wrong results (false positives / false negatives), manual user intervention for correcting these errors is necessary; still, up to this point, the classifier has automatically completed much of the required work. The final training of the classifier is then performed based on the training set resulting from user editing. This process for adapting the prior probabilities $(\ )$, (c) and $(c|\ )$ can either be disabled as soon as the achieved training is deemed sufficient for the purposes of the tracker, or continue as more input images are fed to the system.

3.3 Adaptive Skin Detection

The success of the skin-color detection process presented in section 3.1 depends critically on whether illumination conditions during the on-line operation of the detector are similar to those during the acquisition of the training data set.

Despite the fact that the UV color representation model used has certain illumination independent characteristics, the skin-color detector may produce poor results if the illumination conditions during on-line operation are considerably different compared to the ones represented in the training set. Hence, a means for adapting the representation of skin-colored image pixels according to the recent history of detected skin-colored pixels is required. To solve this problem, skin color detection maintains two sets of prior probabilities.

The fist set consists of $(\ )$, (c), $(c|\ )$ that have been computed off-line from the training set while the second is made up of $(\ )$, (c), $(c|\ )$, corresponding to the evidence that the system gathers during the most recent frames. Clearly, the second set better reflects the "recent" appearance of skin-colored objects and is therefore better adapted to the current illumination conditions. Skin color detection is then performed based on the following weighted moving average formula:

$$(\ |c)=\gamma\ (\ |c)+(1-\gamma)\ (\ |c), \tag{2}$$

where $(\ |c)$ and $(\ |c)$ are both given by eq. (1) but involve prior probabilities that have been computed from the whole training set and from the detection results in the last frames, respectively. In eq. (2), γ is a sensitivity parameter that controls the influence of the training set in the detection process. Setting $\quad = 5$ and $\gamma = 0.8$ gave rise to very good results in a series of experiments involving gradual variations of illumination.

4. Tracking Multiple Objects over Time

Let us assume that at time t, M blobs have been detected as described in section 3. Each blob ${}_j$, $1 \le j \le M$, corresponds to a set of connected skin-colored image pixels. It should be noted that the correspondence among blobs and objects is not necessarily one-to-one. As an example, two crossing hands are two different skin-colored objects that appear as one blob at the time one hand occludes the other. In this work, we assume that an object may correspond to either one blob or part of a blob. Conversely, one blob may correspond to one or many objects.

We also assume that the spatial distribution of pixels depicting a skin-colored object can be coarsely approximated by an ellipse. This assumption is valid for skin-colored objects like hand palms and faces. Extensive experimentation has demonstrated that the tracker still performs very well even in cases where the shape of skin-colored objects deviates significantly from the shape of an ellipse.

Let $\quad$ be the number of skin-colored objects present in the viewed scene at time t and o_i, $1 \le i \le \quad$, be the set of skin pixels that image the i-th object. We also denote with ${}_i = {}_i(c_{x_i}, c_{y_i}, \alpha_i, \beta_i, \theta_i)$ the ellipse model of this object where (c_{x_i}, c_{y_i}) is its centroid, α_i and β_i are, respectively, the lengths of its major and minor axis, and θ_i is its orientation on the image plane. Finally, we use capital letters $= \bigcup_{j=1}^{M} {}_j$, $= \bigcup_{i=1} o_i$, and $= \bigcup_{i=1} {}_i$ to denote the union of all skin-colored pixels, object pixels and ellipses, respectively. Tracking amounts to determining the relation between object models ${}_i$ and observations ${}_j$ over time.

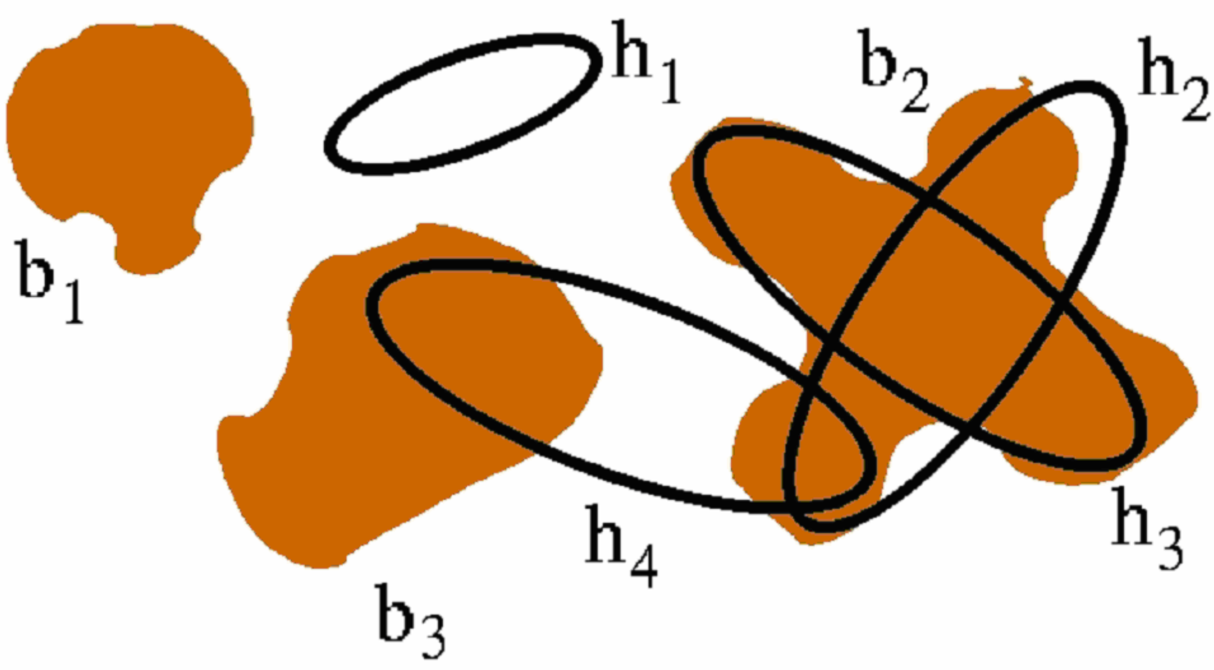


Figure 1. Various possible configurations of skin-colored blobs and object hypotheses. See text for details

Figure 1 exemplifies the problem. In this particular example there are three blobs ($_1$, $_2$ and $_3$) while there are four object hypotheses ($_1$, $_2$, $_3$ and $_4$) carried from the previous frame.
What follows is an algorithm that can cope effectively with the temporal data association problem.

The proposed algorithm needs to address three different sub-problems:

(a) Object hypothesis generation (i.e. an object appears in the field of view for the first time).

(b) Object hypothesis tracking in the presence of multiple, potential occluding objects (i.e. previously detected objects that continue to move arbitrarily in the field of view).

(c) Object model hypothesis removal (i.e. a tracked object disappears from the field of view).

Each of the aforementioned problems is dealt with in the manner explained below.

4.1 Object Hypothesis Generation

We define the distance $D(p, \ )$ of a pixel $p = p(x, y)$ from an ellipse $(c_x, c_y, \alpha, \beta, \theta)$ as follows:

$$D(p, \ ) = \| \vec{v} \|, \tag{3}$$

where $\| \cdot \|$ denotes the ℓ^2 norm of a vector and $\vec{v}$ is defined by

$$\vec{v} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} \dfrac{x - x_c}{\alpha} \\ \dfrac{y - y_c}{\beta} \end{bmatrix}. \tag{4}$$

The distance $D(p, \ )$ is defined so that its value is less than, equal to or greater than 1 depending on whether pixel p is inside, on, or outside ellipse , respectively. Consider now a model ellipse and a pixel p belonging to a blob . In the case where $D(p, \ ) < 1$, we conclude that pixel p and blob support the existence of the object hypothesis and that object hypothesis predicts blob . Consider now a blob such that:

$$\forall p \in \ , \quad \min_{\in} \{D(p, \ )\} > 1. \tag{5}$$

Equation (5) describes a blob whose intersection with all ellipses of the existing object hypotheses is empty. Blob $_1$ in Fig. 1 corresponds to such a case. This implies that none of the existing object hypotheses accounts for the existence of this blob. For each such blob, a new object hypothesis is generated. The parameters of the generated object hypothesis can be derived directly from the statistics of the distribution of pixels belonging to the blob. The center of the ellipse of the object hypothesis becomes equal to the centroid of the blob and the remaining ellipse parameters can be computed from the covariance matrix of the bivariate distribution of the blob pixels location. More specifically, it can be shown that if the covariance matrix Σ corresponding to the distribution of the blob's pixels coordinates

$$\text{is } \Sigma = \begin{bmatrix} \sigma_{xx} & \sigma_{xy} \\ \sigma_{xy} & \sigma_{yy} \end{bmatrix},$$

then an ellipse can be defined with parameters:

$$\alpha = \sqrt{\lambda_1}, \quad \beta = \sqrt{\lambda_2}, \quad \theta = \tan^{-1}\left(\frac{\sigma_{xy}}{\lambda_1 - \sigma_{yy}} \right), \tag{6}$$

where

$$\lambda_1 = \frac{\sigma_{xx} + \sigma_{yy} + \Lambda}{2}, \quad \lambda_2 = \frac{\sigma_{xx} + \sigma_{yy} - \Lambda}{2}, \quad \Lambda = \sqrt{\left(\sigma_{xx} - \sigma_{yy}\right)^2 + 4\sigma_{xy}{}^2} \tag{7}$$

Algorithmically, all blobs that are detected in each frame are tested against the criterion of eq. (5). For all qualifying blobs, a new object hypothesis is formed and the corresponding ellipse parameters are determined based on eqs. (6) and (7). Moreover, all such blobs are excluded from further consideration in the subsequent steps of object tracking.

4.2 Object Hypothesis Tracking

After new object hypotheses have been formed as described in the previous section, all the remaining blobs must support the existence of past object hypotheses. The main task of the tracking algorithm amounts to associating blob pixels to object hypotheses. There are two rules governing this association:

- **Rule 1:** If a skin-colored pixel of a blob is located within the ellipse of a particular object hypothesis (i.e. supports the existence of this hypothesis), then this pixel is considered as belonging to this hypothesis.
- **Rule 2:** If a skin-colored pixel is outside all ellipses corresponding to the object hypotheses, then it is assigned to the object hypothesis that is closest to it in terms of the distance metric of eq. (3).

Formally, the set o of skin-colored pixels that are associated with an object hypothesis is given by the union of two disjoint sets, specifically $o = R_1 \cup R_2$ with

$$R_1 = \{p \in \quad | D(p, \quad) < 1\} \text{ and } R_2 = \left\{ p \in \quad | \quad = \arg\min_{n \in} \{D(p,n) \,|\, D(p,n) \ge 1\} \right\}.$$

In the example of Fig. 1, two different object hypotheses ($_2$ and $_3$) are "competing" for the skin-colored region corresponding to blob $_2$. According to the rule 1 above, all skin pixels within the ellipse of $_2$ will be assigned to it. According to the same rule, the same will happen for skin pixels under the ellipse of $_3$. Note that pixels in the intersection of these ellipses will be assigned to both hypotheses $_2$ and $_3$. According to rule 2, pixels of blob $_2$ that are not within any of the ellipses, will be assigned to their closest ellipse, as this is determined by eq. (3).

Another interesting case is that of a hypothesis that is supported by more than one blobs (such as hypothesis $_4$ in Fig. 1).

Such cases may arise when, for example, two objects are connected at the time they first appear in the scene (e.g. two crossed hands) and later split. To cope with situations where a hypothesis receives support from several blobs, the following strategy is adopted. If there exists only one blob that is predicted by and, at the same time, is not predicted by any other hypothesis, then is assigned to . Otherwise, is assigned to the blob with which it shares the largest number of skin-colored pixels. In the example of Fig. 1, hypothesis $_4$ gets support from blobs $_2$ and $_3$. Based on the above rule, it will be finally assigned to blob $_3$.

After having assigned skin pixels to object hypotheses, the parameters of the object hypotheses $_i$ are re-estimated based on the statistics of the set of pixels o_i that have been assigned to them, according to eq, (6).

4.3 Object Hypothesis Removal

An object hypothesis should be removed either when the corresponding object moves out of the camera's field of view, or when the object is completely occluded by some other non skin-colored object in the scene. Thus, an object hypothesis should be removed from further consideration whenever

$$\forall p \in \quad , \qquad D(p, \quad) > 1. \tag{8}$$

Equation (8) essentially describes hypotheses that are not supported by any skin-colored image pixels. Hypothesis $_1$ in Fig. 1 illustrates such a case. In practice, and in order to account for the case of possibly poor skin-color detection, we allow an object hypothesis to "survive" for a certain amount of time, even in the absence of any support from skin-colored pixels. In our implementation, this time interval has been set to half a second. Thus, a hypothesis will be removed only after fourteen consecutive frames have been elapsed during which it has not received support from any skin-colored pixel. During these frames, the hypothesis parameters do not change but remain equal to the ones computed during the last frame in which it received support from some skin-colored pixels.

4.4 Prediction of Hypotheses Temporal Dynamics

In the processes of object hypothesis generation, tracking and removal that have been described so far, data association is based on object hypotheses that have been formed or updated during the previous time step. Therefore, there is a time lag between the definition of models and the acquisition of data these models are intended to represent. Assuming that the immediate past is a good prediction for the immediate future, a simple linear rule can be used to predict the location of object hypotheses at time t, based on their locations at time $t-2$ and $t-1$.

Therefore, instead of employing ${}_i = {}_i(c_{x_i}, c_{y_i}, \alpha_i, \beta_i, \theta_i)$ as the ellipses describing the object hypothesis i, we actually employ $\hat{}_i = {}_i(\hat{c}_{x_i}, \hat{c}_{y_i}, \alpha_i, \beta_i, \theta_i)$ where

$$\left(\hat{c}_{x_i}(t), \hat{c}_{y_i}(t)\right) = C_i(t-1) + \Delta C_i(t).$$

In the last equation, $C_i(t)$ denotes $\left(c_{x_i}(t), c_{y_i}(t)\right)$ and $\Delta C_i(t) = C_i(t-1) - C_i(t-2)$.

The above definition exploits temporal continuity, i.e. it postulates that an object hypothesis will maintain the same direction and magnitude of translation on the image plane, without changing any of its other parameters. Experimental evaluation has indicated that this simple mechanism for predicting the evolution of hypotheses with time performs surprisingly well even for complex object motions, provided that processing is performed fast enough to keep up with real-time video acquisition.

5. Experiments

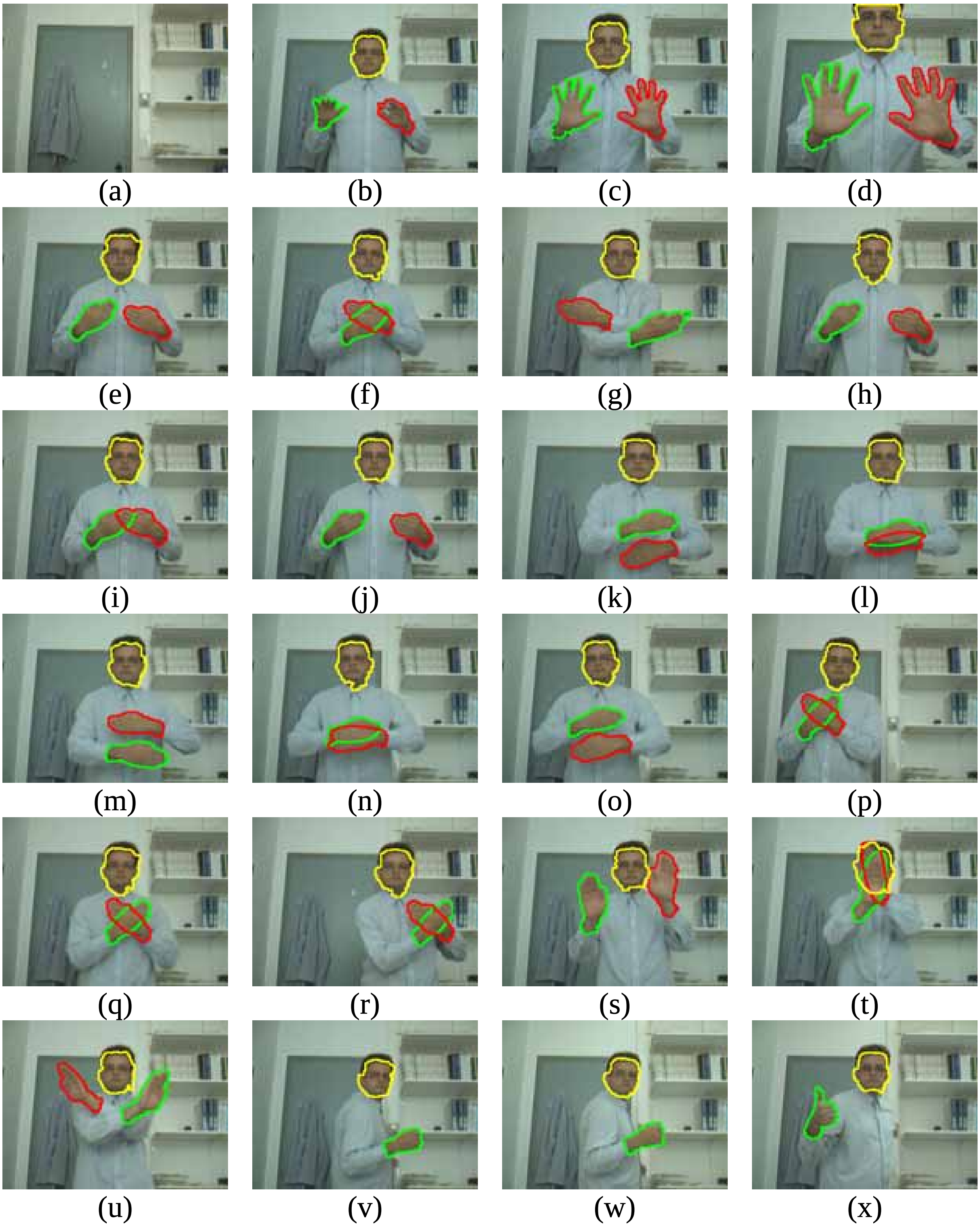
In this section, representative results from an experiment conducted using a prototype implementation of the proposed tracker are provided. The reported experiment is based on a long (3825 frames in total) sequence that has been acquired and processed on-line and in real-time on a Pentium IV laptop computer running MS Windows at 2.56 GHz. A web camera with an IEEE 1394 (Firewire) interface has been used for video capture. In this experiment, the initial, "seed" training set consisted of 20 images and was later refined in a semi-automatic manner using 80 additional images. The training set contains images of four different persons that have been acquired under various lighting conditions.

Figure 2 provides a few characteristic snapshots of the experiment. For visualization purposes, the contour of each tracked object hypothesis is shown. Different contour colors correspond to different object hypotheses.

When the experiment starts, the camera is still and the tracker correctly asserts that there are no skin-colored objects in the scene (Fig. 2(a)). Later, the hand of a person enters the field of view of the camera and starts moving at various depths, directions and speeds in front of it. At some point in time, the camera also starts moving in a very jerky way; the camera is mounted on the laptop's monitor, which is being moved back and forth. The person's second hand enters the field of view; hands now move in overlapping trajectories. Then, the person's face enters the field of view.

Hands disappear and then reappear in the scene. All three objects move independently in disjoint trajectories and in varying speeds ((b)-(d)), ranging from slow to fast; at a later point in time, the person starts dancing, jumping and moving his hands very fast. The experiment proceeds with hands moving in crossing trajectories. Initially hands cross each other slowly and then very fast ((e)-(g)).

Later on, the person starts applauding which results in his hands touching but not crossing each other ((h)-(j)). Right after, the person starts crossing his hands like tying in knots ((k)-(o)). Next, the hands cross each other and stay like this for a considerable amount of time; then the person starts moving, still keeping his hands crossed ((p)-(r)). Then, the person waves and crosses his hands in front of his face ((s)-(u)). The experiment concludes with the person turning the light on and off ((v)-(x)), while greeting towards the camera (Fig. 2(x)).



(a) (b) (c) (d)

(e) (f) (g) (h)

(i) (j) (k) (l)

(m) (n) (o) (p)

(q) (r) (s) (t)

(u) (v) (w) (x)

Figure 2. Characteristic snapshots from the on-line tracking experiment

As it can be verified from the snapshots, the labeling of the object hypotheses is consistent throughout the whole sequence, which indicates that they are correctly tracked. Indeed, the proposed tracker performs very well in all the above cases, some of which are challenging. It should also be mentioned that no images of the person depicted in this experiment were contained in the training set.

With respect to computational performance, the 3825 frames sequence presented previously has been acquired and processed at an average frame rate of 28.45 fps with each frame being of dimensions 320x240. It is stressed that the reported frame rate is determined by the maximum acquisition frame rate supported by the employed camera, since the acquisition delay for a single frame dominates the tracker's cycle time. When employing prerecorded image sequences that are loaded from disk, considerably higher tracking frame rates can be achieved. Performance can be further improved by running the tracker on lower resolution images that result from subsampling original images by a factor of two.

Apart from the reported example, the proposed tracker has also been extensively tested with different cameras and in different settings involving different background scenes and human subjects.

Demonstration videos including the reported experiment can be found online at http://www.ics.forth.gr/~argyros/research/colortracking.htm.

6. Conclusion

In this paper, a method for tracking multiple skin-colored objects has been presented. The proposed method can cope successfully with multiple objects moving in complex patterns as they dynamically enter and exit the field of view of a camera. Since the tracker is not based on explicit background modeling and subtraction, it may operate even on image sequences acquired by a moving camera.

Moreover, the color modeling and detection modules facilitate robust performance in the case of varying illumination conditions. Owing to the fact that the proposed approach treats the problem of tracking under very loose assumptions and in a computationally efficient manner, it can serve as a building block of larger vision systems employed in diverse application areas.

Further research efforts have focused on (1) combining the proposed method with binocular stereo processing in order to derive 3D information regarding the tracked objects, (2) providing means for discriminating various types of skin-colored areas (e.g. hands, faces, etc), (3) developing methods that build upon the proposed tracker in order to be able to track interesting parts of skin-colored areas (e.g. eyes for faces, fingertips for hands, etc) and (4) employing the proposed tracker for supporting human gesture interpretation in the context of applications such as effective human computer interaction.

Acknowledgements

This work was partially supported by EU IST-2001-32184 project ActIPret.

7. References

Argyros, A. & Lourakis, M. (2004). Real-time tracking of multiple skin-colored objects with a possibly moving camera. In *Proc. of ECCV'04*, vol. 3, pages 368-379.

Arulampalam, M.S. & Maskell, S. & Gordon, N. & Clapp, T. (2002). A tutorial on particle filters for on-line non-linear/non-Gaussian Bayesian tracking, *IEEE Trans. on Signal Proc.*, 50(2):174-188.

Canny, J.F. (1986). A computational approach to edge detection. *IEEE Trans. on Pat. Anal. and Mach. Intel.*, 8(11):769–798.

Chai, D. & Ngan, K.N. (1998). Locating facial region of a head-and-shoulders color image. In *Proc. of FG'98*, pages 124–129.

DeCarlo, D. & Metaxas, D. (2000). Optical flow constraints on deformable models with applications to face tracking. *International Journal of Computer Vision*, 38(2):99-127.

Delamarre, Q. & Faugeras, O. (2001). 3D articulated models and multi-view tracking with physical forces. *Computer Vision and Image Understanding*, 81:328–357.

Dempster, A.P & Laird, N.M. & Rubin, D.B. (1977). Maximum-likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1-38.

Gavrila, D.M. (1999). The visual analysis of human movement: A survey. *Computer Vision and Image Understanding*, 73(1):82–98.

Hue, C. & Le Cadre, J.-P. & Perez, P. (2002). Sequential monte carlo methods for multiple target tracking and data fusion. *IEEE Trans. on Signal Proc.*, 50(2):309–325.

Isard, M. & Blake, A. (1998). Icondensation: Unifying low-level and high-level tracking in a stochastic framework. In *Proc. of ECCV'98*, pages 893–908.

Jack, K. (2004). Video demystified. Elsevier science, 4th edition.

Javed, O. & Shah, M. (2002). Tracking and object classification for automated surveillance. In *Proc. of ECCV'02*, pages 343–357.

Jebara, T.S. & Pentland, A. (1997). Parameterized structure from motion for 3d adaptive feedback tracking of faces. In *Proc. of CVPR'97*, pages 144–150.

Jebara, T.S. & Russel, K. & Pentland, A. (1998). Mixture of eigenfeatures for real-time structure from texture. In *Proc. of ICCV'98*, pages 128–135.

Jones, M.J. & Rehg, J.M. (1999). Statistical color models with application to skin detection. In *Proc. of CVPR'99*, volume 1, pages 274–280.

Kalman, R.E. (1960). A new approach to linear filtering and prediction problems. *Transactions of the ACME-Journal of Basic Engineering*, pages 35–45.

Kim, S.H. & Kim, N.K. & Ahn, S.C. & Kim, H.G. (1998). Object oriented face detection using range and color information. In *Proc. of FG'98*, pages 76–81.

Koller-Meier, E. & Ade, F. (2001). Tracking multiple objects using the condensation algorithm. *Journal of Robotics and Autonomous Systems*, 34(2-3):93–105.

Plänkers, R. & Fua, P. (2001). Tracking and modeling people in video sequences. *Computer Vision and Image Understanding*, 81(3): 285-302.

Raja, Y. & McKenna, S. & Gong, S. (1999). Tracking color objects using adaptive mixture models. *Image and Vision Computinl*, 17(3-4):225–231.

Saxe, D. & Foulds, R. (1996). Toward robust skin identification in video images. In *Proc. of FG'96*, pages 379–384.

Siebel, N.T. & Maybank, S. (2002). Fusion of multiple tracking algorithms for robust people tracking. In *Proc. of ECCV'02*, pages 373–387.

Spengler, M. & Schiele, B. (2003). Multi object tracking based on a modular knowledge hierarchy. In *Proc. of International Conference on Computer Vision Systems*, pages 373–387.

Terrillon, J.C. & Shirazi, M.N. & Fukamachi, H. & Akamatsu, S. (2000). Comparative performance of different skin chrominance models and chrominance spaces for the automatic detection of human faces in color images. In *Proc. of FG'00*, pages 54–61.

Triesch, J. & von der Malsburg, C. (2001). Democratic integration: Self-organized integration of adaptive cues. *Neural Computation*, 13(9):2049–2074.

Vezhnevets, V. & Sazonov, V. & Andreeva, A. (2003). A survey on pixel-based skin color detection techniques. In *Proc of Graphicon'03*, pages 85-92.

Yang, M.H. & Ahuja, N. (1998). Detecting human faces in color images. In *Proc. of ICIP'98*, volume 1, pages 127–130.

Yang, M.H. & Ahuja, N. (2001). *Face detection and gesture recognition for human computer interaction*. Kluwer Academic Publishers, New York.

Yang, M.H. & Kriegman, D.J. & Ahuja, N. (2002). Detecting faces in images: A survey. IEEE Trans. on Pat. Anal. and Mach. Intel., 24(1):34–58.

Feature Extraction and Grouping for Robot Vision Tasks

Miguel Cazorla & Francisco Escolano

1. Introduction

This paper focuses on feature extraction and perceptual grouping in computer vision. Our main purpose has been to formulate and test new methods for feature extraction (mainly those related to corner identification and junction classification) and grouping (through junction connection). Our purpose is to build a geometric sketch which can be useful for a wide range of visual tasks. The context of application of these methods, robot vision, imposes special real-time constraints over their practical

use, and thus, the following requirements are observed: efficiency, robustness, and flexibility.

The paper is structured in three parts which cover junction classification, grouping, and estimation of the relative orientation of the robot with respect to the environment (an example of visual task). We follow a bottom-up exposition of the topics (Cazorla, 2000):

- Junction Classification: It relies on combining corner detectors and template matching. Corner detection is performed through two well known operators: SUSAN and Nitzberg. These operators provide an initial localization of the junction center. Given this localization, junction classification is posed in terms of finding the set of angular sections, or wedges, that explain as better as possible the underlying evidence in the image. We propose two greedy methods which rely on elements of Bayesian inference. These methods are tested with indoor and outdoor images in order to identify their robustness to noise and also to bad localizations of junction centers. Experimental results show that these methods are saver than other methods recently proposed, like Kona, and their computational efficiency is even better. However, we have detected some problems due to the local extent of junction detection. These problems and those derived from bad center localization can be aliviated through local-to-global interactions, and these interactions are inferred by grouping processes.
- Grouping: Using classified junctions as starting elements, this stage is performed by finding connecting paths between wedge limits belonging to pairs of junctions, and these paths exist when there is sufficient contrast or edge support below them. Given that corners are usually associated to points of high curvature in the image, it can be assumed that connecting paths must be smooth if they exist. Contrast support and smoothness are quantified by a cost function. As it can be assumed that there will be no more than one path between two junctions through a pair of wedge limits, such a path can be found by the Bayesian version of the well known A* algorithm. This method recently proposed searches the true path, according to the cost function, in a population of false path, instead of the best path among a population of possible paths.

Such a simplification yields a computational average cost which is linear with the length of the path. We propose two versions of the cost functions and explore the possibilities of the method for the task at hand by selecting adequate thresholds which control the penalization introduced by the lack of support or smoothness. Penalized partial paths can be discarded by a pruning rule. We also modify this rule to provide stronger pruning although the admissibility of the algorithm is not guaranteed in this case. Grouping is completed by end-path conditions, and the robustness of the process is ensured by bidirectional search. Experimental results show that the grouping stage improves individual junction identification, and in this sense grouping gives feedback to a low level task. But grouping also provides a schematic representation which is useful to higher-level tasks like computing relative orientation from a single image.

- Relative orientation: Grouping results followed by straight line detection are very useful to compute the relative orientation between the observer and the environment. A recent proposal, which also relies on Bayesian inference, has proved that this task can be performed by using a single image, at least when the horizontal viewing direction is assumed. This method exploits perspective information and assumes that lines follow a particular configuration known as Manhattan world. We have replaced the original pixel-based strategy by the edges resulting from grouping in the preceding stage, yielding robust and fast results.

A graphical example of the complete process is shown in Fig. 1.

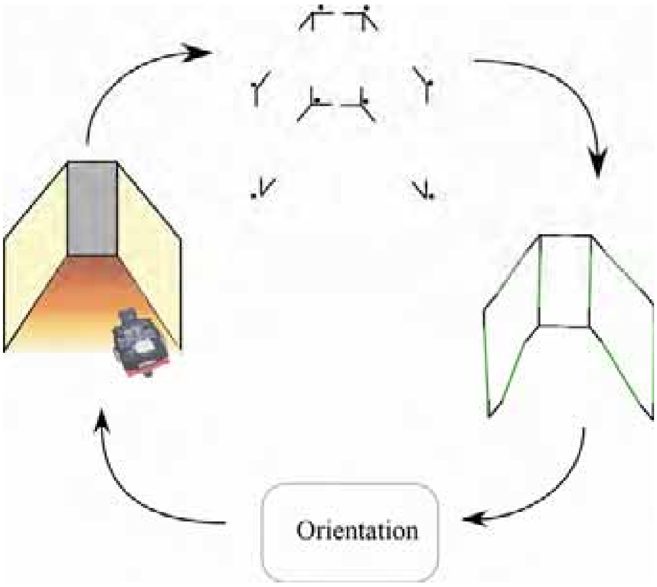


Figure 1. Sequence in the process: first, the corners and junctions are calculated; then the grouping step is applied; the last step is the orientation (compass) estimation, which is used to guide the robot

2. Juction Classification

A generic junction model can be encoded by a parametric template

$$\Theta = \left(x_c, y_c, r, M, \{\phi_i\}, \{\ _i\}\right) \tag{1}$$

where (x_c, y_c) is the center, r is the radius, M is the number of wedges (sectors of near constant intensity), $\{\phi_i\}$ with $i = 1, \ldots,$ are the wedge limits (supposed to be placed on the edge segments convergent in the center) and $\{\ _i\}$, the intensity distributions associated to the wedges (see Fig. 2).

Some practical assumptions will reduce the number of parameters to be estimated by a junction detector. First, potential junction centers (x_c, y_c) may be localized by a local filter, like the Plessey detector (Harris & Stephens, 1988) or the more recently the SUSAN detector (Smith & Brady, 1997) which declares a corner when the intensity of the center is similar to that of a small fraction of points in the neighborhood. Second, although the optimal radius r may be found (Parida, Geiger & Hummel, 1998) the cost of doing it is prohibitive for real-time purposes. This is why this parameter is assumed to be set by the user. Furthermore, in order to avoid distortions near the junction center, a small domain with radius $R_{\min}$ around it should be discarded and then $r = R_{\max} - R_{\min}$ where $R_{\max}$ is the scope of the junction.

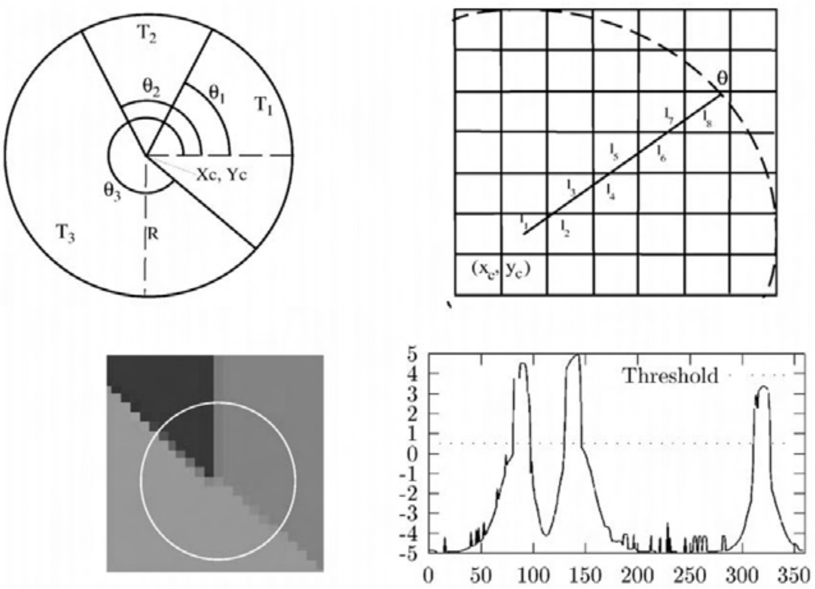


Figure 2. (Top-left) Parametric model for junctions; (top-right) discrete accumulation of intensity along a direction; (bottom-left) example of an ideal junction; (bottom-right) intensity profile of the junction where each peak represents the location of a wedge limit

Given the latter simplifications, a junction classification method will focus on finding the optimal number of wedges M , the wedge limits $\{\phi_i\}$ and the wedge intensity distributions $\{\ _i\}$. To that purpose one may follow either a region-based or an edge-based approach, exploiting both of them the fact that wedge intensity distributions are assumed to be near constant (that is, junctions are built on piecewise smooth areas in the image). In the region-based approach the optimal position of a wedge limit is the equilibrium point between the neighbouring wedge regions which can be in turn fused into a greater one. However, in the edge-based method, edges emanating from the junction center are detected and thresholded. We have proposed these methods in (Cazorla & Escolano, 2003) and we have found that the second one yields a lower failure rate although both of them are prone to over-segmentation. This is why we describe here the edge-based method. Consequently, finding the wedge limits is addressed by analyzing the one-dimensional contrast profile (see Fig. 2, bottom right) associated to the junction. Such a profile is estimated by computing, for each angle $\phi \in [0, 2\pi]$ the averaged accumulated contrast $\tilde{I}_\phi$ along the radius in such direction

$$\tilde{I}_\phi = \frac{1}{r}\sum_{i=1} l_i \times \mathrm{E}_{\mathrm{u}} \qquad (2)$$

where E_{u} is the intensity contrast of the pixel $\mathrm{u}=(u,v)$ associated to segment l_i as it is shown in Fig. 2 (top right), being the number of segments needed to discretize the radius along a given direction and not necessarily the same for all directions. As the reliability of the junction detector depends on how "edgeness" is defined. As starting point we choose $\mathrm{E}_{\mathrm{u}}=|\ \ *\nabla I(\mathrm{u})|$, that is, the Gaussian smoothed gradient (with typically unitary standard deviation). The robustness of such a measure can be improved by embodying it into a decision test which performs a good classification of both edge and non-edge pixels. Recent studies in edge modeling applied to road tracking tasks (Geman & Jedynak, 1996), (Coughlan & Yuille, 1999), point towards building such decision test on the log-likelihood ratio, that is, the logarithm of the ratio between the probability of being "on" and "off" an edge. This criterion guarantees an optimal decision in the sense that it minimizes the Bayesian classification error, but the underlying distributions for "on" and "off" must be known beforehand. Such distributions can be estimated empirically by gathering and quantizing the frequencies of the filter responses in both cases. Then, the empirical probability that a given response is associated to an edge pixel is denoted by $_{on}(\mathrm{E}_{\mathrm{u}})$ and the empirical probability that a given response corresponds to a non-edge pixel is denoted by $_{o}\ (\mathrm{E}_{\mathrm{u}})$. In this paper we will use the empirical distributions presented in (Yuille & Coughlan, 2000). These distributions, were extracted from a range of images and quantized to take 20 values. Both distributions, and the corresponding plots of log-likelihood ratios, are shown in Fig. 4. Taking into account the log-likelihood ratio equation 2 is rewritten as

$$\tilde{I}_{\phi}=\frac{1}{r}\sum_{i=1} l_i \times \log \frac{_{on}(\mathrm{E}_{\mathrm{u}}\,|\,\phi^{*})}{_{o}\ (\mathrm{E}_{\mathrm{u}})} \qquad (3)$$

Given an input image, as shown in Fig. 3, the log-likelihood ratio between both the "on" and "off" probabilities gives a robust identification of edge pixels. However, the latter probabilistic model of the smoothed gradient filter can be easily improved by incorporating the orientation of such gradient in the definition of both the "on" and "off" probabilities. Consequently, the smoothed gradient at a given point is defined by $\mathrm{E}_{\mathrm{u}}=(\ _{\mathrm{u}},\sigma_{\mathrm{u}})$, where σ_{u} is the local estimation of σ^{*} the true orientation of the edge to which the pixel belongs. In (Coughlan & Yuille, 1999), where the local estimations of the gradient are used to accumulate evidence through a Hough-like process addressed to estimate the vanishing points in an image, the "on" probability is defined in terms of $_{on}(\mathrm{E}_{\mathrm{u}}\,|\,\sigma^{*})$, the conditional probability of a gradient vector given the true orientation, that is, as a function of the true orientation. Such definition makes sense because the probability of an edge being on must decrease as the estimated orientation diverges from the true orientation, and conversely it must increase as both orientations converge. Furthermore, it can be assumed that the magnitude of the gradient is independent from its orientation and viceversa, which leads to the factorization of the "on" probability into two terms, one of them depending on the gradient vector and the other one on the divergence between the real and the estimated orientations:

$$_{on}(\mathrm{E}_{\mathrm{u}}\,|\,\sigma^{*})=\ _{on}(\ _{\mathrm{u}})\ _{ang}(\sigma_{\mathrm{u}}-\sigma^{*}) \qquad (4)$$

where $_{ang}(\sigma_{\mathrm{u}}-\sigma^{*})$ is the probability of having the correct orientation. Although this probability can be estimated empirically its shape is consistent with having a maximum both in 0 (when both orientations coincide) and π (when both orientations are opposite). In this paper we have used this simple definition as shown in Fig. 3 (bottom right).

On the other hand, "off" probability can be redefined without considering the dependence between the estimated orientation and the true orientation. Therefore such probability $_{o}(\mathrm{E_u})$ does only depends on the gradient vector. Also assuming independence between gradient magnitude and orientation the resulting probability depends also on two terms:

$$_{o}(\mathrm{E_u}) = {}_{o}(\ {}_{\mathrm{u}})\ (\sigma_{\mathrm{u}}) \tag{5}$$

where $(\sigma_{\mathrm{u}}) = 1/2\pi$ is the uniform distribution. The effectiveness of this model for estimating "edgeness" is shown in Fig. 3 where we represent the log-likelihood ratio and the magnitude and orientation of the gradient.In Fig. 3 we represent the log-likelihood ratio and the magnitude and orientation of the gradient.

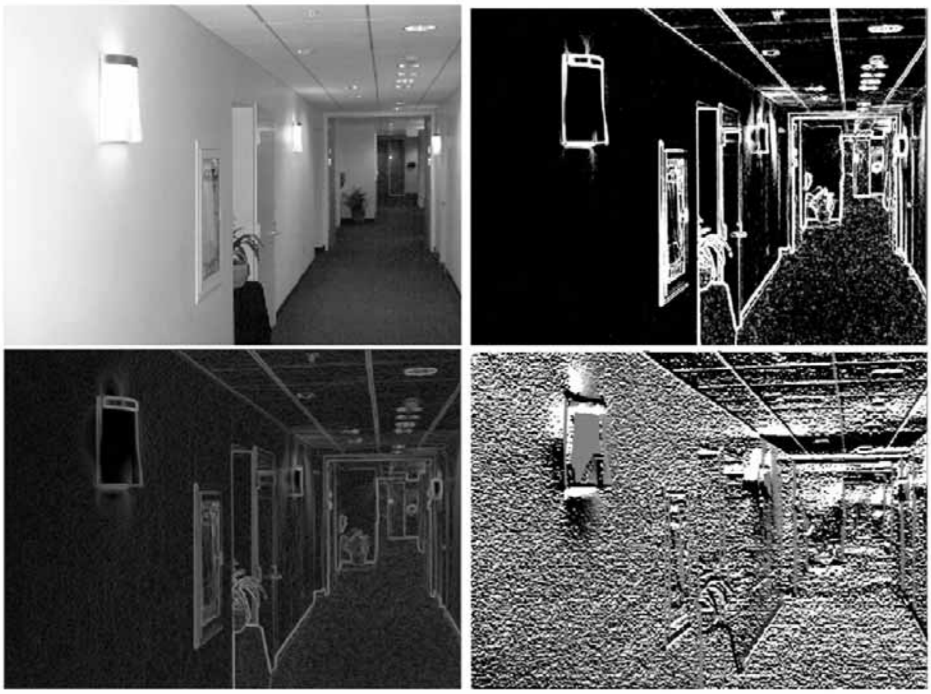


Figure 3. (Top) Sample image and the value of the log-likelihood ratio for pixels; (bottom) magnitude (left) and orientation (right) of the gradient. In the case of orientation, grey is 0, white π and black is $-\pi$

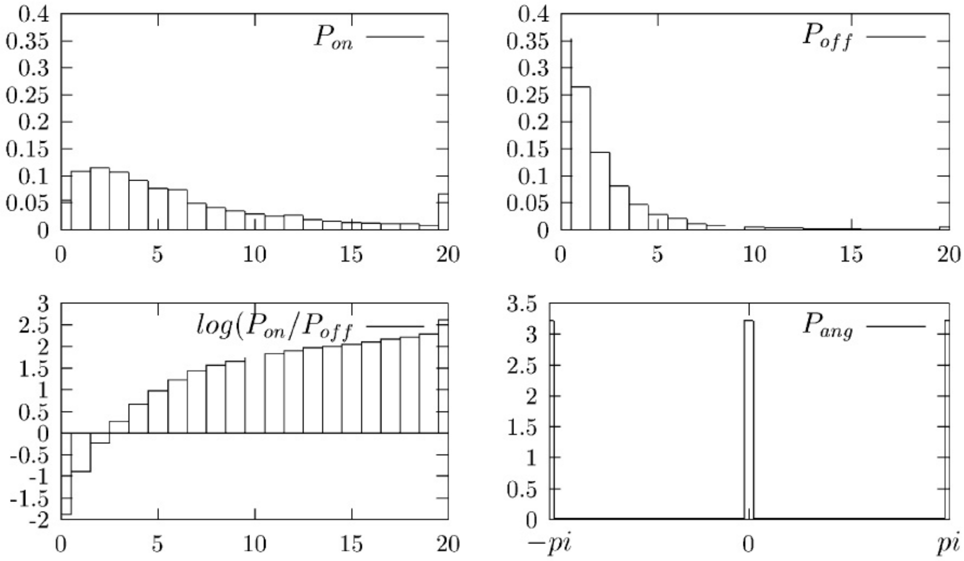


Figure 4. (Top) Plot of the $_{on}/_{o}$ information. (Bottom) $\log\ _{on}/_{o}$ and angle distribution

Given the latter profile, junction classification simply consists of thresholding it properly. Furthermore, as the quality of the contrast profile depends on the correct localization of the junction center, we compensate for small localization errors (2-3 pixels) by replacing the average $\tilde{I}_{\phi}$ by the median $\hat{I}_{\phi}$. In Fig. 5 (top) we represent the resulting contrast profile. In practice, after thresholding we filter junctions with less than two wedges or with two wedges defining a quasi-straight line. We show some results in Fig. 5 (bottom). The parameters typically used were: $R_{\min} = 4, R_{\max} = 10$ that is $r = 6$, being the threshold $= 0.5$. Due to incorrect junction scopes or to bad localizations, the latter method may yield either false positives or false negatives (see Fig. 5). However, some of these errors may be corrected by a proper grouping strategy along potential connecting edges between wedges of different junctions.

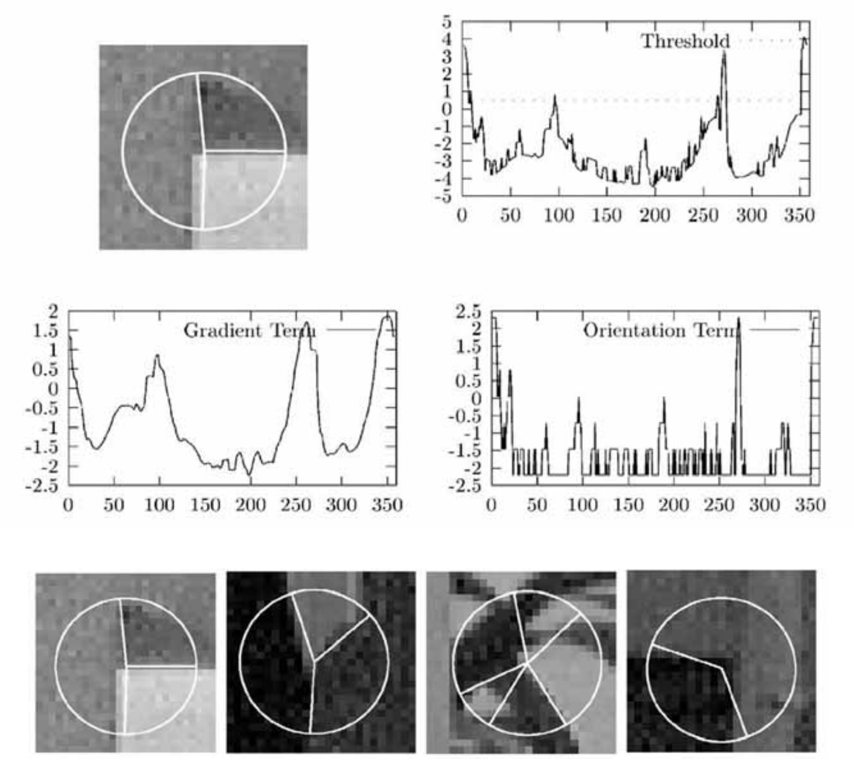


Figure 5. (Top and middle) Contrast profile using the Bayesian edge model. Peaks above the threshold represent suitable wedge limits; (bottom) some results

3. Connecting and Filtering Junctions

The grouping strategy relies on finding "connecting paths". A connecting path of length L rooted on a junction center (x_c, y_c) and starting from the wedge limit defined by ϕ is defined by a sequence of connected segments $p_1, p_2, \ldots, p_L$ with fixed or variable length. As points of high curvature are usually associated to corners and these corners are placed at the origin or at the end of the paths, we assume that the curvature of these paths is smooth. For describing curvature we define second-order orientation variables $\alpha_1, \alpha_2, \ldots, \alpha_{L-1}$ where $\alpha_j = \phi_{j+1} - \phi_j$ is the angle between segments p_{j+1} and p_j. Then, following (Yuille & Coughlan, 2000) a connecting path * should maximize

$$\left(\{p_j, \alpha_j\}\right) = \sum_{j=1}^{L} \log\left\{\frac{{}_{on}(p_j)}{{}_{o}\ (p_j)}\right\} + \sum_{j=1}^{L-1} \log\left\{\frac{{}_{\Delta}\ (\alpha_{j+1} - \alpha_j)}{(\alpha_{j+1} - \alpha_j)}\right\} \quad (6)$$

It is straightforward to transform the latter cost function into the typical cost function for snakes if we assume a Gibbs function whose exponent is given by Equation (5). Actually the first term is related to the external energy whereas the second one is related to the internal energy (curvature). More precisely, the first term is the "intensity reward" and it depends on the edge strength along each segment p_j. Defining the intensity reward of each segment of fixed length F in terms of the same edge model that was used to compute the contrast profile yields

$$\log\left\{\frac{{}_{on}(p_j)}{{}_{o}\ (p_j)}\right\}=\frac{1}{F}\sum_{i=1} l_i \times \log\frac{{}_{on}(\mathrm{E_u}\,|\,\phi^*)}{{}_{o}\ (\mathrm{E_u})} \tag{7}$$

Alternatively, ${}_{on}(p_j)$ and ${}_{o}\ (p_j)$ may be built on a non-linear filter depending on the gradient magnitude and also on the relative orientation of the segment with respect to the underlying edge.

On the other hand, the second term is the "geometric" reward", which relies on a first-order Markov chain on orientation variables which implements geometric smoothness by exponentially penalizing angular variations

$$(\alpha_{j+1}\,|\,\alpha_j)= {}_{\Delta}\ (\alpha_{j+1}-\alpha_j) \tag{8}$$

and

$${}_{\Delta}\ (\Delta\alpha_j)\propto \exp\left\{-\frac{C}{2}\left|\Delta\alpha_j\right|\right\} \tag{9}$$

where $\Delta\alpha_j=\alpha_{j+1}-\alpha_j$, C modulates the rigidity of the path, and $(\alpha_{j+1}-\alpha_j)$ is the uniform distribution of the angular variation, defined to keep both the intensity and geometric terms in the same range. Such a choice of the geometric reward is dictated by our smoothness assumption but it is desirable to learn it from the data of the application domain. As we will see later, the effectiveness of this reward depends on its departure from the uniform distribution.

Once the cost function is defined in Equations (6-9) its maximization is addressed by the Bayesian A* algorithm (Coughlan & Yuille, 1999). Given an initial junction center (x_c^0, y_c^0) and an orientation ϕ^0 the algorithm explores a tree in which each segment p_j may expand successors. Although there are paths for path lengths of $=L$, the Bayesian A* exploits the fact that we want to detect one target path against clutter, instead of taking the best choice from a population of paths. Then, the complexity of the search may be reduced by pruning partial paths with "too low" rewards. Then, the key element of Bayesian A* is the pruning rule. The algorithm finds the best path surviving to the pruning with an expected convergence rate of $(\ )$. In order to do so the pruning relies on evaluating the averaged intensity and geometric rewards of the last L_0 segments of a path. These segments are called the "segment block" and the algorithm discards them when their averaged intensity or geometric reward is below a given threshold:

$$\frac{1}{L_0}\sum_{j=zL_0}^{(z+1)L_0-1}\log\left\{\frac{{}_{on}(p_j)}{{}_{o}\ (p_j)}\right\}<T_I \tag{10}$$

$$\frac{1}{L_0}\sum_{j=zL_0}^{(z+1)L_0-1}\log\left\{\frac{{}_{\Delta}\ (\Delta\alpha_j)}{(\Delta\alpha_j)}\right\}<T \tag{11}$$

being T_I and T are respectively the intensity and geometric thresholds. These thresholds determine the "minimum averaged reward" required for survive the pruning. What is key in this context is the thresholds are not arbitrary and they must satisfy the following conditions

$$-D({}_{o} \| {}_{on}) < T_I < D({}_{on} \| {}_{o}) \tag{12}$$

and

$$-D({}_{\Delta} \| {}_{\Delta}) < T \ < D({}_{\Delta} \| {}_{\Delta}) \tag{13}$$

being $D(.\|.)$ a distance between two distributions, so called Kullback-Leibler divergence, defined as

$$D({}_{1} \| {}_{2}) = \sum_{i=1}^{M} {}_{1}({}_{i}) \log \frac{{}_{1}({}_{i})}{{}_{2}({}_{i})} \tag{14}$$

The latter thresholds are typically chosen as close to their upper bounds as possible. The rationale of the latter conditions is summarized as follows: If ${}_{on}$ diverges from ${}_{o}$ the then T_I may have a high value and then the pruning cuts many partial paths; otherwise, the pruning will be very conservative. The same reasoning follows for ${}_{\Delta}$ and ${}_{\Delta}$. More precisely, $D({}_{on} \| {}_{o})$ quantifies the quality of the edge detector. Then, having large divergences means that the log-likelihood ratio will decide easily between being "on" or "off" the edge, and we will discover false paths soon. Otherwise, it is hard to know whether a given segment is a good choice or not and the algorithm will wait more before discarding a partial path. Similarly, $D({}_{\Delta} \| {}_{\Delta})$ measures the departure from "geometric ignorance" (dictated by the uniform distribution). The more geometric knowledge the better it helps to constrain the search.

The latter rationale on divergences is independent on the algorithm because there are fundamental limits for the task of discriminating the true task among clutter. There is an order parameter whose value determines whether the task may be accomplished (when it is positive) or not:

$$= 2 \ ({}_{on}, {}_{o}) + 2 \ ({}_{\Delta} , {}_{\Delta}) - \log \tag{15}$$

being $(.,.)$ the Battacharyya bound between two distributions:

$$({}_{1}, {}_{2}) = -\log \left\{ \sum_{i=1}^{M} {}_{1}^{1/2}({}_{i}) \ {}_{2}^{1/2}({}_{i}) \right\} \tag{16}$$

The order parameter depends on the quality of the edge detector and the quality of the geometric knowledge and determines $\tilde{F}$, the expected number of completely false paths having greater reward than the true paths: < 0 implies $\tilde{F} \to \infty$, > 0 implies $\tilde{F} = 0$, and for $= 0$ there is a phase transition.

Although the $(\ )$ convergence is ensured, the existence of real-time constraints motivates the extension of the basic pruning rule introducing an additional rule, though it is not-admissible in terms of optimality: In practice we prime the "stability of long paths". As long paths are more probable than shorter ones to be close to the target, because they have survived to more prunes, we will also prune paths with length L_j when

$$L \ {}_{t} - L_j > Z \times L_0 \text{, that is, } L_j > L \ {}_{t} - Z \times L_0 \tag{17}$$

where $L\ _t$ is the length of the best path so far and $Z \geq 0$ sets the minimum allowed difference between the best path and the rest of path. For low Z we introduce more pruning and consequently increase the risk of loosing the true path. When Z is high, shorter paths may survive. It is desirable to find (experimentally) a value for Z representing a trade-off between admissibility and efficiency.

Then, the Bayesian A* algorithm with the latter pruning expands the search tree until it reaches the center (x_c, y_c) within a given neighbourhood at the end of the selected path. Such search is done by means of a "range tree". The cost of building it is $(\ \log\ )$ being the number of junctions whereas the cost of a query is logarithmic in the worst case.

After reaching a new junction it is checked that the last segment of the path coincides with a wedge limit ϕ and, if so, this limit is labelled as "visited". However, the search may finish without finding a junction (when the search queue is empty) or at a "termination point" whose coordinates must be stored. In the first case, if the length of the path is below the block size L_0 we assume that it corresponds to a false wedge limit. In the second case we assume that we have discovered a potential junction. Then, our "local-to-global" grouping algorithm performs path searching from each non-visited limit and an edge may be tracked in both directions.

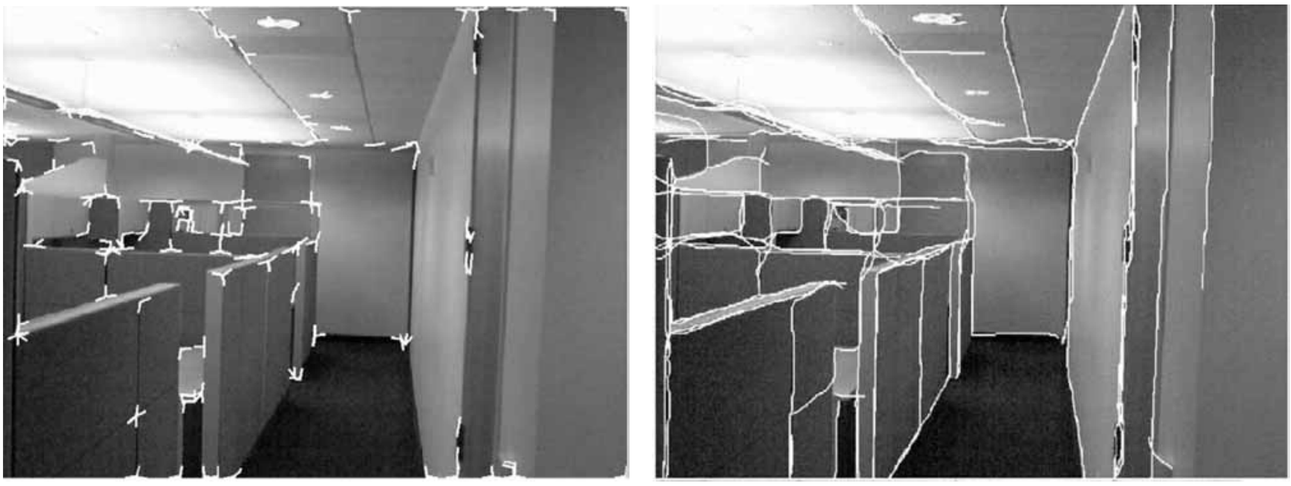


Figure 6. Junction detection results (Top) and grouping results (Bottom)

4. Obtaining the Compass Angle

Here we explain a method for obtaining the compass angle of a camera, using the features obtained previously. Structured (man-made) environments (like a building, roads, corridors, doors, and so on) share some common features: they are usually built using straight forms. It seems convient to exploit this fact to make inferences about quantitative properties like the relative position of the observer.

4.1 3D Geometry

We define now some geometrical concepts used later on. Let Ψ be the camera orientation angle in the Manhattan world (Faugeras, 1993): the camera is oriented in the direction $\cos\Psi \mathrm{i} - \sin\Psi \mathrm{j}$ (see Fig. 7). Coordinates in the image plane $\mathrm{u} = (u, v)$ are related with world coordinates (x, y, z) by

$$u = \frac{\cdot(-x\sin\Psi - y\cos\Psi)}{x\cos\Psi - y\sin\Psi} \quad v = \frac{\cdot z}{x\cos\Psi - y\sin\Psi} \tag{18}$$

where *f* is the camera focal length.

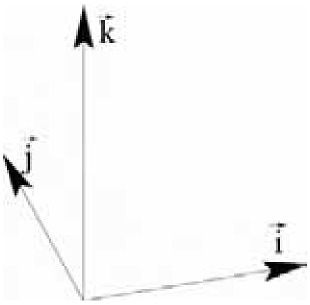


Figure 7. Axis labeling in the Manhattan world

The vanishing points in the directions **i** and **j** are found at the points (-ftanΨ,0) and $(\text{-f}\cdot\tan\Psi,0)$, respectively, in the image plane. Lines in the **k** direction are verticals due to the assumption of being in a structured world.

Finally, we need to relate the intensity gradient in a pixel with the orientation angle. An image point **u**=(*u,v*) with intensity gradient (cosθ,sinθ) is consistent with a line **i**, i.e. it points to the vanishing point, if -vtanθ=u+ftanψ . This equation does not change if we add $\pm\pi$ to θ. In a similar way we have vtanθ=-u+fcotΨ for **j** lines.

4.2 Initial Bayesian Model

Let we describe the Bayesian model proposed in (Coughlan, 99). The main difference from this model with respect to other methods is that this does not need to decide wich pixel is on or off an edge. Furthermore, it allows labelling a pixel as an edge pixel of one of the three Manhattan types: i, j, k .

Using the previous model defined in Section 2, let E_u be the gradient (magnitude and orientation) in an image point u. This value can be explained with one of the following models m_u: $m_\text{u} = 1,2,3$ for each of the edges generated by i, j, k, respectively; $m_\text{u} = 4$ means that the gradient has been generated by an edge in a random direction (not i, j, k); and, finally, $m_\text{u} = 5$ means the pixel is not in an edge (off the edge). We have set the a priori probability $\;(m_\text{u})$ for each model experimentally.

We assume that the gradient probability of the image E_u has two factors corresponding to the magnitude ($_\text{u}$) and orientation (σ_u):

$$(\mathbf{E}_\text{u} \,|\, m_\text{u}, \Psi, \text{u}) = \;(E_\text{u} \,|\, m_\text{u}) \;(\sigma_\text{u} \,|\, m_\text{u}, \Psi, \text{u}) \tag{19}$$

where $\;(E_\text{u} \,|\, m_\text{u})$ is $_o(\;_\text{u})$ if $m_\text{u} = 5$ or $_{on}(\;_\text{u})$ if $m_\text{u} \neq 5$, and $\;(\sigma_\text{u} \,|\, m_\text{u}, \Psi, \text{u})$ is $_{ang}(\sigma_\text{u} - \theta(m_\text{u}, \Psi, \text{u}))$ if $m_\text{u} = 1,2,3$ or $\;(\sigma_\text{u})$ if $m_\text{u} = 4,5$. The orientation term $\theta(m_\text{u}, \Psi, \text{u})$ is the normal determined by the equation -vtanθ=u+ftanΨ for i lines, vtanθ=-u+fcotΨ for j lines and $\theta = 0$ for k lines. Ψ is the camera orientation angle.

Instead of determining which model better describe the pixel, the evidence is accumulated for the five models:

$$(\mathbf{E}_\text{u} \,|\, \Psi, \text{u}) = \sum_{m_\text{u}=1}^{5} \;(\mathbf{E}_\text{u} \,|\, m_\text{u}, \Psi, \text{u}) \;(m_\text{u}) \tag{20}$$

Thus, we can determine the evidence for the camera angle Ψ without knowing to which of those five models owns the pixel.

Now, we want to calculate the evidence for the complete image $\{E_u\}$. We assume that the data is condicionally independent in all the pixels, given an orientation Ψ:

$$(\{\mathbf{E}_u\}\,|\,\Psi) = \prod_u \ (\mathbf{E}_u\,|\,\Psi, u) \tag{21}$$

Thus, the a posteriori distribution over the orientation is

$$\prod_u \ (\mathbf{E}_u\,|\,\Psi, u) \ \ (\Psi)/Z \tag{22}$$

where Z is a normalization factor and $P(\Psi)$ is the uniform a priori distribution over Ψ. In order to find the Maximum A Posteriori (MAP), we need to maximize (we ignore Z, as it is independent from Ψ):

$$\Psi^* = \arg\max_{\Psi} \ \ (\{\mathbf{E}_u\}\,|\,\Psi) \ \ (\Psi) \tag{23}$$

Applying log to the previous equation:

$$\log\left[\arg\max_{\Psi} \ \ (\{\mathbf{E}_u\}\,|\,\Psi) \ \ (\Psi)\right] = \log \ \ (\Psi) + \sum_{m_u=1}^{5} \log\left[\arg\max_{\Psi} \ \ (\mathbf{E}_u\,|\,m_u, \Psi, u) \ \ (m_u)\right] \tag{24}$$

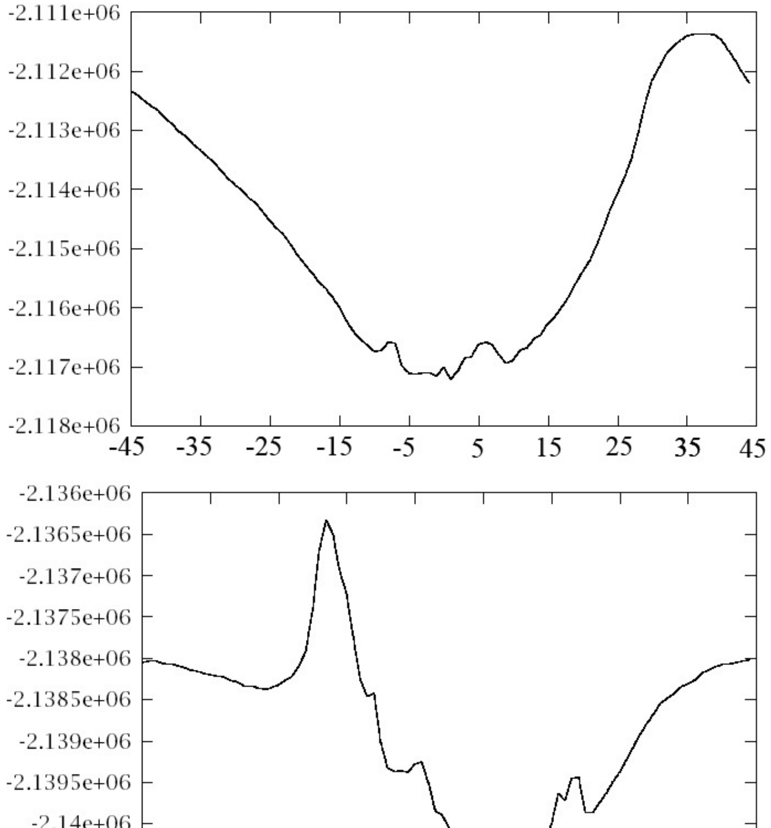


Figure 8. Plots of the equation 24 evaluated at each angle, from -45° to 45°

We need to find the angle Ψ maximizing the latter equation. It is easily calculated evaluating the function for each angle, from -45° to 45°. Fig. 8 shows a plot of this evaluation. The maximum value is the camera orientation angle.However, this is a slow process and we need to speed up. This part is explained in the next section.

4.3 Obtaining the Compass Angle From Edge Information

Now, we are interested in using the grouping information in order to speed up the overall process. The only thing we need to do is redefine the ${}_{on} / \; {}_{o}$ functions. They are defined now as:

$$
{}_{on}(\; {}_{u}) = \begin{cases} 0.99 & \text{if } \; {}_{u} \in \\ 0.01 & \text{otherwise} \end{cases}
$$

$$
{}_{o}\;(\; {}_{u}) = \begin{cases} 0.99 & \text{if } \; {}_{u} \notin \\ 0.01 & \text{otherwise} \end{cases} \tag{25}
$$

where A is the set of edges obtained during the grouping process.

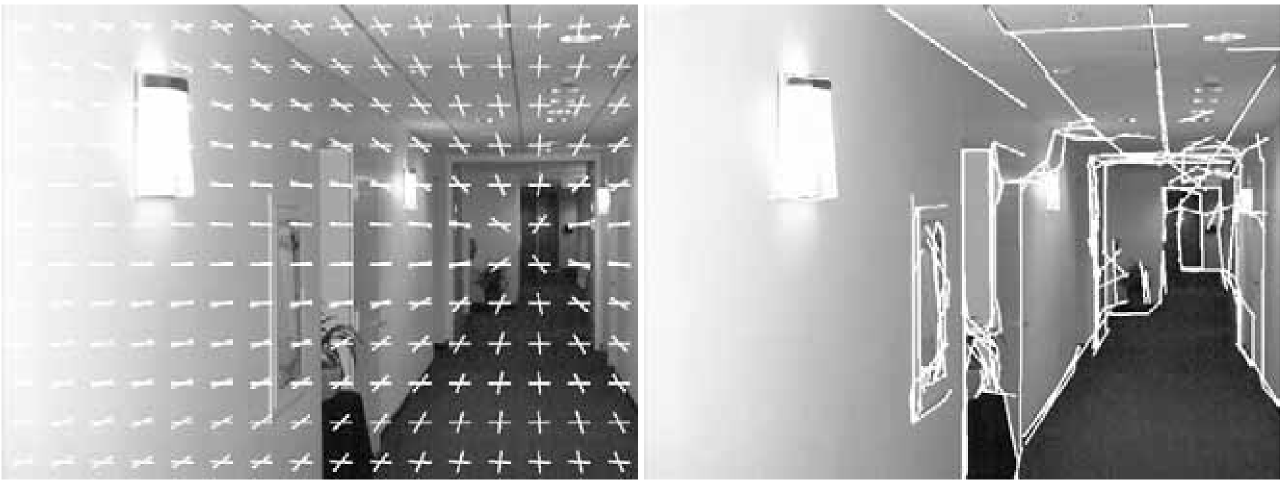


Figure 9. Grouping information and orientation angle obtained

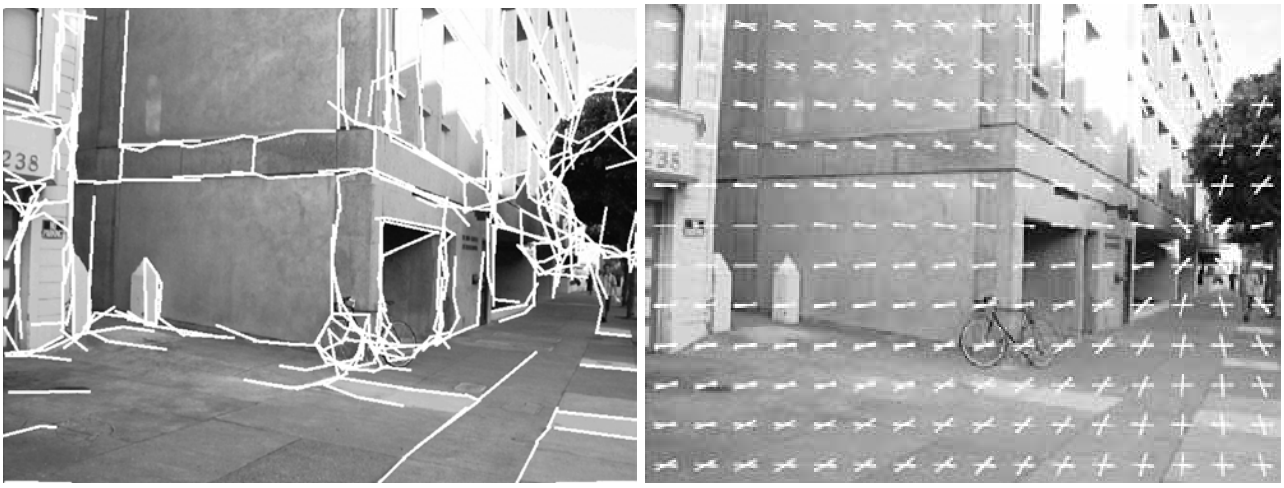


Figure 10. Grouping information and orientation angle obtained

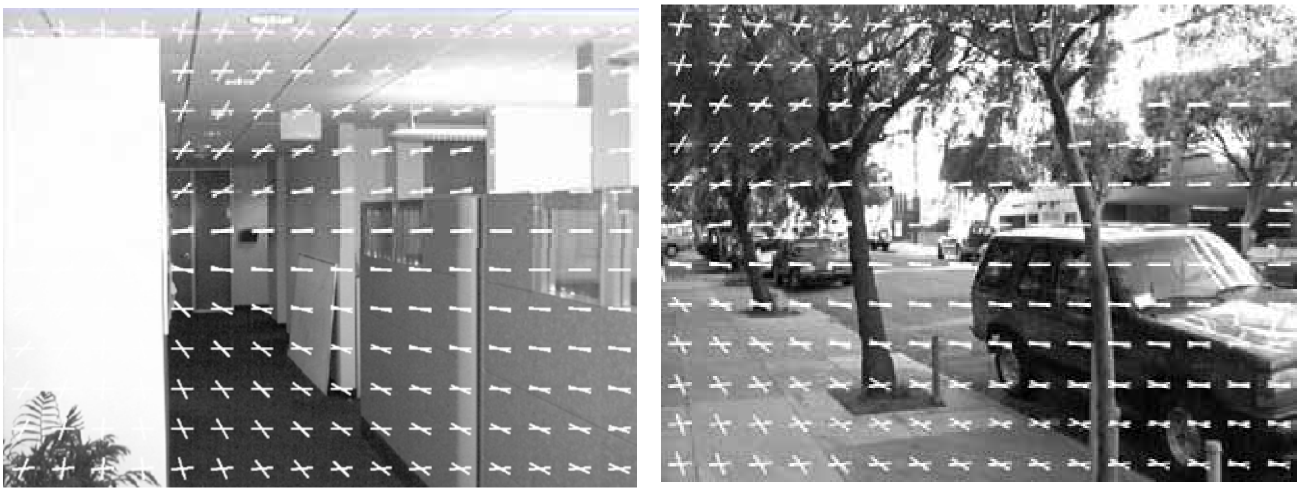


Figure 11. Two more examples of calculating the orientation angle

We now do not need to calculate the image gradient, as we have it included in the grouping information. So the pixel orientation is obtained as the normal in a point in the edge. With this information we now know which pixels are on and off the edge. The overall process is simple: just take the grouping output and, for each edge evaluate Equation 24 for each angle. The maximum value (see Fig. 8) is the camera orientation angle.

Figures 8, 9 and 10 are examples of applying this algorithm using grouping information. In both figures we have first the grouping information obtained in one of the image and its corresponding result. The crosses are a way to show the calculated angle.

5. Conclusions and Future Work

Exploiting computer vision for performing robotic tasks, like recognizing a given place in the environment or simply computing the relative orientation of the robot with respect to the environment, requires an in depth analysis of the vision modules involved in such computations. In this paper, we have considered three types of computations: local computations (for the estimation of junctions), local-to-global computations (for finding a geometric sketck) and voting-accumulation computations (for obtaining the relative orientation of the robot). We have addressed the analysis of the latter modules from the point of view of three practical requirements: reliability (robustness), efficiency and flexibility. These requirements are partially fullfiled by the methodology used: the three modules (junction detection, grouping, and orientation estimation) share elements of Bayesian inference. Sometimes, as in the case of junction detection, these elements yield statistical robustness. In other cases, as in the grouping module, the Bayesian formulation has a deep impact in the reduction of computational complexity. The Bayesian integration of feature extraction, grouping and orientation estimation is a good example of how to get flexibility by exploiting both the visual cues and the prior assumptions about the environment in order to solve a given task. On the other hand, as the basic visual cues are edges and there are fundamental limits regarding whether certain tasks relying on edge cues may be solved or not, independently of the algorithm, we also stress the convenience of having this bounds in mind in order to devise practical solutions for robotics tasks driven by computer vision.

6. References

Cazorla, M. (2000) PhD Thesis. University of Alicante, Spain

Cazorla, M. & Escolano, F., Gallardo, D. & Rizo, R. (2002). Junction detection and grouping with probabilistic edge models and Bayesian A*. Pattern Recognition. Vol. 35. pp. 1869-1881

Cazorla, M. & Escolano, F. (2003). Two Bayesian methods for junction classification. IEEE Transactions on Image Processing. Vol. 12, No. 3, pp. 317-327

Coughlan, J. & Yuille, A.L. (1999) Bayesian A* tree search with expected O(N) convergence for road tracking. LNCS 1654, pp. 189-2004

Coughlan, J. & Yuille, A. (2003) Manhattan World: Orientation and Outlier Detection by Bayesian Inference. Neural Computation. Vol. 15, No. 5, pp. 1063-88.

Faugeras, O. Three-Dimensional Computer Vision. The MIT Press, Cambridge, Massachusetts.

Geman, D. & Jedynak, B. An active testing model for tracking roads in satellite images. IEEE Transaction on Pattern Analysis and Machine Intelligence. Vol. 18 No. 1. pp. 1-14.

Harris, C.G. & Stephens, M. (1988). A combined corner and edge detection. Proceedings of the Fourth Alvey Vision Conference . pp. 147-151

Parida, L., Geiger, D. & Hummel, R. (1998). Junctions: detection, classification and reconstruction. IEEE Transactions on Pattern Analysis and Machine Intelligence. Vol. 20, No. 7. pp. 687-698.

Smith, S.M. & Brady, J.M. (1997). SUSAN = a new approach to low level image processing. International Journal on Computer Vision. Vol. 23, No. 1 pp. 45-78

Yuille, A.L. & Coughlan, J. (2000) Fundamental limits of Bayesian inference: order parameters and phase transitions for road tracking. IEEE Transactions on Pattern Analysis and Machine Intelligence. Vol. 22, No. 2. pp. 160-173

Note

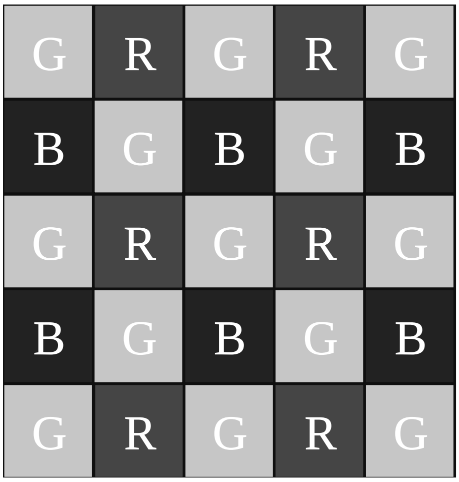
Part of this paper is reprinted from Pattern Recognition, Vol. 35, M. Cazorla, F. Escolano, D. Gallardo, R. Rizo, “Junction Detection and Grouping with Probabilistic Edge Models and Bayesian A*”, Pages 1869-1881, Copyright (2002), with permission from Elsevier.

Comparison of Demosaicking Methods for Color Information Extraction

Flore Faille

1. Introduction

Most digital color cameras are based on a single CCD or CMOS sensor combined with a color filter array (CFA): each pixel measures only one of the RGB colors. The most popular CFA is the Bayer CFA (Bayer, 1976) shown in fig. 1. Demosaicking algorithms interpolate the sparsely sampled color information to obtain a full resolution image, i.e. three color values per pixel. Many demosaicking algorithms were designed. However, even recent methods are prone to interpolation errors or artifacts, especially near edges. The most common artifacts are shown in fig. 2: "zipper" effects, wrong colors and wrong saturation of colored details. These artifacts are influenced by the sampled channel (R, G or B) and by the image content, like e.g. the orientation of the nearby edges. As a consequence, the same scene point may get a very different color value after a camera movement. This raises the two questions whether images acquired with a single chip camera can be used to extract reliable color information for further computer or robot vision tasks, and which demosaicking method suits best. Hence, this paper provides an overview and a detailed comparison of several state of the art and several recent demosaicking algorithms to enable the reader to choose the most appropriate demosaicking algorithm for his application. The previous comparisons between demosaicking methods, like the ones in (Lu & Tan, 2003; Ramanath et al., 2002), aimed at visually pleasing images. As a consequence, their evaluation criteria were, in addition to Mean Square Error (MSE) in RGB space, visual inspection and measures based on human perception like Δ^{*}_{a} (Lu & Tan, 2003; Ramanath et al., 2002). In computer vision tasks, separation between intensity and chrominance is widely used to increase robustness to illumination changes (Funt et al., 1998; Gevers & Smeulders, 1999). However, none of the previously used criteria can evaluate chrominance quality. For that reason, a detailed analysis using MSE in typical color spaces (HSI, Irb and YUV) is provided here. In addition, performance differences in colored, textured and homogeneous areas are emphasized. After an overview of the existing demosaicking algorithms, the methods chosen for comparison are introduced in section 2. The comparison framework and the results are presented in section 3. A conclusion is given in section 4.



G	R	G	R	G
B	G	B	G	B
G	R	G	R	G
B	G	B	G	B
G	R	G	R	G

Figure 1. Bayer color filter array

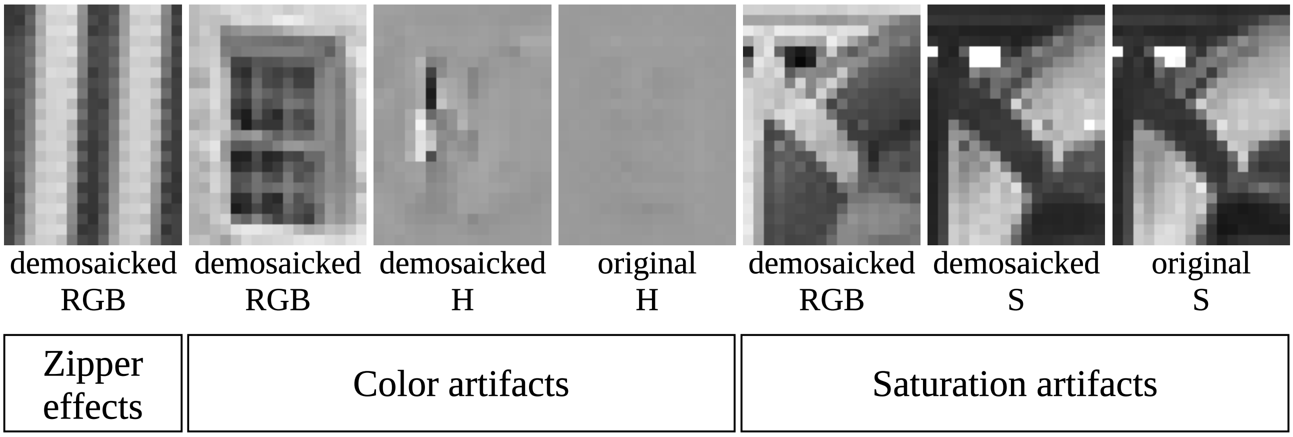


Figure 2. Overview of typical demosaicking artifacts on simulated CFA images. For a better visualization of the color and saturation artifacts, the hue (H) and the saturation (S) components are given for the demosaicked image and for the original three channel image. H and S components are scaled between 0 and 255. The images are available online in color (Faille, 2005)

2. Demosaicking Algorithms

2.1 Overview

The simplest demosaicking algorithm consists of a separate bilinear interpolation of the three channels. The obtained images are however very blurry and present many artifacts as shown in figs. 8(a) and 9(a). Demosaicking quality can be improved by enlarging the considered neighborhood. In addition, gradient information can be taken into account to adapt the used neighborhood to the image, as in (Hamilton & Adams, 1997). This allows interpolation to be performed along rather than across edges, hence reducing artifacts. Finally, many algorithms make use of the high inter-channel correlation to constrain the interpolation process: either the color differences R – G and B – G (Freeman, 1988; Hamilton & Adams, 1997) or the color ratios R/G and B/G (Cok, 1987) are assumed to remain constant in a small neighborhood. These principles improve demosaicking at a moderate increase of complexity. A good overview and comparison of many state of the art methods is given in (Ramanath et al., 2002). In this paper, the best two methods will be analyzed. The median-based postprocessing (MBP) enforces high inter-channel correlation to reduce artifacts after a first demosaicking step (Freeman, 1988). In addition to inter-channel correlation, the adaptive color plane interpolation (ACPI) takes gradients into account to select either horizontal or vertical interpolation (Hamilton & Adams, 1997). MBP and ACPI are described in more detail in sections 2.2 and 2.3.

The most interesting of the recently developed algorithms can be divided into three categories. The algorithms of the first category use principles similar to ACPI but provide more powerful and flexible ways to adapt the considered neighborhood. In (Lu & Tan, 2003) and in the first step of (Kimmel, 1999), each direction contributes to the interpolation with a weight which is proportional to the gradient inverse. In (Ramanath, 2003), the weights depend on the similarity to the center pixel after a first demosaicking step. Additionally, the framework to estimate R and B channels is improved in (Lu & Tan, 2003). This enhances the chrominance quality, so the method by (Lu & Tan, 2003) will be analyzed here. More details are given in section 2.4.

The second category of methods locally selects between horizontal and vertical interpolation as in ACPI, however based on more complex image measures than gradients. (Hirakawa & Parks, 2003) use image homogeneity in CIELAB color space.

(Omer & Werman, 2004) use the variation of R/G and B/G color ratios and the response to the Harris corner detector. As these complex measures cannot be estimated directly from the CFA sampled images, one horizontally interpolated image and one vertically interpolated image are generated first. The local direction selection is performed subsequently to generate a final image. This results in a very high computing time, even when direction selection is only performed in textured areas as in (Omer & Werman, 2004). For that reason, these methods will not be analyzed here.

The last category of methods works similarly to MBP. After a first demosaicking step, the result is postprocessed to enforce one or more constraints, hence reducing artifacts. In the second step of (Kimmel, 1999), the locally constant color ratio assumption is enforced, using thereby an adaptive neighborhood. In (Gunturk et al., 2002), a compromise between the following two constraints is reached: locally constant color differences and fidelity to sampled data. As these methods yield no significant enhancement over MBP, they will not be analyzed here.

2.2 Median-Based Postprocessing (MBP and EMBP)

MBP reduces demosaicking artifacts by enforcing high inter-channel correlation (Freeman, 1988). After a first demosaicking step, the difference images $\delta_R = R - G$ and $\delta_B = B - G$ are median-filtered. The image is then reconstructed using the filtered difference images δ_R and δ_B as well as the CFA sampled data. For example, at a sampled G pixel:

$$(\mathrm{R}',\mathrm{G}',\mathrm{B}') = (\delta_\mathrm{R} + \mathrm{G}, \mathrm{G}, \delta_\mathrm{B} + \mathrm{G}) \,. \tag{1}$$

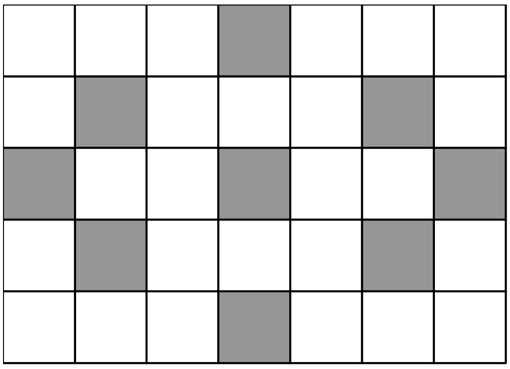


Figure 3. Kernel used for median filtering

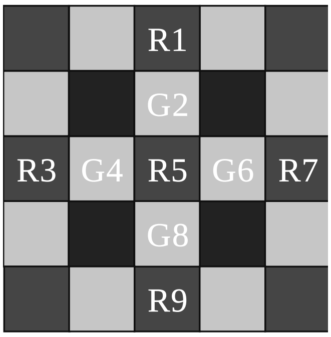
The algorithm works similarly at sampled R and B pixels. A larger filter kernel enhances the demosaicking results. As a consequence, a compromise between image quality and complexity is reached using the kernel proposed in (Freeman, 1988) and shown in Fig. 3: the value of the center pixel is the median of the nine pixel values indicated in grey. Demosaicking results are shown in figs. 8(b) and 9(b). Pixels with wrong intensity and wrong saturation appear near color edges with low inter-channel correlation such as for example red/white edges. To avoid such artifacts due to contradictions between the high inter-channel correlation model and the sampled data, an Enhanced Median-Based Postprocessing (EMBP) is used in (Hirakawa & Parks, 2003; Lu & Tan, 2003), in which sampled values are changed too. The reconstruction step becomes the same for all pixels:

$$(\mathrm{R}',\mathrm{G}',\mathrm{B}') = (\delta_\mathrm{R} + \mathrm{G}', (\mathrm{R} - \delta_\mathrm{R} + \mathrm{B} - \delta_\mathrm{B})/2, \delta_\mathrm{B} + \mathrm{G}') \tag{2}$$

The implementation proposed in (Lu & Tan, 2003) was chosen here. Already processed pixels are used to filter following pixels for a faster diffusion of the estimation. In addition, only textured areas (areas where the laplacian of the G channel is above a given threshold) are postprocessed, as homogeneous areas are less prone to artifacts. Results are presented in figs. 8(c) and 9(c). For the comparison in section 3, MBP and EMBP will be applied after the method by Lu and Tan (Lu & Tan, 2003) to obtain the best possible results.

2.3 Adaptive Color Plane Interpolation (ACPI)

ACPI is a state of the art method using gradient information and inter-channel correlation (Hamilton & Adams, 1997). As the Bayer CFA contains twice as many G pixels as R or B pixels, the G channel is interpolated first and is used to estimate R and B channels in a second step. Fig. 4 presents the G channel estimation. The interpolation is performed in the direction of the minimum gradient. To take the high inter-channel correlation into account, gradients and interpolated G values depend on the laplacian of the R (or B) channel. Similarly, estimated R and B values depend on the laplacian of the interpolated G channel. Gradient based neighborhood adaptation is only performed to estimate R (or B) values at sampled B (or R) pixels, for which four R (or B) neighbors exist. The demosaicking results are illustrated in figs. 8(d) and 9(d).



Compute the horizontal and the vertical gradients:

$H = |G4 – G6| + |R5 – R3 + R5 – R7|$

$V = |G2 – G8| + |R5 – R1 + R5 – R9|$

If H > V, interpolate along the vertical direction:

$G5 = (G2 + G8) / 2 + (R5 – R1 + R5 – R9) / 4$

else if V > H, interpolate along the horizontal direction:

$G5 = (G4 + G6) / 2 + (R5 – R3 + R5 – R7) / 4$

else use both directions:

$G5 = (G2 + G8 + G4 + G6) / 4 + (4 R5 – R1 – R9 – R3 – R7) / 8$

Figure 4. Interpolation of the G value at a sampled R pixel with ACPI (Hamilton & Adams, 1997). Sampled B pixels are processed similarly

2.4 Weighted Adaptive Color Plane Interpolation (WACPI)

WACPI (Lu & Tan, 2003) is based on the same principles as ACPI. G values are estimated first. Gradients and interpolated values depend on the laplacian of the other color channels, like in ACPI. Yet WACPI provides a more flexible framework to adapt the interpolation neighborhood to the image. Four directions (instead of two) are considered as shown in fig. 5, which enhances the performance near slanted edges and corners (see figs. 8(e) and 9(e)). The contributions of every direction to the interpolation $\tilde{G}_i$ are weighted by the gradient inverses α_i before they are summed up and normalized to build the estimate:

$$G = \frac{\alpha_{LEFT}\tilde{G}_{LEFT} + \alpha_{RIGHT}\tilde{G}_{RIGHT} + \alpha_{UP}\tilde{G}_{UP} + \alpha_{DOWN}\tilde{G}_{DOWN}}{\alpha_{LEFT} + \alpha_{RIGHT} + \alpha_{UP} + \alpha_{DOWN}} \tag{3}$$

The formulas for α_{LEFT} and $\tilde{G}_{LEFT}$ are given in fig. 6 as an example for all α_i and $\tilde{G}_i$. As can be seen, the contributions to the interpolation $\tilde{G}_i$ are the same as for ACPI. In contrast, the gradients are estimated on a larger neighborhood to compute the weights α_i. The constant additive term in the denominator of α_i (see fig. 6) avoids division by zero in homogeneous areas. The estimation of the R and B channels is also improved in comparison to ACPI, as the gradient based neighborhood adaptation is performed for the interpolation of all R and B values. To achieve this, R (or B) values at sampled B (or R) pixels are interpolated first. After this step, sampled G pixels have two sampled R (or B) neighbors and two estimated R (or B) neighbors. As a consequence, the neighborhood adaptation framework can then be applied without any restriction. The reader should refer to (Lu & Tan, 2003) for the complete algorithm description.

3. Comparison of the Algorithms

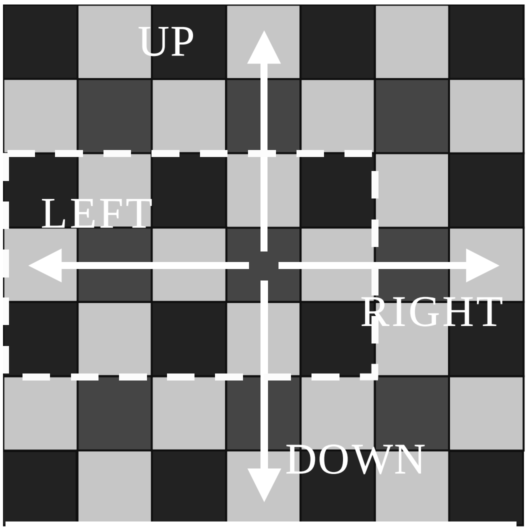
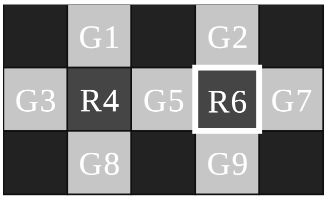


Figure 5. Considered directions and neighbourhood shown in fig. 6

The algorithms are evaluated on simulated CFA sampled images by comparison with the original three channel images. 24 images of the Kodak color image database are used. These images represent scenes of various content, like for example landscapes, persons, natural and man-made objects, as can be seen in Fig. 7.

Before the quantitative evaluation, the demosaicking results are illustrated in figs. 8 and 9 to show the different artifacts. The figures show one colorful area and one textured area of the Small Lighthouse image, a downsampled version of one of the Kodak database images used in (Kimmel, 1999; Lu & Tan, 2003).

The chosen areas illustrate best the results. To enable better visual analysis and understanding of the results, hue (H), saturation (S) and intensity (I) components are also presented, as HSI is an intuitive color space. For better visualization, the theoretical ranges of hue [−π, π] and saturation [0, 1] are mapped to [0, 255]. Bilinear interpolation produces blurred images and artifacts known as "zipper" effects near edges (see figs. 8 and 9). For ACPI and WACPI, color artifacts may appear especially near slanted edges or corners as shown in fig. 9. MBP and EMBP correct these artifacts (see fig. 9) but introduce new ones near color edges (see fig. 8): MBP generates pixels with wrong intensity and wrong saturation, whereas EMBP generates areas with wrong saturation. In addition, MBP reconstructs hue more accurately than EMBP in colorful areas (see fig. 9).



$$\tilde{G}_{LEFT} = G5 + (R6 - R4)/2$$

$$\alpha_{LEFT} = \frac{1}{1 + |G7 - G5| + |G5 - G3| + |R6 - R4| + \frac{|G2 - G1| + |G9 - G8|}{2}}$$

Figure 6. Weight and contribution of the left direction (cf. fig. 5) to the interpolation of the G value at pixel position R6

Figure 7. Four images of the Kodak color image database

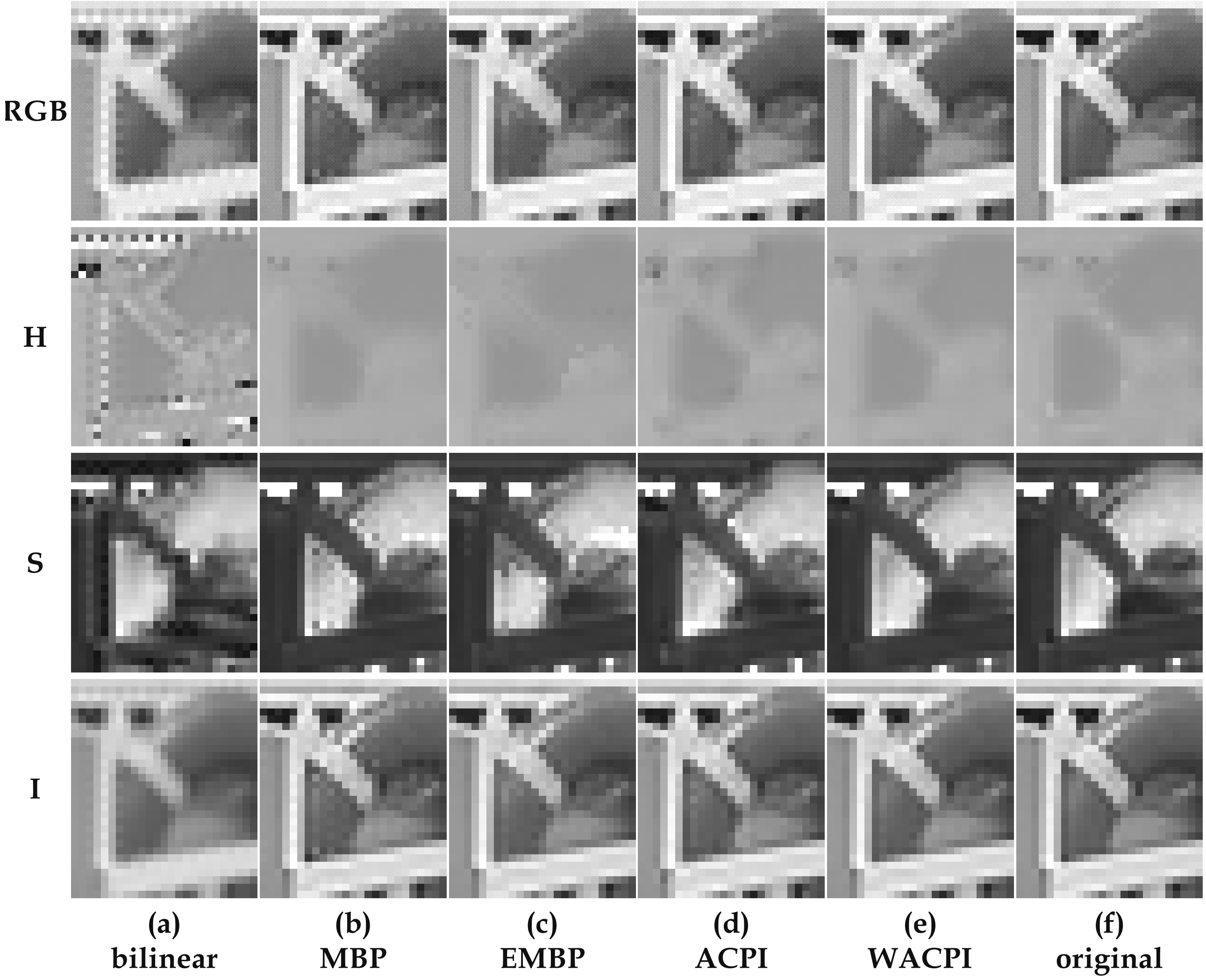


Figure 8. Enlarged colorful detail of the *Small Lighthouse* image showing a buoy. From left to right: (a) bilinear interpolation, (b) MBP, (c) EMBP, (d) ACPI, (e) WACPI, (f) original three channel image. From top to bottom: RGB image, hue, saturation, intensity. This figure is available online in color (Faille, 2005)

As mentioned before, popular color spaces for computer vision separate intensity and chrominance information to reduce sensitivity to light direction and intensity (Funt et al., 1998; Gevers & Smeulders, 1999).

To achieve this, chrominance components are based on ratios between channels. In addition, robustness to specularities can be achieved using channel differences (Gevers & Smeulders, 1999). HSI and Irb are two well-known color spaces based on these principles:

$$\begin{pmatrix} H \\ S \\ I \end{pmatrix} = \begin{pmatrix} \arctan \dfrac{\sqrt{3}(G-B)}{2R-G-B} \\ 1-\min(R,G,B)/I \\ (R+G+B)/3 \end{pmatrix} \text{ and } \begin{pmatrix} I \\ r \\ b \end{pmatrix} = \begin{pmatrix} (R+G+B)/3 \\ R/I \\ B/I \end{pmatrix} \qquad (4)$$

YUV is another popular color space, as it is linear and often directly delivered by color cameras. Y gives the intensity information. Color information is represented by channel differences U and V instead of channel ratios like in HSI and Irb spaces:

$$\begin{pmatrix} Y \\ U \\ V \end{pmatrix} = \begin{pmatrix} 0.3 & 0.59 & 0.11 \\ -0.168 & -0.33 & 0.498 \\ 0.499 & -0.421 & -0.078 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix}. \tag{5}$$

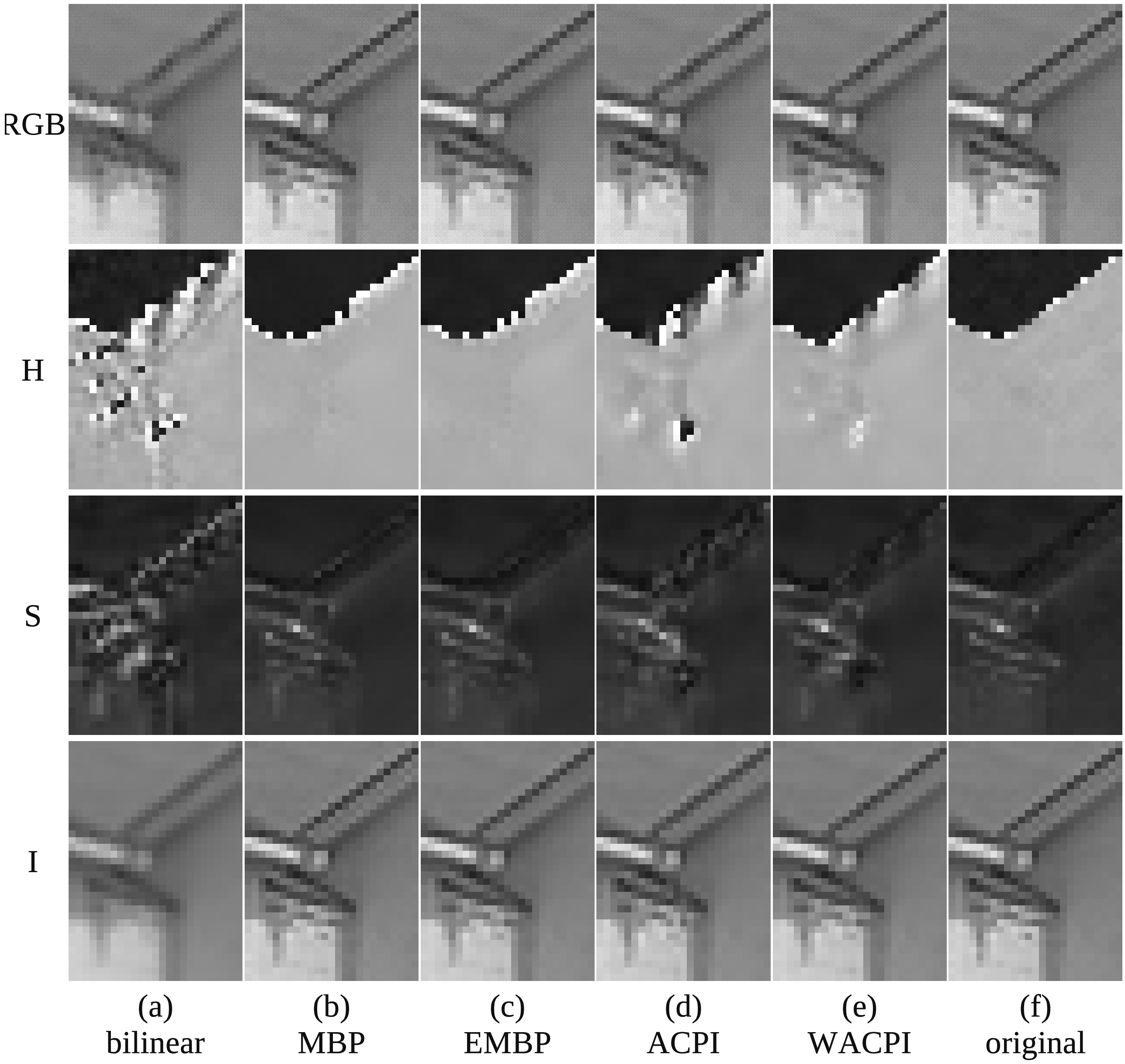


Figure 9. Enlarged textured detail of the *Small Lighthouse* image showing roofs. From left to right: (a) bilinear interpolation, (b) MBP, (c) EMBP, (d) ACPI, (e) WACPI, (f) original three channel image. From top to bottom: RGB image, hue, saturation, intensity. This figure is available online in color (Faille, 2005)

To analyze demosaicking quality for further computer or robot vision applications, performances will be evaluated using the Mean Square Error (MSE) between original and demosaicked images in HSI, Irb and YUV spaces. The MSE in RGB space is added for comparison with previous evaluations like (Lu & Tan, 2003; Ramanath et al., 2002). H, S, r and b are scaled by 100, so that their order of magnitude is similar to the other components. To avoid numerical instabilities, MSE for S, r and b was only estimated in sufficiently bright areas (I > 0.9). Similarly, MSE for H was only estimated in sufficiently colored areas (S > 0.1). In addition, a three pixel wide border was left out to suppress any influence by border effects. As the results depend on image content, the average MSE over all 24 images is presented in table 1.

The performance discrepancy in textured and in homogeneous areas is emphasized (texture was detected in the original images with a Laplacian filter). As far as complexity is concerned, ACPI, WACPI, MBP (+ WACPI) and EMBP (+ WACPI) require on average over all images 2, 8.5, 20 and 14 times as much computation time as bilinear interpolation.

Algorithm	Edges				Homogeneous areas				Entire images			
	RGB	YUV	HSI	rb	RGB	YUV	HSI	rb	RGB	YUV	HSI	rb
Bilinear	215	84.2	21.1	1.40	14.1	5.32	.900	.0545	104	40.4	10.7	.669
	87.1	49.3	.966	1.57	5.64	3.42	.0549	.0587	42.0	24.0	.460	.748
	222	45.8	95.4		15.1	3.14	6.14		108	22.2	45.9	
ACPI	38.1	18.1	3.10	.394	4.85	1.97	.370	.0245	20.1	9.60	1.79	.202
	24.9	9.59	.268	.319	2.60	1.27	.0223	.0232	13.2	5.20	.141	.168
	33.6	10.1	15.8		4.52	1.26	1.89		18.4	5.30	8.46	
WACPI	**23.6**	10.4	1.63	**.232**	**3.44**	**1.34**	.273	**.0178**	**12.4**	5.57	.936	**.118**
	13.3	5.99	**.187**	**.210**	**1.71**	**.956**	**.0173**	**.0180**	7.22	3.25	**.0959**	.108
	21.6	**5.99**	9.44		**3.52**	**.884**	**1.35**		11.7	3.15	5.08	
MBP	25.2	**8.51**	**1.62**	.249	4.01	1.52	**.269**	.0206	12.6	**4.39**	**.814**	.120
	9.93	**5.57**	.195	.212	1.91	1.09	.0204	.0212	**5.18**	**2.95**	.0976	**.106**
	18.4	6.08	**8.13**		3.67	.962	1.49		**9.72**	**3.02**	**4.20**	
EMBP	29.7	11.0	2.30	.543	3.51	1.36	.281	.0201	14.4	5.63	1.09	.245
	18.1	6.38	.264	.283	1.79	.980	.0182	.0188	8.57	3.18	.124	.134
	24.0	11.0	9.64		3.58	.938	1.36		12.1	4.86	5.07	

Table 1. Demosaicking performance near edges, in homogeneous areas and on entire images. The average MSE over 24 images of the Kodak image database is given for RGB, YUV, HSI and Irb spaces. Each row shows a channel (top row for R, Y, H and r, etc.). As I is the same in HSI and Irb, the MSE for I is only given for HSI. The best performance in each category is indicated in boldface

Bilinear interpolation, which does not use gradients and inter-channel correlation, yields by far the worst results.

Despite its low complexity, ACPI allows significant enhancement, proving the strength of the used principles. WACPI and MBP perform best. The quality of color ratios, of saturation and of color differences are on average comparable for both (see MSE for r, b, S, U and V). MBP improves texture estimation (see MSE for R, G, B, Y and I) and reduces color artifacts (see MSE for H). But its performance in homogeneous areas is worse. This is mostly due to its higher sensitivity to the inaccuracy of the high inter-channel correlation model (constant color differences in a neighborhood) in colored areas. Even very low contrast edges in colored areas may result in pixels with wrong intensity and saturation. As told in the overview in section 2.1, the recent methods similar to MBP (Kimmel, 1999; Gunturk et al., 2002) do not significantly reduce this sensitivity to model inaccuracy. The overall performance of EMBP is moderate. It better estimates homogeneous areas than MBP, but achieves poor chrominance quality compared to WACPI and MBP.

To emphasize the model inaccuracy in colored areas, table 2 gives the average MSE in areas with high saturation (S ≥ 0.3). As in table 1, WACPI and MBP perform best. Yet MBP shows the maximal performance drop, especially for the estimation of texture (see MSE for

R, G, B, Y and I) and color differences (U and V). In this case, WACPI achieves the best results. MBP's higher sensitivity to the model inaccuracy is also shown by a higher number of negative (hence invalid) interpolated values: if these are not corrected to 0, the average MSE for all 24 entire images e.g. for r becomes 0.441 for WACPI and 0.965 for MBP. To summarize, MBP better estimates texture and reduces wrong color artifacts, but it is outperformed by WACPI in homogeneous and in colored areas. This explains our observation that MBP is better on images with fine textures (e.g. landscapes or natural objects), while WACPI is better on images showing man-made objects and on close-ups. Another consequence is that MBP should be applied after white balancing, so that the number of edges in colored areas is reduced to a minimum.

ACPI				WACPI				MBP				EMBP			
RGB	YUV	HSI	rb	RGB	YUV	HSI	rb	RGB	YUV	HSI	rb	RGB	YUV	HSI	rb
30.8	14.1	2.15	.747	**19.2**	**8.05**	1.12	**.435**	28.2	9.43	**.817**	.452	27.2	9.25	1.06	.806
19.6	7.13	.756	.598	**10.7**	**4.67**	.584	.409	11.3	5.54	**.524**	**.407**	15.8	6.02	.759	.484
27.6	8.78	12.8		**18.8**	**5.49**	7.75		19.2	7.18	8.99		21.0	10.3	7.95	

Table 2. Demosaicking performances in colored areas (with S ≥ 0.3). As in table 1, the average MSE over all 24 images is given in RGB, YUV, HSI and Irb color spaces. The results for the bilinear interpolation are omitted, as it yields by far the worst results in table 1

4. Conclusion

After an overview over state of the art and recent demosaicking methods, selected algorithms were compared using images with various content. To verify if demosaicked images are suitable for computer or robot vision tasks, the average MSE in typical color spaces (RGB, YUV, HSI and Irb) was measured. While the high inter-channel correlation model improves interpolation results significantly, it was also shown to be inaccurate in colored areas. WACPI and MBP (+WACPI) provide the best results. WACPI performs better in colored and in homogeneous areas. MBP better reconstructs texture and reduces wrong color artifacts. The choice between both algorithms should depend on the application, according to the most relevant information and to the image type (indoor/outdoor, natural/man-made objects). The MBP algorithm could be enhanced by processing only edge pixels like in EMBP: this would reduce the computation time and improve the performance in homogeneous areas. In addition, regions with saturated colors could be left unchanged. If emphasis lies on efficiency, ACPI and WACPI achieve the best compromise between speed and quality.

5. References

Bayer, B. E. (1976). Color imaging array. United States Patent No. 3,971,065

Cox, D. R. (1987). Signal processing method and apparatus for producing interpolated chrominance values in a sampled color image signal. United States Patent No. 4,724,395

Faille, F. (2005) http://www.rcs.ei.tum.de/~faille/ARSbook.html

Freeman, W. T. (1988). Method and apparatus for reconstructing missing color samples. United States Patent No. 4,774,565

Funt, B.; Barnard, K. & Martin, L. (1998). Is machine colour constancy good enough?, Proceedings of the European Conference on Computer Vision (ECCV98), Burkhardt,

H. & Neumann, B. (Eds.), Vol. 1, pp. 445-459, ISBN: 3-540-64569-1, Freiburg, Germany, June 1998, Springer

Gevers, T. & Smeulders, A. W. M. (1999). Color-based object recognition. Pattern Recognition, Vol. 32, No. 3, March 1999, pp. 453-464, ISSN: 0031-3203

Gunturk, B. K.; Altunbasak, Y. & Mersereau, R. M. (2002). Color plane interpolation using alternating projections. IEEE Transactions on Image Processing, Vol. 11, No. 9, September 2002, pp. 997-1013, ISSN: 1057-7149

Kimmel, R. (1999). Demosaicing: image reconstruction from color CCD samples. IEEE Transactions on Image Processing, Vol. 8, No. 9, September 1999, pp. 1221-1228, ISSN: 1057-7149

Hamilton, J. & Adams, J. (1997). Adaptive color plane interpolation in single sensor color electronic camera. United States Patent No. 5,629,734

Hirakawa, K. & Parks, T. W. (2003). Adaptive homogeneity-directed demosaicing algorithm, Proceedings of the IEEE International Conference on Image Processing (ICIP03), Vol. 2, pp. 669-672, ISBN: 0-7803-7750-8, Barcelona, Spain, September 2003, IEEE Press

Lu, W. & Tan, Y.-P. (2003) Color filter array demosaicking: New method and performance measures. IEEE Transactions on Image Processing, Vol. 12, No. 10, October 2003, pp. 1194-1210, ISSN: 1057-7149

Omer, I. & Werman, M. (2004). Using natural image properties as demosaicing hints, Proceedings of the IEEE International Conference on Image Processing (ICIP04), pp. 1665-1670, ISBN: 0-7803-8554-3, Singapore, October 2004, IEEE Press

Ramanath, R.; Snyder, W. E.; Bilbro, G. L. & Sander, W. A. (2002). Demosaicking methods for bayer color arrays. Journal of Electronic Imaging, Vol. 11, No. 3, July 2002, pp. 306-315, ISSN: 1017-9909

Ramanath, R. & Snyder W. E. (2003). Adaptive demosaicking. Journal of Electronic Imaging, Vol. 12, No. 4, October 2003, pp. 633-642, ISSN: 1017-9909

Robot Motion Trajectory-Measurement with Linear Inertial Sensors

Bernard Favre-Bulle

1. Introduction

When referring to the term "industrial robot" in the following, we mean universal, programmable processing machines, which are usually incorporated in industrial manufacturing processes. Industrial robots are multi-functional handling automata which consist of a set of more or less rigid members, interconnected to a kinematic chain by rotational or prismatic joints. One end of this chain represents the base of the robot, the other end is equipped with a tool or a gripper to execute the desired production work. For the considerations presented here (which deal with dynamic trajectory accuracy), we think of both common programming methods, on-line and off-line.

On-line programming usually is based on a teaching-process, where the start- and end-points of a trajectory are defined by moving the robot to the desired poses. An interconnecting trajectory for interpolated movement is then defined by choosing a connective geometry from a library (line, circle, spline-curve) and by specifying additional dynamic parameters as speed or time. Additional "pass-points" can be specified which are not actually a part of the end-effector trajectory but which are passed-by within a certain distance. In point-to-point movements the end-effector does not follow a user-determined path, but is rather defined by the motion of drives in axis-space.

Off-line programming makes use of the known kinematic properties of the robot, the geometrical and structural conditions of the work task and a simulative environment to test the program before implementation on the actual shop-floor. The simulation environment can handle additional tasks like cycle time evaluation, robot cell layout optimization, reach determination, trajectory optimization, simulation and calibration. When a program is transferred from simulation to a real robot station, considerable disparities between simulated and real results can be observed. Deviations of several millimeters can ensue—sufficient to make manufactured parts unusable.

1.1 Robot's Dynamic Errors and Error Measurement

When accurate six-degrees-of-freedom paths with specified maximal errors are required, there is a need for specialized measurement systems for quality assurance and a standard to define a "common language" and information interface between robot manufacturers and robot users. Concerning robot performance characteristics, there is a common standard of evaluating manipulator performance. The International Organization for Standardization (ISO) published the ISO 9283 standard of *Performance Criteria and Related Test Methods* for industrial manipulators. Many robot manufacturers and users utilize this standard to determine the requirements and actual performance of their products and publish extended specifications based a subset of the tests recommended in ISO 9283. The main points covered in this standard are

- unidirectional pose accuracy and pose repeatability,
- multi-directional pose accuracy variation,
- distance accuracy and distance repeatability,
- pose stabilization time,
- drift of pose characteristics,
- path accuracy and path repeatability,
- cornering deviations,
- path velocity characteristics,
- minimum positioning time and
- static compliance.

Within the scope of the present article, we mainly deal with dynamic performance parameters of the robot motion. Thus, the major criteria to be measured are path accuracy and path repeatability, path velocity characteristics, cornering deviations and pose stabilization characteristics such as swing-over behavior and settling time. There are a lot of sources for deviations between desired and actual robot motion. First of all we have to consider an industrial robot (without external sensors) as a feedback-controlled motion system, which measures kinetic feedback signals at a different location than the task would optimally require, i. e. at the joint drives. The robot's internal sensors usually measure the joint angle positions (and joint velocities, either directly or indirectly) at the drive axes. The joint angles on the robot body are influenced by transmission gear effects (transmission ratios, backlash, elastic deformation of the gears, nonlinear behavior, friction etc.). The robot's mechanical links usually cannot be treated as rigid mechanical elements, since they are elastic and also show a nonlinear mechanical behavior to a certain extent, especially, if high forces are applied to them. Another main source for errors is caused by friction- and backlash-effects within intermediate coupling-elements between the joints and the links. When the motion speed of the robot's links rise, the time dependent interactions between the robots masses, the payload masses and the inertial effects cannot be neglected anymore. At high-speed motion, additional error sources arise from centrifugal and coriolis forces, as well as from mass inertia. All these effects must be taken into consideration for industrial robot calibration. Much research work has been done to achieve an accurate dynamic calibration by means of external sensors (for example Spiess, Vincze et al. 1997, Albada et al. 1996 and Hinüber 1993). Within the presented approach, we utilize inertial sensors, mounted to the robot's end-effector in order to determine its actual dynamic path. We will present a solution to avoid the use of expensive gyroscopic sensors on the basis of a set of linear accelerometers.

1.2 Application of Inertial Measurement Methods

Inertial navigation has a long history (Wrigley, Wrodbary, and Hovorka, I. 1957). The original idea was to utilize inertial effects for the guidance of vehicles (aircraft, spacecraft, missiles, ships etc.). In a historical definition, the guidance process consists of

- measurement of vehicle position and velocity,
- computation of control actions necessary to properly adjust position and velocity, and
- delivery of suitable adjustment commands to the vehicle's control system.

Basically, an inertial guidance system consists of a set of linear accelerometers, e. g. mounted on a gyro-stabilized platform, some kind of computer equipment to perform the inertial calculations and a set of actors to influence the path of the vehicle. A horizontally stabilized platform has the advantage that accelerometer signals can be mathematically integrated within a coordinate system with known orientation relative to the earth's surface, in order to derive speed and actual position of the vehicle. In the first implementations, the platform was stabilized by means of mechanical servo-gyroscopes. Nowadays, these have been replaced by laser-gyros, which show an ever increasing performance. Several new technologies are in development as e. g. an atom interferometer gyroscope, reported in Gustavson, Bouyer and Kasevich (1997), which is based on the Sagnac-effect and which is capable of measuring the earth's rotation rate in a laboratory setup. Systems without stabilized platforms are called "strap-down inertial systems" and have found wide applications in missile systems. They also require gyroscopes, which are comparatively expensive devices for industrial applications. Since one or two decades, inertial navigation systems are also employed for terrestrial vehicles, and thus have found their application for mobile robotics, too.

In aeronautics and spacecraft as well as terrestrial navigation, inertial methods are usually combined with other sources of navigation information. The reason lies in the following fact: Position, orientation, speed and acceleration of the system under measurement must be derived from linear accelerations and angle speeds by multiple integration vs. time. Thus, integrator offset errors and noise contribute to an increasing total position error in time. Starting from an initialization sequence (initial posture of the vehicle, including all temporal derivatives), the inertial measurement system must rely on its high accuracy in order to keep the overall error low. So, the total error rises with operation time (nevertheless, commercial inertial navigation systems for airplanes can keep the total position error as low as a few hundred meters for a 24-hour flight, even without re-calibration and without using other position information).

Concluding, the strength of inertial navigation lies in the short term measurement of highly dynamic motion, as it is the case with industrial robot motion analysis. A proper initialization of the system at the beginning of motion is required, and thus needs at least one other (static) source of navigation information.

2. Inertial Measurement Techniques

Comparing the application properties of inertial vehicle navigation and inertial robot measurement, there are significant differences. In case of robot trajectory analysis, there are

- shorter measurement times,
- shorter absolute path lengths,
- higher accelerations and more acceleration changes per time unit,
- a constant gravity vector along the measurement path and
- easy ways to determine the initial conditions before measurement.

All these circumstances even favor the application of inertial robot path analysis in contrast to vehicle navigation, which on the other hand has proven to be very successful in the past. We are going to recall the basic principles of inertial measurement in the following. From this basis, we will derive a concept for a strap-down inertial measurement platform without gyroscopes. Although low-cost gyroscopic sensors are available on the market today, their application range is limited due to performance parameters. To avoid the relative high cost for ultra-precision laser gyros, we go for linear accelerometer setups in the present approach.

2.1 Measurement Principles

There are four main system components which are necessary for inertial measurement. We think of a sensor head which is mounted at the end-effector flange (strap-down system) and which contains all necessary components for data acquisition. Categorizing the included devices by their functions, we need

- a sensor system to measure the resulting acceleration vector,
- a device to determine the present orientation in space, relative to the gravity vector,
- a computer system with data acquisition capabilities and
- a real-time clock.

The resulting acceleration vector of the sensor head can be measured by three orthogonally mounted acceleration sensors. The gravity vector needs to be subtracted from the resulting acceleration vector. This procedure requires a) the knowledge of the orientation of gravity, relative to the sensor head orientation and b) the absolute value of gravity acceleration. The orientation of the sensor head can also be determined by a set of accelerometers. After being initialized at the beginning of the measurement process, the actual orientation can be calculated by integrating the angular accelerations twice with respect to time. At initialization time, the gravity vector is known in orientation and size, thus can be tracked and isolated from motion acceleration during rotation of the sensor head.

General motion equation of a rigid body in 3-D space

Let us consider a rigid body with its center of mass **S** which is moved through space by an industrial robot. The motion has six degrees of freedom, and allows each kind of translation and rotation. There is a homogenous acceleration field, induced by the earth's gravity. A rigid body is a system of particles in which the distances between the particles do not vary. To describe the motion of the body we use two different coordinate-systems, a space-fixed system (x_0, y_0, z_0) (vectors with time derivations marked with index *F*) and a moving system (x, y, z) which is rigidly fixed in the body and participates in its motion (vectors with time derivations marked with index *B*). Now let **A** be a point inside the rigid body (see Fig. 1).

With the notation of Fig. 1, we can write down the fundamental motion equation of a rigid body in space with Eq. (1).

$$\ddot{\mathrm{A}}_F = \ddot{\mathrm{R}}_{F,tran\,l.} + \ddot{\mathrm{A}} \quad + \dot{\boldsymbol{\omega}} \times \mathrm{A} + 2\boldsymbol{\omega} \times \dot{\mathrm{A}} \quad + \boldsymbol{\omega} \times (\boldsymbol{\omega} \times \mathrm{A}) + \mathrm{G} \tag{1}$$

Index *F* … time derivations with reference to the space-fixed inertial system

Index *B* … time derivations with reference to the body coordinate system

$\ddot{\mathrm{A}}$ … acceleration of point **A** relative to the center of mass **S**

$\ddot{\mathrm{R}}_{F,tran\,l}$ … translational acceleration of **S**

$\dot{\boldsymbol{\omega}} \times \mathrm{A}$ … tangential-acceleration

$2\boldsymbol{\omega} \times \dot{\mathrm{A}}$ … coriolis-acceleration

$\boldsymbol{\omega} \times (\boldsymbol{\omega} \times \mathrm{A})$ … centripetal-acceleration

G … vector of gravity

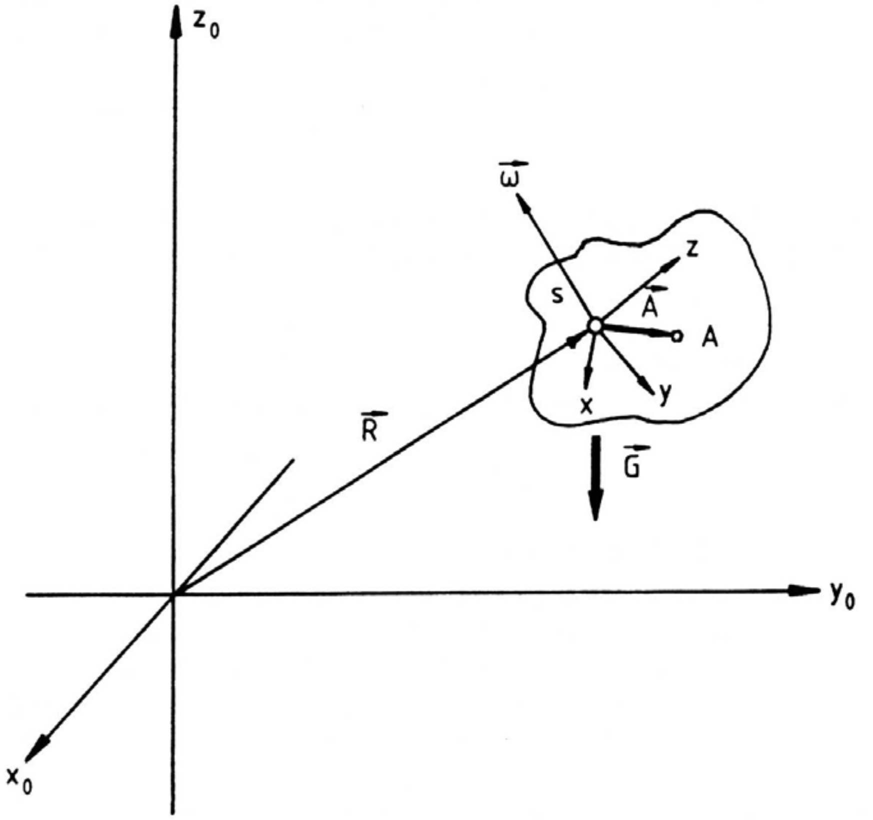


Figure 1. A rigid body in space with six degrees of motion freedom

In case that the point **A** is fixed to the body, the terms $\ddot{\mathrm{A}}_\mathrm{B}$ and $2\boldsymbol{\omega}\times\dot{\mathrm{A}}_\mathrm{B}$ become zero and we get

$$\ddot{\mathrm{A}}_\mathrm{F} = \ddot{\mathrm{R}}_{\mathrm{F,transl.}} + \dot{\boldsymbol{\omega}}\times\mathrm{A} + \boldsymbol{\omega}\times(\boldsymbol{\omega}\times\mathrm{A}) + \mathrm{G} \tag{2}$$

The body coordinate system is the coordinate system of the sensor head, and forms the basis for the description of the position and orientation of the accelerometers. To calculate the acceleration $\ddot{\mathrm{A}}_{\mathrm{F,n}}$ at the location of the *n*-th accelerometer in the sensor head, we need a coordinate transformation which is determined by the rotation matrix $\mathrm{C}_{\mathrm{F,n}}$, thus getting

$$\ddot{\mathrm{A}}_{\mathrm{B,n}} = \ddot{\mathrm{A}}_{\mathrm{F,n}} \cdot \mathrm{C}_{\mathrm{F,n}} . \tag{3}$$

$\mathrm{C}_{\mathrm{F,n}}$ contains the Euler-angles $\varphi_\mathrm{x},\varphi_\mathrm{y},\varphi_\mathrm{z}$ which describe the orientation of the *n*-th accelerometer in the inertial system. When gravitation influence is present (as it is usual in laboratory environments) we have to take this into account:

$$\ddot{\mathrm{R}}_\mathrm{B} = \ddot{\mathrm{R}}_{\mathrm{B,transl.}} + \mathrm{G}_\mathrm{B} \tag{4}$$

In order to calculate the position and orientation of the sensor head with reference to the fixed-world coordinate system at any time, we have to determine the progression of $\ddot{\mathrm{R}}_\mathrm{B}$ and $\boldsymbol{\omega}$ in time from the sensor signals. The number of accelerometers must be at least as high as the number of spatial degrees of freedom.

A system of *i* linear independent acceleration signals in the sensor head leads to a system of *i* equations with variables, consisting of the components of $\ddot{\mathrm{R}}_\mathrm{B}$ and $\dot{\boldsymbol{\omega}}$. If the equation system is solvable, we can isolate $\ddot{\mathrm{R}} \quad = \ddot{\mathrm{R}}_{\ ,tran\ l.} + \mathrm{G}$ and $\dot{\boldsymbol{\omega}}$. The actual values of $\varphi_x, \varphi_y, \varphi_z$ at a given time can be determined by integration of $\dot{\boldsymbol{\omega}}$ with respect to time, using $\varphi(0)$ as the integration constant. The gravity influence needs to be compensated by

$$\ddot{\mathrm{R}}_{\ ,tran\ l.} = \ddot{\mathrm{R}} \quad - \mathrm{G} \quad , \tag{5}$$

which we will call "auto-compensation" of gravity acceleration (see next paragraph). To get the translational acceleration in fixed world coordinates, we use the direction-cosine matrix **C**

$$\mathrm{C}^T_{F,} \quad = \mathrm{C}(\varphi_x, \varphi_y, \varphi_z), \tag{6}$$

and get

$$\ddot{\mathrm{R}}_{F,tran\ l.} = \ddot{\mathrm{R}}_{\ ,tran\ l.} \cdot \mathrm{C}^T_{F,} \quad . \tag{7}$$

$\varphi(t)$ can be calculated from $\dot{\boldsymbol{\omega}}$ by integration. We get $\mathrm{R}(t)$ also by integration:

$$\mathrm{R}(t) = \iint \ddot{\mathrm{R}}_{F,tran\ l.}(t) dt^2 + \mathrm{R}(0). \tag{8}$$

Auto-compensation of gravity influence

To obtain the pure translatory acceleration component in the sensor head coordinate system, we need to know exactly the amount and orientation of $\mathrm{G} \quad (t)$ at each moment. Since $|\mathrm{G} \quad (t)| = \mathrm{const.}$ under laboratory conditions, we can calculate $\mathrm{G} \quad (t)$ by means of the difference between $(\varphi_x, \varphi_y, \varphi_z)^T$ and the initial orientation of the sensor head $(\gamma_x, \gamma_y, \gamma_z)^T$.

$$\mathrm{G} \quad (0) = g \cdot (1 \quad 1 \quad 1)^T \cdot \mathrm{C}_{(\gamma)}, \tag{9}$$

$$\mathrm{G} \quad (t) = g \cdot (1 \quad 1 \quad 1)^T \cdot \mathrm{C}_{(\varphi - \gamma)}, \tag{10}$$

with

$C_{(\gamma)}$ … direction cosine matrix with the initial angles $\gamma_x, \gamma_y, \gamma_z$

$C_{(\varphi-\gamma)}$ … direction cosine matrix with the angle differences $\varphi_x - \gamma_x, \varphi_y - \gamma_y, \varphi_z - \gamma_z$

g … local gravity acceleration (about 9,81 ms^{-2})

Now, we obtain the current values of $\varphi_x, \varphi_y, \varphi_z$ by integration:

$$\varphi(t) = \iint \dot{\boldsymbol{\omega}}(t) dt^2 + \varphi(0). \tag{11}$$

The auto-compensation of gravity now can be accomplished by finally calculating (5). The process of auto-compensation in gyro - less strap - down systems is very sensitive to any measurement- and computational errors, since the double integration of $\ddot{R}(t)$ and $\dot{\boldsymbol{\omega}}$ propagate progressively in time, and the magnitude of the gravity vector may be in the same order of magnitude as the motion accelerations of the sensor head.

2.2 Error Influences

Inertial Navigation Systems ("INS") are intrinsically exposed to a series of error influences. Since the operation principle of each type of INS is based on the integration of signals over time, the total measurement error grows with progression of time.
Schröder-Brzosniowsky (1969) divides the major sources of error in conventional strap-down systems for aeronautics into several categories. This view is based on INS with mechanical gyroscopes. Replacing mechanical gyros by laser gyroscopes or utilizing INS without gyroscopes avoids the occurrence of certain error categories. In the following list, items marked with one asterisk become obsolete when replacing mechanical gyros with laser gyros. Two asterisks indicate errors which are not present in gyro-less systems.

- Drift of gyroscope**
- Mechanical torque perpendicular to the gyro-axis*
- Alignment Errors of gyro**
- Accelerometer system errors
- Accelerometer alignment errors
- INS alignment errors
- Altimeter system errors (for correction of gravity, coriolis- and centripetal forces)
- Geodesic errors (with respect to gravity compensation)

Influence of gravity

As already outlined in the previous paragraph, the physical quantity of acceleration is uniform with respect to motion acceleration and gravity. Thus, the gravity influence needs to be compensated by means of additional information in order to gain measurement signals which are only based on motion-induced acceleration. We can write

$$\ddot{R}_i = \frac{F_B + F_G}{m} = A + g_i, \tag{12}$$

with

$\ddot{R}_i$	…	acceleration vector of the seismic mass
m	…	magnitude of seismic mass
$F_B/m, A$	…	motion-induced acceleration of the seismic mass
F_G/m	…	gravity-induced acceleration of the seismic mass
Index *i*	…	quantities with respect to a fixed, non-rotating inertial system

In order to compensate gravity, the gravity-induced acceleration vector needs to be known both in magnitude and orientation. Gravity influence is the most dominant source of error in connection with inertial measurement systems for robotic trajectory analysis.

Influence of earth rotation

The earth represents a system which performs complex motion processes within the stellar system. When neglecting its movement around the sun along the ecliptic, the tidal forces induced by the gravity of moon and sun as well as the earth's precession (conical motion of the earth axis along a circle within 26000 years), the remaining major influence for inertial navigation is the diurnal rotation of the globe with an angle speed of 7,3.10^{-5} rad/s. The acceleration vector of a mass point in an inertial system can be expressed as

$$\ddot{\mathrm{R}}_i = \ddot{\mathrm{R}}_r + \dot{\boldsymbol{\omega}} \times \mathrm{R} \; + 2\boldsymbol{\omega} \times \dot{\mathrm{R}}_r + \boldsymbol{\omega} \times (\boldsymbol{\omega} \times \mathrm{R}) = \mathrm{A} + \mathrm{g}, \tag{13}$$

where $\boldsymbol{\omega}$ is the rotation vector of the earth. The index i denotes acceleration and speed measured in a fixed inertial system. Index r refers to the rotating coordinate system of the earth (diurnal motion). Note, that only the first and second derivation of **R** with respect to time need to be distinguished with these indices.

For robot trajectory analysis, we assume a typical measurement duration of a few seconds, which represents the time it takes for the end-effector to move from the starting point of its trajectory to the endpoint. Under these circumstances, the effects of earth rotation can be neglected, and so Eq. (13) can be simplified to

$$\ddot{\mathrm{R}}_i = \mathrm{A} + \mathrm{g}. \tag{14}$$

2.3 A Two-Dimensional Inertial System Model without Gyroscopes

We will clarify the proposed principle of inertial motion analysis by means of a simple two-dimensional example. We discuss a sensor-head, consisting of three linear accelerometers, arranged like in Fig. 2.

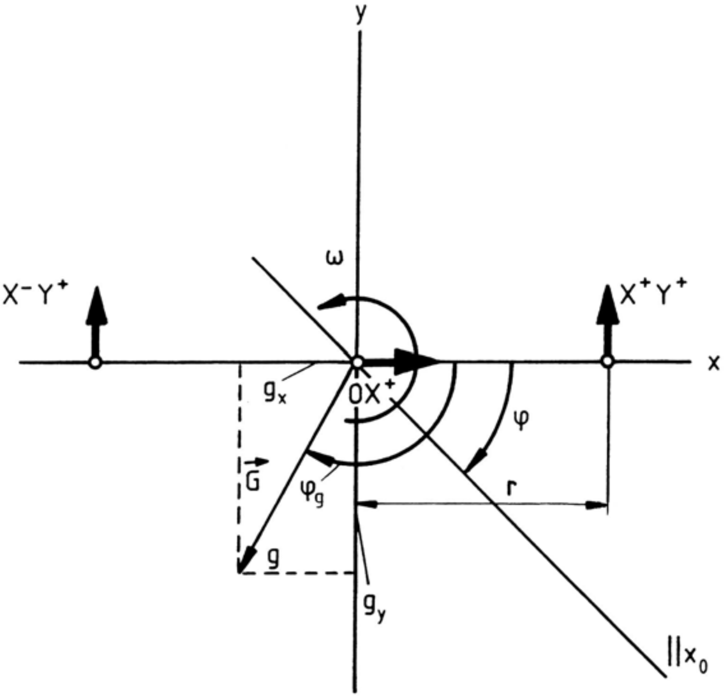


Figure 2. A simple two-dimensional arrangement of three linear accelerometers

The dynamic sensor signals will be denoted as BX^+Y^+, BX^-Y^+ and BOX^+, as indicated by the corresponding arrows in Fig. 2 to show the main sensitivity axes of the accelerometers. The motion-induced acceleration can be split up into two cartesian components b_x and b_y. The parasitic transversal sensitivity of the accelerometers is taken into account with the parameter α. The denotation of variables in equations (15), (16) and (17) are corresponding to the diagram in Fig. 2. The terms *R* denote random components in the sensor signals, such as noise or thermal offset errors.

$$BX^{+}Y^{+} = b_y + g_y + \alpha\left|b_x + g_x\right| + r\dot{\omega} + \alpha r\omega^2 + R_{X^{+}Y^{+}} \tag{15}$$

$$BX^{-}Y^{+} = b_y + g_y + \alpha\left|b_x + g_x\right| - r\dot{\omega} + \alpha r\omega^2 + R_{X^{-}Y^{+}} \tag{16}$$

$$BOX^{+} = b_x + g_x + \alpha\left|b_y + g_y\right| + R_{OX^{+}} \tag{17}$$

Solving the equation system as explained in paragraph 2.1, we get

$$b_x + g_x = BOX^{+} - R_{OX^{+}} - \alpha/2\left|BX^{-}Y^{+} + BX^{+}Y^{+} - R_{X^{+}Y^{+}} - R_{X^{-}Y^{+}}\right| \tag{18}$$

$$b_y + g_y = 1/2\,(BX^{+}Y^{+} + BX^{-}Y^{+} - R_{X^{+}Y^{+}} - R_{X^{-}Y^{+}}) - \alpha\left|BOX^{+} - R_{BOX^{+}}\right| - \alpha r\omega^2 \tag{19}$$

$$\dot{\omega} = 1/2r\,(BX^{+}Y^{+} - BX^{-}Y^{+} - R_{X^{+}Y^{+}} + R_{X^{-}Y^{+}}) \tag{20}$$

$$\omega = \int \dot{\omega}\,dt + c(0),\ c(0) \text{ from initial conditions} \tag{21}$$

The auto-compensation of gravity can be achieved as following:

$$\varphi = \int \omega\,dt + \varphi(0), \tag{22}$$

$$\varphi_g = \varphi - \gamma, \tag{23}$$

with γ as initial sensor head orientation,

$$b_x = (b_x + g_x) - |g|\cos\gamma, \text{ and} \tag{24}$$

$$b_y = (b_y + g_y) - |g|\sin\gamma \tag{25}$$

To get the translational accelerations, expressed in coordinates of the fixed inertial system, we need to apply coordinate transformations as following:

$$\begin{pmatrix} \ddot{s}_x \\ \ddot{s}_y \end{pmatrix} = \begin{pmatrix} b_x \\ b_y \end{pmatrix} \begin{pmatrix} \cos\varphi & \sin\varphi \\ \sin\varphi & \cos\varphi \end{pmatrix} \tag{26}$$

The final result of the navigation task is to determine the location of the sensor head by integraton of Eq. (26):

$$s_x(t) = \iint \ddot{s}_x(t)\,dt + s_x(0), \tag{27}$$

$$s_y(t) = \iint \ddot{s}_y(t)dt + s_y(0) \tag{28}$$

The discussed calculations are visualized as a block diagram in Fig. 3.

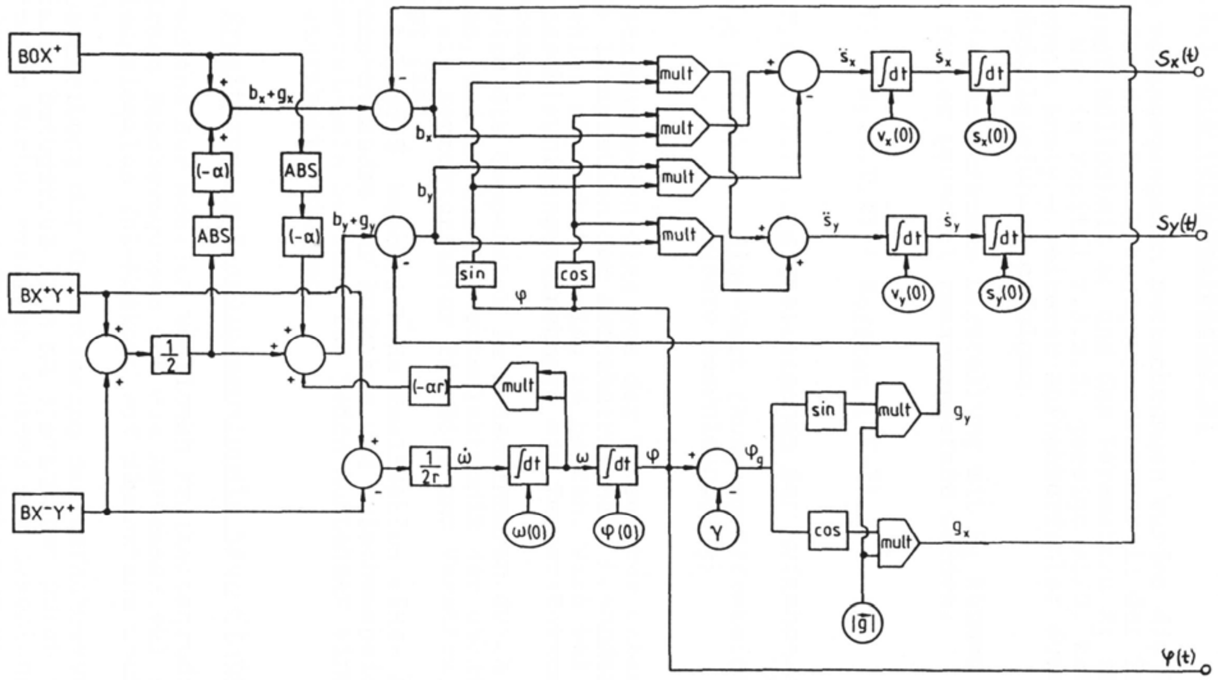


Figure 3. Block diagram for the calculation task with reference to Fig. 2

2.4 Sensors

Industrial acceleration sensors utilize a wide variety of different physical operation principles. We restrict the scope in the present context to *seismic* principles, where the acceleration to be measured exerts a force on a seismic mass. The force is then measured, using different physical effects. The operation principle of spring/mass/damper systems is usually applied in piezo-electric or piezo-resistive transducers. These low-cost sensors types offer a stable performance and a wide application range, but usually don't satisfy the requirements for inertial navigation tasks.

Requirements for inertial navigation accelerometers

Inertial navigation tasks require double integration of acceleration signals to obtain spatial information. This principle imposes specific requirements on the performance parameters of transducers. The following list shows typical requirements, which are fulfilled by the high performance accelerometers series Q-Flex (Honeywell) for aerospace INS:

- Low bias (signal, present at stationary operation at zero gravity without acceleration), typically < 5.10^{-6} g
- Input range +/– 60 g approx.
- Low misalignment (typically < 2000 μrad)
- Low temperature sensitivity on bias (< 20 ppm/°C) and scale factor (< 200 ppm/°C)
- Low cross sensitivity (< 10^{-6} g)
- High linearity (<10^{-5} g/g^2)
- Low intrinsic noise (< 7 μg at 10 Hz, < 1,5 mg at 500–10000 Hz)

High performance accelerometers for inertial navigation purposes are usually designed as servo systems, where a spring-supported seismic mass is held at a mechanical zero position by means of an electromechanical actuator. A control loop is built around the actuator and a position sensor. The closed loop system is kept in balance by force compensation, and the required current to hold the mass in balanced zero position is then proportional to the external acceleration. Fig. 4 shows the model QA3000 from Honeywell for aerospace navigation.

Figure 4. QA3000 Q-Flex® Accelerometer from Honeywell

3. A Three-Dimensional Inertial Sensor Head without Gyroscopes

The principle of gyro-less inertial navigation can be extended for three-dimensional measurement. To analyze trajectories in space (position and orientation), at least 6 accelerometers have to be used. Fig. 5 shows the basic coordinate system for a 3D sensor head.

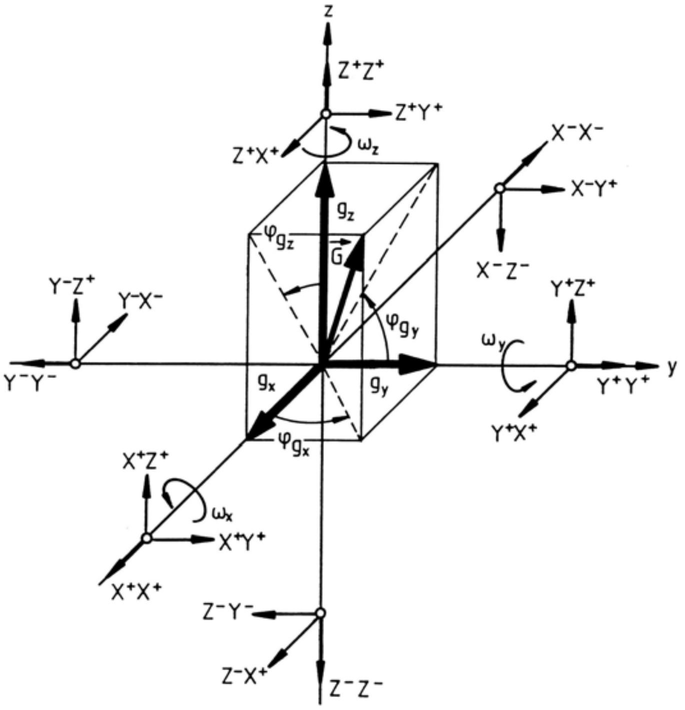


Figure 5. Base coordinate system and accelerometer systems for three-dimensional navigation

An example of a simple accelerometer setup is shown in Fig. 6. The axes of main sensor sensitivity all intersect at the origin of the system. It is easy to understand, how such a setup can measure pure translational accelerations. Changes in orientation of the sensor head can be sensed by means of the centripetal accelerations. Since these are proportional to ω , we can see that the said setup A will be only favorable for high rotation rates. For slow rotations, the centripetal accelerations are very small, which leads to a poor signal/noise ratio.

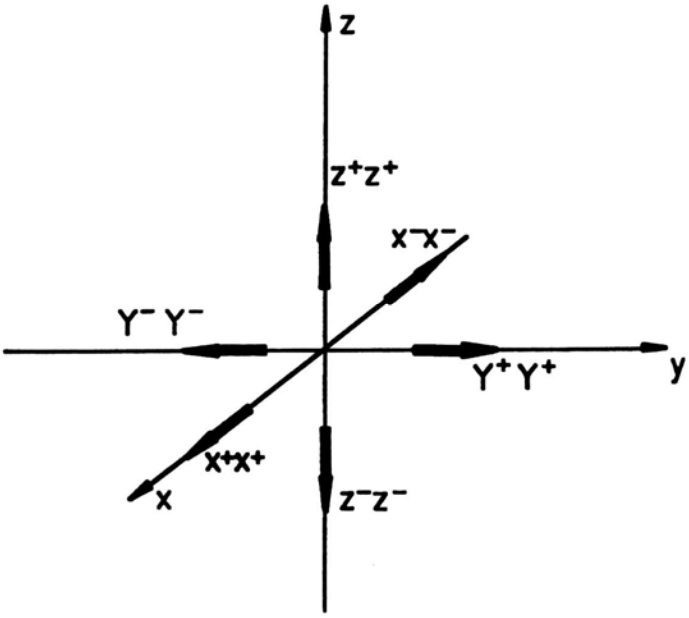


Figure 6. Sensor setup A

To overcome this drawback, one could use a different setup B (Fig. 7). The translational accelerations can be calculated from the average of two corresponding transducers. The orientation change is derived by solving the corresponding equation systems.

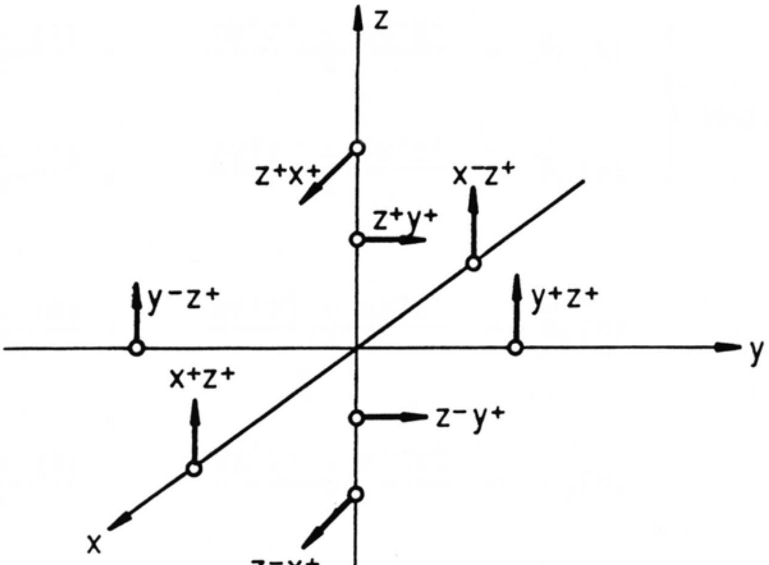


Figure 7. Sensor setup B (using redundancy)

For example, we get

$$\dot{\omega}_x = \frac{BY^+Z^+ - BY^-Z^+ - BZ^+Y^+ + BZ^-Y^+}{-4}, \tag{29}$$

and

$$\omega_z = \frac{BX^+Z^+ - BX^-Z^+ + BZ^+X^+ - BZ^-X^+}{2(\int \dot{\omega}_x(t)dt + \omega_x(0))}. \tag{30}$$

Counting the number of accelerometers in Fig. 7 (eight) we can see, that there is redundancy, i. e., certain kinetic parameters can be determined in different ways. Doing so, one can improve the accuracy of the total measurement result by choosing appropriate accelerometer signals with optimal signal amplitude.

4. Accuracy Improvement by Utilizing Redundant Motion Information

In aeronautical and space applications, INS solutions make use of redundant information to improve the overall accuracy of the position and speed measurement, especially for long-term measurements. The barometric pressure e. g. can be used to estimate the actual distance from the earth's gravity center. The relative airspeed can give an indication of the vehicle speed with respect to the ground, if wind data are available.

Also for robot trajectory analysis with gyro-less strap-down INS, different means of redundancy can be applied to improve overall accuracy of measurements. A few methods and aspects are going to be discussed in the following.

4.1 Utilizing Redundant Accelerometers

As already discussed in the previous paragraph, an increase in the number of accelerometers can improve the overall accuracy.

- Averaging the signals of two corresponding redundant transducers, measuring the *same* quantity can improve signal-to-noise ratio
- Transducers, measuring different quantities at a specific moment (e. g. ω and $\dot{\omega}$) can be selectively chosen on the basis of their current signal magnitude
- Alternative methods can be employed for a more universal approach (Complementary- and Kalman Filters)

Complementary Filters

Higgins (1975) introduces the principle of complementary and Kalman filters and gives a comparison. A complementary filter is used to obtain the estimation of a signal out of two redundant information sources (Fig. 8).

Both signals x and y have their origin in the same information source, although they result from different measurements (e. g. from different transducers). The complementary filter obtains an estimation $\hat{z}$ by filtering the signals through *complementary* networks $1-G(s)$ and $G(s)$. If, for example, signal y is disturbed by high-frequency noise, then it is appropriate to choose a low-pass characteristic for $G(s)$, thus obtaining a high- pass filter with $1-G(s)$. This should be suitable to suppress low frequency or bias effects within the signal x.

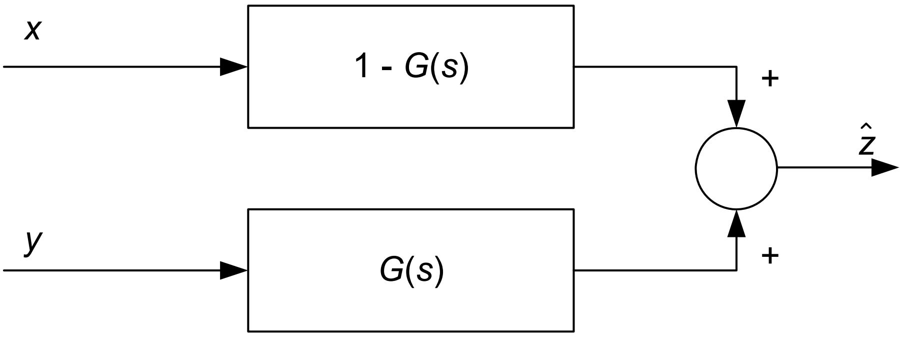


Figure 8. Principle of complementary filters

Kalman Filter

In 1960, R. E. Kalman published a paper describing a recursive solution to the discrete data linear filtering problem. Since that time, the Kalman filter has been subject to extensive investigations. The Kalman filter allows certain predictions within stochastic processes, by separation of noise and signal. It also allows the detection of signals with previously known shape (impulses, sine-waves etc). Fig. 9 shows the application of the filter in inertial measurement tasks. A necessary requirement of successful application is the knowledge of stochastic parameters of the measurement noise (covariance). For further reading, see e. g. Grewal, Weill and Andrews (2000).

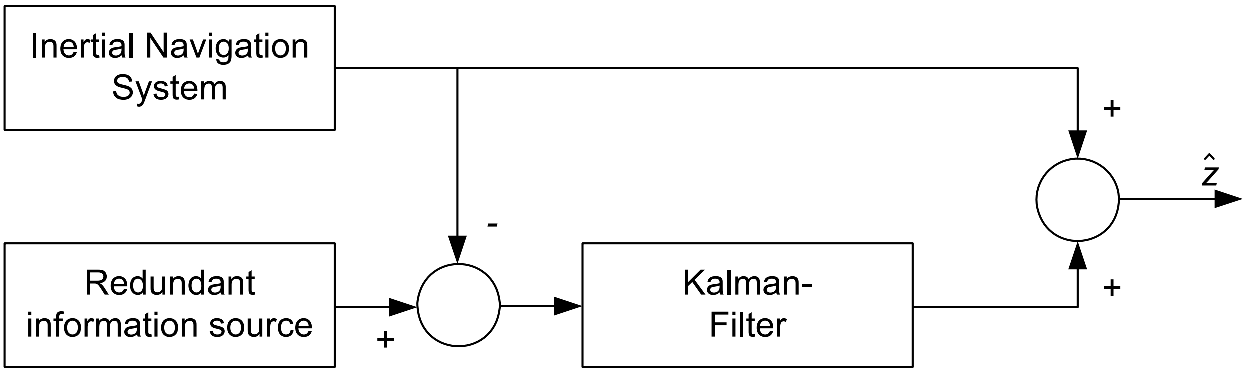


Figure 9. Kalman Filter application in Inertial Navigation

5. Simulation

A simulation with the simple two-dimensional sensor head arrangement from Fig. 2 has been introduced in Favre-Bulle (1993). The goal was to determine influence factors of the sensor head setup and the accelerometer performance parameters on the overall accuracy of the inertial trajectory measurement. A standard two-dimensional motion test pattern has been chosen (Fig. 10).

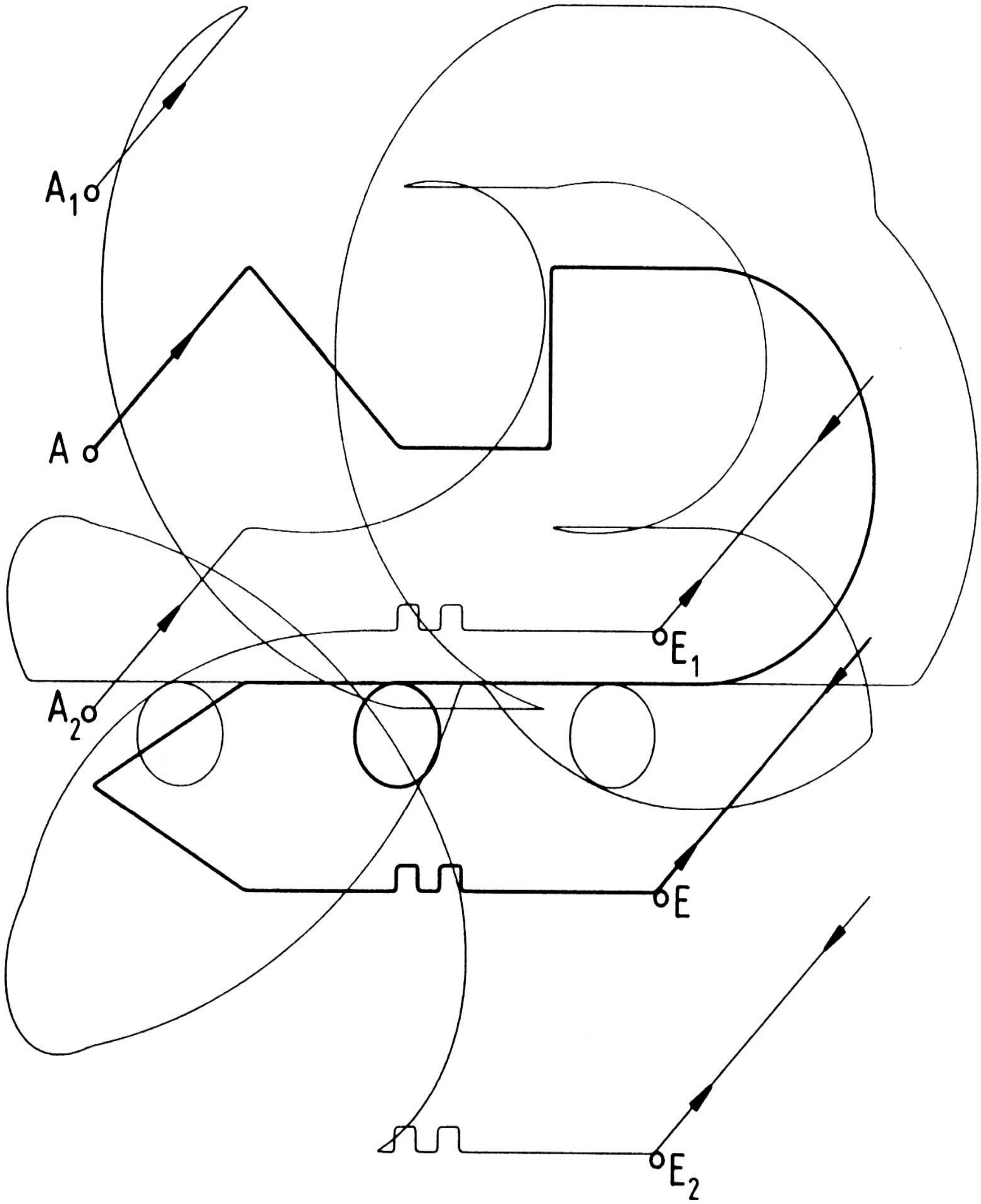


Figure 10. Test pattern for the simulation

This test pattern includes both translation and rotation parts and some combinations of it. Fig. 10 shows the end-effector path in bold line, and the traces of reference points of the three accelerometers. (A to E, A$_1$ to E$_1$ and A$_2$ to E$_2$).

A dynamic robot model (SCARA-Type) has been established, in order to provide realistic motion behavior of the tool-center-point. Fig. 11 shows the acceleration profiles $\ddot{x}$, $\ddot{y}$ and $\dot{\omega}$ for the first part of the pattern, as they occur at the location of the tool-tip.

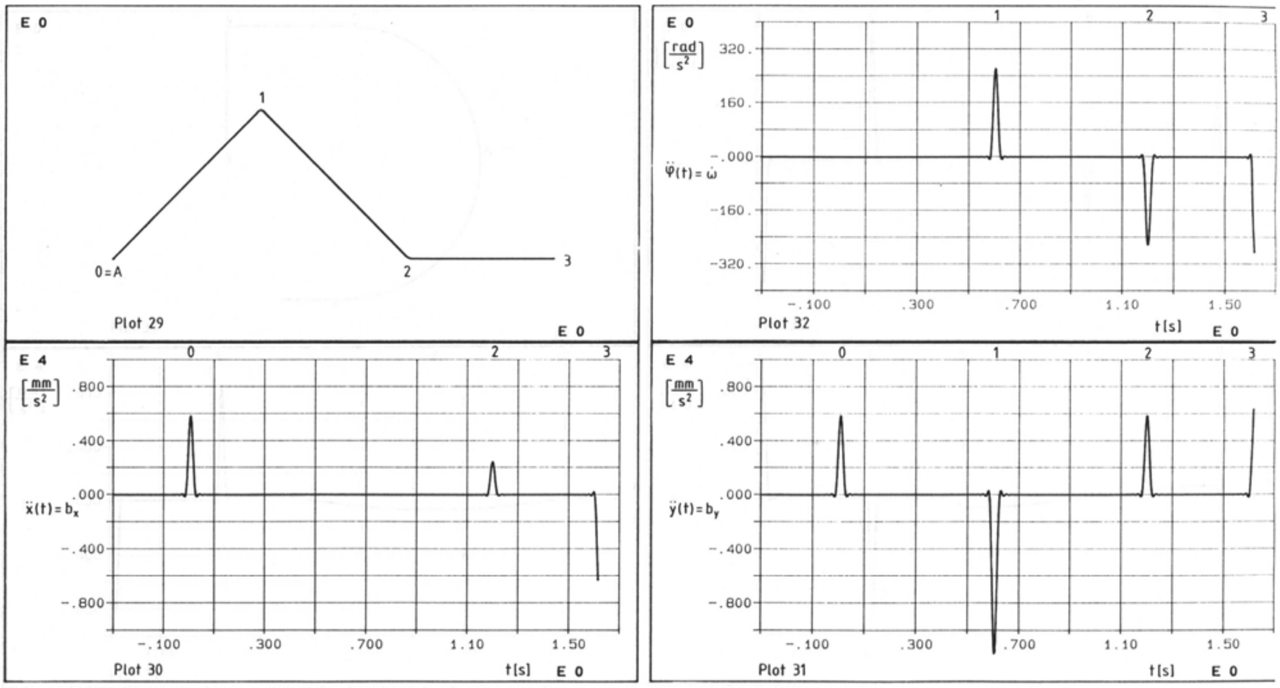


Figure 11. First part of the test trajectory and corresponding acceleration profiles

The linear servo accelerometers in the simulation setup have been modeled with respect to their dynamics, bias-error, scale-error, non-linearities of second and third order, cross-coupling, transversal sensitivity, angle-acceleration sensitivity and misalignment within the sensor head.

Fig. 12 shows the overall measurement error for increasing bias error ($0,2.10^{-3}$ g, $0.5.10^{-3}$g and 10^{-3} g) in the accelerometers. These bias errors have been chosen three magnitudes higher than the best possible errors with modern transducers in order to make the resulting effect better visible.

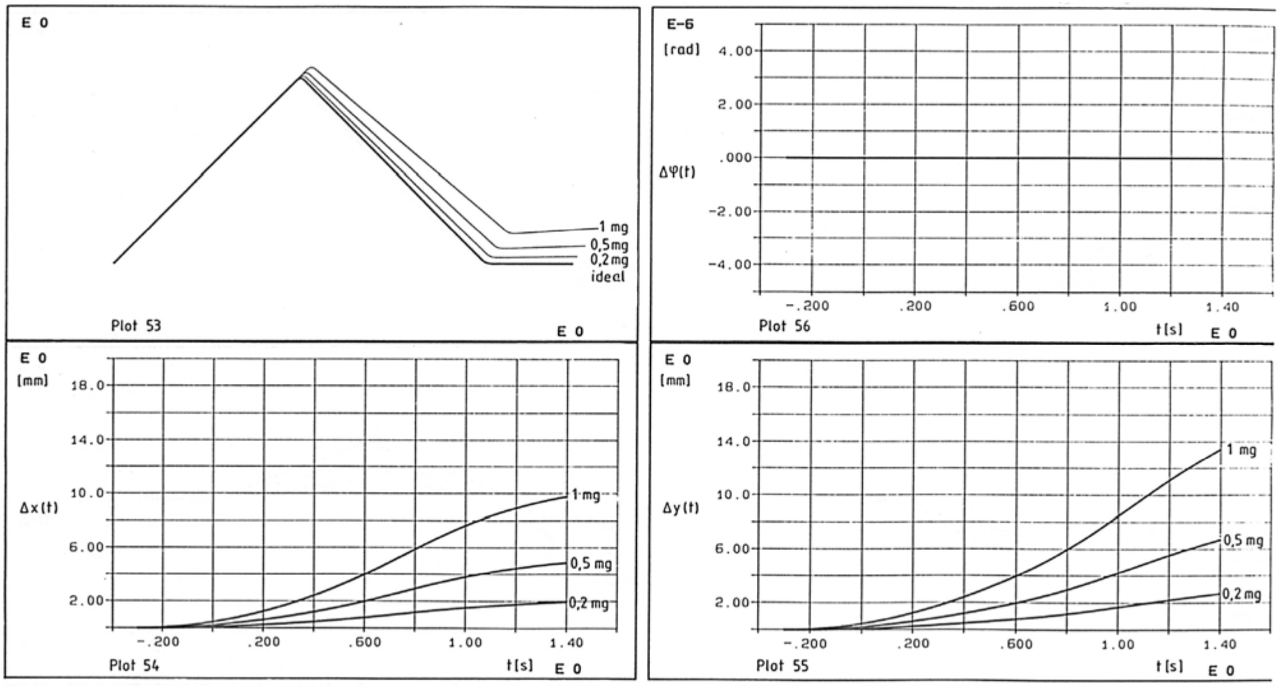


Figure 12. Overall measurement error due to bias errors

Another interesting result is the high sensitivity of the system performance on the angle-acceleration sensitivity of the transducers (Fig. 13).

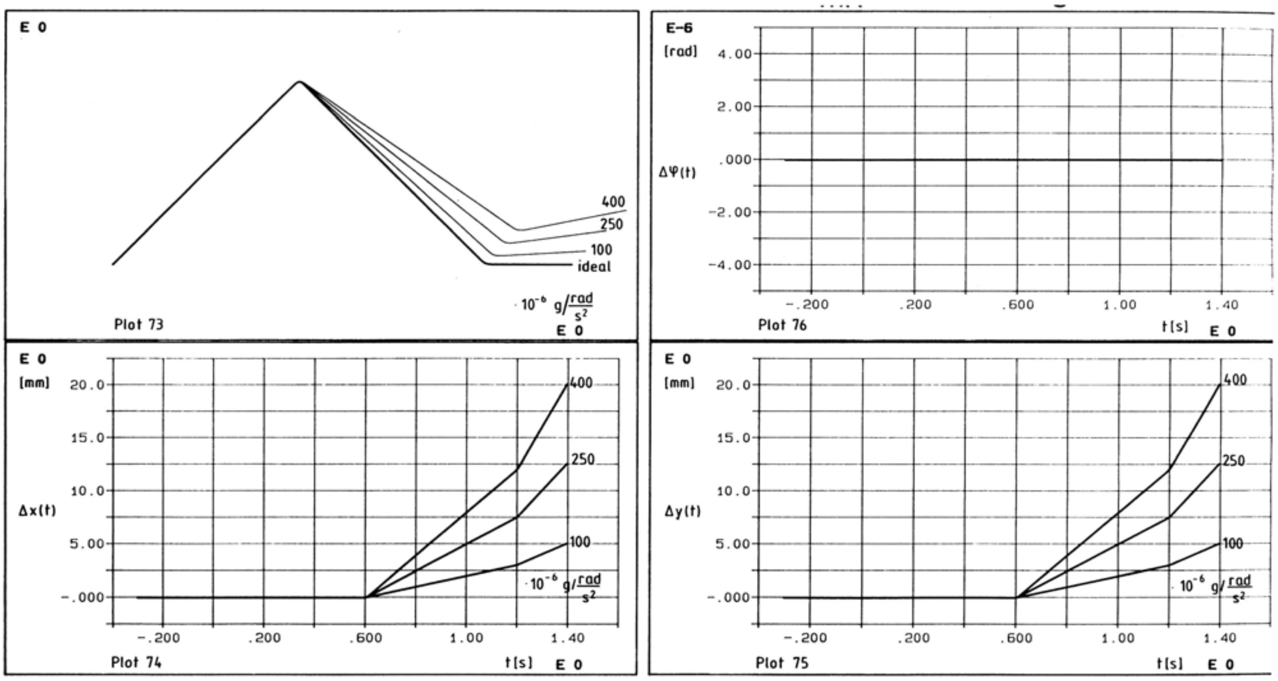


Figure 13. Performance errors of the INS due to angle-acceleration sensitivities

The reason for transducer sensitivity of several accelerometer models with respect to rotational acceleration lies in the fact that the measurement principle uses pendulum-like hinges to support the seismic mass. Thus, not only linear accelerations but also changes in rotation speed cause erroneous behavior.

6. Conclusion

The previous investigations have shown that it is possible to measure dynamic trajectory properties of industrial robots with gyro-less inertial navigation systems. As typical for INS, there are certain sources of errors which distort the overall analysis results to a certain extent. The proposed measurement principle is especially very sensitive to misalignment of the transducers in the sensor head and to bias errors of the accelerometers.

With redundant configurations (e. g. a higher number of transducers than spatial degrees of freedom) there are certain possibilities to improve the overall accuracy by means of complementary filters and Kalman filters.

Further research should investigate the use of low-cost gyroscopes as a source of additional motion information. One interesting "candidate" would be a piezoelectric vibration type gyro, like the Murata Gyrostrar®. Utilizing hybrid, highly-redundant sensor heads made from low-cost components could be a feasible solution for high-accurate trajectory analysis heads for affordable prices.

7. References

Albada, G. D. v., Lagerberg, A. and Visser, J. M. (1996) A low-cost pose-measuring system for robot calibration. Tech. Rep. University of Amsterdam, Faculty of Mathematics and Computer Science

Diewald, B. (1995) Über-alles-Kalibrierung von Industrierobotern zur lokalen Minimierung der Posefehler. PhD thesis, Universität des Saarlandes

Favre-Bulle, B. (1993) An Inertial Navigation System for Robot Measurement and Control. In: 2nd IEEE Conference on Control Applications, Sept. 13–16, Vancouver, B. C.

Grewal, M. S., Weill, L. R. and Andrews, P. (2000) Global Positioning Systems, Inertial Navigation, and Integration. Control Systems Technology & Automation. Wiley

Gustavson, T. L., Bouyer, P. and Kasevich M. A. (1997) Precision Rotation Measurements with an Atom Interferometer Gyroscope. Physical Review Letters, vol. 78 (11), pp. 2046–2049, Department of Physics, Stanford University, Stanford, California

Higgins, W., T. (1975) A Comparison of Complementary and Kalman Filtering. In: IEEE Transactions on Aerospace and Electronic Systems, Vol. AES–11, No. 3, pp. 321–325

Hinüber, E. v. (1993), Bahn- und Positionsvermessung von Industrierobotern mit inertialen Meßsystemen. PhD thesis, Universität des Saarlandes

Hinüber, E. v. and Janocha, H. (1993) Inertial Measurement System for Calibration and Control of Robots. In: Industrial Robots, Vol. 20 No. 2 (1993) pp. 20–27

ISO 9283 (1993) Manipulating Industrial Robots—Performance Criteria and Related Test Methods. International Standardisation Organisation

Janocha, H. and Hinüber, E. v. (1995) Leistungspotential moderner inertialer Meßsysteme. In: ATZ – Automobiltechn. Zeitschr., 97 (1995) , 1, 30–35

Leigh-Lancaster, C. J., Shirinzadeh and B. Koh, Y. L. (1997) Development of a Laser Tracking System. 4th Annual Conference on Mechatronics and Machine Vision in Practice (M2VIP '97), Sept. 23–09, 1997, Toowoomba, Australia

Roos, E., Behrens, A., Anton, S. (1997) RDS—realistic dynamic simulation of robots. In: 28th International Symposium on Robotics, Detroit USA

Schröder-Brzosniowsky, J., (1969) Fehleranalyse bei fahrzeugfesten Trägheitsnavigationssystemen. Berlin

Schröer, K., Albright, S. L. and Grethlein, M. (1997) Complete, minimal and model-continuous kinematic models for robot calibration. In: Robotics and Computer-Integrated Manufacturing, vol. 12, no. 1, pp. 73–85

Shirinzadeh, B. (1998) Laser-interferometry-based tracking for dynamic measurements. Industrial Robot: An International Journal, Vol. 25, No. 1, pp. 35–41, 1998.

Spiess, S., Vincze, M. and Ayromlu, M. (1997) On the calibration of a 6-D laser tracking system for contactless, dynamic robot measurements. In: Proceedings of the IEEE Instrumentation & Measurement Technology Conference, IMTC, Ottawa, pp. 1203–1208

Wrigley, W., Wrodbary, R., Hovorka, I. (1957) Inertial Guidance. Preprint Nr. 698, Institute of the Aeronautical Science, New York

Supervisory Controller for Task Assignment and Resource Dispatching in Mobile Wireless Sensor Networks

Vincenzo Giordano, Frank Lewis, Prasanna Ballal & Biagio Turchiano

1. Introduction

Wireless sensor networks are one of the first real-world examples of pervasive computing [1, 14, 24]. Small, smart, and cheap sensing devices will eventually permeate a certain environment and, suitably coordinated, will automatically recognize the present context and accordingly readjust their behavior. Smart environments represent the next evolutionary development step in building, utilities, industrial, home, shipboard, and transportation systems automation. Although this technology is still in its early days, the range of potential applications is mind-boggling – Robotics, health monitoring, defense systems, habitat monitoring etc. Sensor networks would greatly help monitor the environment and detect occurrences of natural calamities. For example, sensor networks that measure seismic activity from remote locations and provides tsunami warnings to coastal areas. However they present a range of challenges as they are closely coupled to the physical world with all its unpredictable variation, noise, and asynchrony; they involve many energy-constrained, resource-limited devices operating in concert; they must be largely self-organizing, adaptable to different environmental sensing applications and robust to sensor losses and failures.

To meet these challenges, recently there has been increased research interest in systems composed of autonomous mobile robotic sensors exhibiting cooperative behavior [13]. Sensor mobility, in fact, holds out the hope to support self-configuration mechanisms [5], guaranteeing adaptability, scalability and optimal performance, since the best network configuration is usually time-varying and context dependent. An example of mobile wireless sensor network is a warehouse guarded by mobile robots constantly interacting with strategically placed ground sensors to keep the risk of fire, robbery, etc to a minimum, while reducing the personnel costs.

In related literature, effective coordination strategies have been proposed to coordinate mobile robotic sensing units cooperating for a common goal. In decentralized coordination approach, the robots rely only on information about their local neighbors and the behavior of the overall network of robots emerges from the interactions of single units. In [4] and in [7] decentralized coordination algorithms for robots teams are proposed, to guarantee minimum exploration time and complete coverage respectively. In [10] a potential field approach to reach uniform deployment of sensors in an unstructured environment is proposed. It is implicitly assumed that the network cannot be configured for different missions apart from blanket coverage. [2, 3] propose behavior based formation control for exploration purposes, but no specific performances can be guaranteed and changes in the mission plans are not straightforward. In all the before mentioned decentralized approaches, robots possess similar functionalities, perform similar tasks and just one mission at a time is usually implemented.

To overcome the inherent limitations of decentralized approaches, supervisory (centralized) control techniques have been studied. In [8], a supervisory controller is proposed which reschedules the mission planning in response to uncontrollable events (node failures) using computationally efficient algorithms. Also the use of a centralized coordinator can ensure that the group possesses certain desired properties and remains within the bounds of pre-specified behavioral constraints. Some significant results in supervisory control have also been obtained using Petri nets [11]. Nevertheless the implementation of high-level mission specifications is not straightforward, the dynamical description of the system is incomplete and a new design stage, almost from scratch, is required if objectives or resources change. Summarizing, in related literature, there is a lack of supervisory control techniques which can sequence different missions according to the scenario (adaptability) and reformulate the missions if some of the robots fail (fault tolerance) in a predictable way and using a high-level interface.

In this chapter we present a discrete-event controller (DEC) for the centralized coordination of cooperating heterogeneous wireless sensors, namely unattended ground sensors (UGSs) and mobile robots. The DEC sequences the most suitable tasks for each agent according to the current perception of the environment. Priority rules for efficiently dispatching shared resources and handling simultaneous missions can also be easily taken into account. A novel and easy to implement matrix formulation makes the assignment of the mission planning straightforward and easily adaptable if agents or applications change. It represents a complete dynamical description which allows one to simulate and implement the system with the same controller software, simplifying the implementation of a mobile WSN in real world scenarios (e.g. when hostile terrains and huge areas have to be monitored).

2. Discrete Event Controller (DEC)

This section presents a matrix-based discrete event controller for modeling and analysis of complex interconnected DE systems with shared resources and dynamic resource management. Its matrix formulation gives a very direct and efficient technique for both computer simulation and actual on-line supervisory control of DE systems. It provides a rigorous, yet intuitive mathematical framework to represent the dynamic evolution of DE systems according to linguistic *if-then* rules:

Rule i: If <conditions^i hold > *then* <consequences^i>

For coordination problems of multi-agent systems (e.g. mobile robots and wireless sensors), we can write down a set of if-then rules to define the mission planning of the sensor agents, such as:

Rule i: If <sensor1 has completed task1, robot2 is available and a chemical alert is detected > *then* <robot 2 starts task4 and sensor 1 is released>

These linguistic rules can be easily represented in mathematical form using matrices. Let r be the vector of resources used in the system (e.g. mobile robots and UGSs), v the vector of tasks that the resources can perform (e.g. *go to a prescribed location*, *take a measurement*, *retrieve and deploy UGS*), u the vector of input events (occurrence of sensor detection events, node failures, etc.) and y the vector of completed missions (outputs). Finally, let x

be the state logical vector of the rules of the DE controller, whose entry of '1' in position *i* denotes that rule *i* of the supervisory control policy is currently activated.

Then we can define two different sets of logical equations, one for checking the conditions for the activation of rule *i* (matrix controller state equation), and one for defining the consequences of the activation of rule *i* (matrix controller output equation). In the following, all matrix operations are defined to be in the or/and algebra, where + denotes logical or and 'times' denotes logical and.

The controller state equation is

$$\overline{\mathrm{x}} = \mathrm{F_v}\overline{\mathrm{v}} + \mathrm{F_r}\overline{\mathrm{r}} + \mathrm{F_u}\overline{\mathrm{u}} + \mathrm{F_{ud}}\overline{\mathrm{u}}_\mathrm{d} \tag{1}$$

where x is the task or state logical vector, $\mathrm{F_v}$ is the task sequencing matrix, $\mathrm{F_r}$ is the resource requirements matrix, $\mathrm{F_u}$ is the input matrix. $\mathrm{F_{ud}}$ is the conflict resolution matrix and $\mathrm{u_d}$ is the conflict resolution vector. They are used to avoid simultaneous activation of conflicting rules, as will be shown later. The current status of the DE system includes task vector *v*, whose entries of `1' represent `completed tasks', resource vector *r*, whose entries of `1' represent `resources currently available', and the input vector *u*, whose entry of 1 represent occurrence of a certain predefined event (fire alarm, intrusion etc.). The overbar in equation (1) denotes logical negation so that tasks complete or resources released are represented by '0' entries.

F_v is the task sequencing matrix (used by Steward [22] and others in manufacturing), and has element *(i,j)* set to '1' if the completion of task v_j is an immediate prerequisite for the activation of logic state x_i.

F_r is the resource requirements matrix (used by Kusiak [12] and others in manufacturing) and has element *(i,j)* set to '1' if the availability of resource *j* (robot or UGS) is an immediate prerequisite for the activation of logic state x_i.

On the ground of the current status of the DE system, equation 1 calculates the logical vector *x*, i.e. which rules are currently activated. The activated rules determine the commands that the DEC has to sequence in the next iteration, according to the following equations

$$\mathrm{v_s} = \mathrm{S_v}\mathrm{x} \tag{2}$$

$$\mathrm{r_s} = \mathrm{S_r}\mathrm{x} \tag{3}$$

$$\mathrm{y} = \mathrm{S_y}\mathrm{x} \tag{4}$$

S_v is the task start matrix and has element *(i,j)* set to '1' if logic state x_j determines the activation of task *i*.

S_r is the resource release matrix and has element *(i,j)* set to '1' if the activation of logic state x_j determines the release of resource *i*.

S_y is the output matrix and has element *(i,j)* set to '1' if the activation of logic state x_j determines the completion of mission *i*.

The task start equation (2) computes which tasks are activated and may be started, the resource release equation (3) computes which resources should be released (due to completed tasks) and the mission completion equation (4) computes which missions have been successfully completed.

Vector v_s, whose `1' entries denote which tasks are to be started, and vector $r_s$, whose `1' entries denote which resources are to be released, represent the commands sent to the DE system by the controller. ‘1’ entries in vector y denote which missions have been successfully completed.

Equations 1-4 represent the rule-base of the supervisory control of the DE system. All the coefficient matrices are composed of Boolean elements and are sparse, so that real time computations are easy even for large interconnected DE systems.

The task sequencing matrices (F_v and S_v) are direct to write down from the required operational task sequencing. On the other hand, the resource requirements matrices (F_r, S_r) are written down based on the resources needed to perform the tasks and are assigned independently of the task sequencing matrices.

Given the presence of shared resources, simultaneous activation of conflicting rules may arise. Matrix F_{ud} in equation (1) is used to resolve conflicts of shared resources, i.e. conflicts deriving by the simultaneous activation of rules which start different tasks requiring the same resource. Matrix F_{ud} has as many columns as the number of tasks performed by shared resources. Element *(i,j)* is set to '1' if completion of shared task j is an immediate prerequisite for the activation of logic state x_i. Then an entry of ‘1’ in position j in the conflict resolution vector u_d, determines the inhibition of logic state x_i (rule i cannot be fired). It results that, depending on the way one selects the conflict-resolution strategy to generate vector u_d, different dispatching strategies can be selected, in order to avoid resource conflicts or deadlocks.

In the conversion of linguistic rules into the matrix formulation of the DEC, the following assumptions must be considered:

1. A resource cannot be removed from a task until it is complete.
2. A single resource can be used for only one task at a time.
3. A process holds the resource allocated to it until it has all the resources required to perform a task.
4. Resource is released immediately after it has executed its task.

3. Control Architecture

To use the DEC as a Supervisory Controller for task assignment and resource dispatching in mobile wireless sensor networks, we need to have an architecture that is modular, flexible and adaptable. It consists of three layers [20] with distinct functionalities, namely agent control layer, network control layer and supervisor control layer (figure 1). In this way improvements and updates on one layer results in minor changes in the other layers.

The first layer (agent control level) deals with the control of each agent (being either a UGS or a mobile robot), keeping into account its peculiar functionalities. At this level we define the processing capabilities of the UGSs (e.g. signal processing) and the control algorithms for the behavior of each robot (e.g. reach the target, follow another robot etc.).

The second layer (network control level) deals with the implementation of communication protocols for energy efficient data transmission between the UGS, robots and the supervisor.

The third layer (supervisor control level) consists of the matrix-based DE supervisory controller whose matrix formulation allows one to employ a high-level human interface to define the mission planning, the resource allocation and the dispatching rules. The supervisor is in charge of sequencing the tasks each agent has to perform according to the perception of the environment, assuming that the agent level controllers correctly perform the assigned tasks and that the communication protocol for each agent perfectly works. The feedback information about the evolution of the scenario is provided by the “sensor to context mapping” modules (e.g. see [25]), constituted by sensor fusion and decision making algorithms implemented on some (or all) of the agents. In order to keep track of the dynamic behavior of the network, the supervisor also receives notification from each agent about the completed tasks and the released resources.

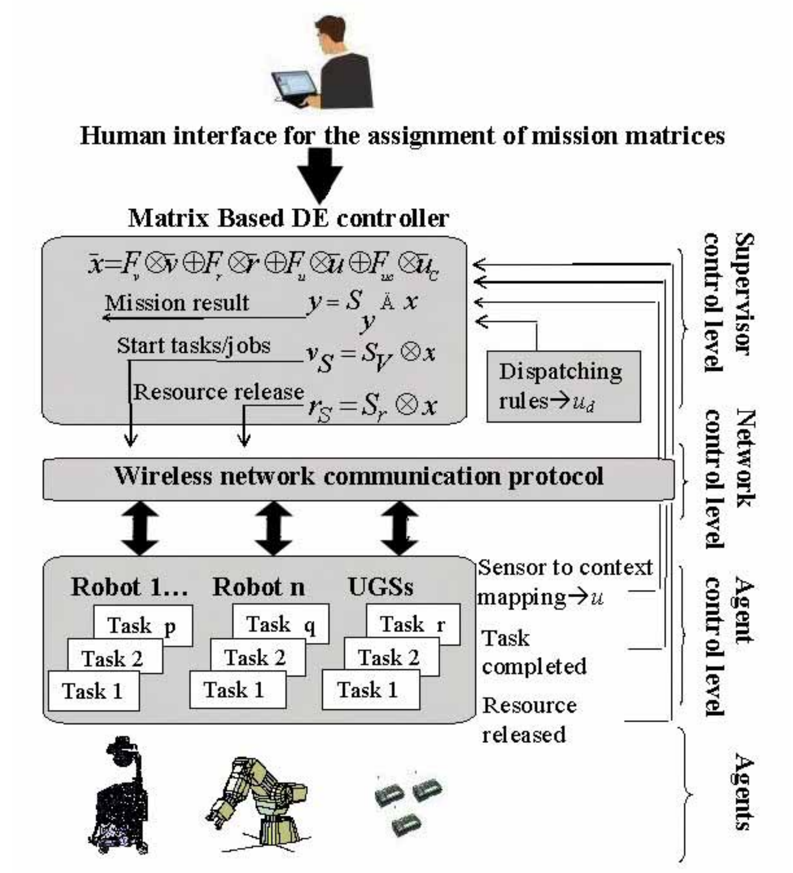
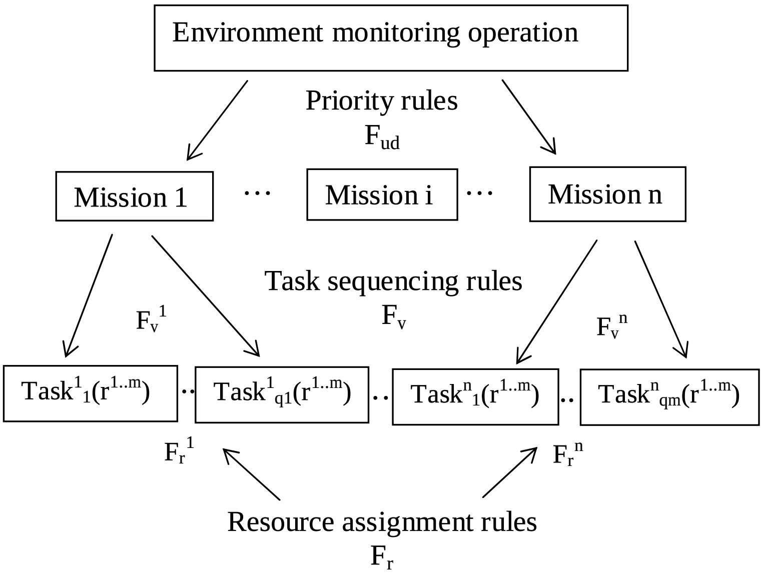


Figure 1. Complete control architecture

Thus, using this architecture, a complex system can be decomposed into missions, tasks and rules for task sequencing, resource dispatching and conflict resolution (figure 2a).

This block diagram can be represented using block matrices in the framework of our DEC. Suppose that we have *m* resources *r^j^* *j=1…m* (mobile robots and stationary sensors) each one capable of performing p_j tasks, and define *n* different missions, each one composed of q_i tasks. For each mission, we define the corresponding set of matrices F_v^i, F_r^i, S_v^i, S_r^i which represent the coordination rules of the agents in the execution of the tasks. In order to take into account the priority among missions, we derive the global conflict resolution matrix F_{ud} according to the following procedure. After assigning a priority order *k* to each mission, we calculate, for every resource *j* and every mission *i*, a matrix ($F_{ud}^i\left(r_j\right)$), creating a new column for every '1' appearing in the *jth* column of F_r^i. Then we construct the global conflict resolution matrix of resource r_j ($F_{ud}\left(r_j\right)$) inserting each $F_{ud}^i\left(r_j\right)$ matrix in position *(i,k)*.

As shown in figure 2b, the matrix formulation of the overall environment monitoring operation is then obtained by stacking the set of matrices together. The correspondence between figure 2a and 2b is straightforward.



(a)

$$F_{ud} = \left[F_{ud}(r_1)\ldots F_{ud}(r_j)\ldots F_{ud}(r_m)\right]$$

$$\begin{array}{cc} & \begin{array}{ccccc} q_1 & \ldots & q_i & \ldots & q_n \end{array} \\ F_v = & \begin{bmatrix} F_v^1 & & & & \\ & \ldots & & & \\ & & F_v^i & & \\ & & & \ldots & \\ & & & & F_v^n \end{bmatrix} \end{array} \quad \begin{array}{cc} & m \\ F_r = & \begin{bmatrix} F_r^1 \\ \ldots \\ F_r^i \\ \ldots \\ F_r^n \end{bmatrix} \end{array}$$

(b)

Figure 2. Decomposition of the environment monitoring operation (a) and its matrix representation (b)

4. Dec for Mobile Wireless Sensor Networks

Differently from manufacturing systems where discrete event controllers have been already successfully applied, the implementation of a DEC for a mobile WSN is indeed much more challenging. In fact a WSN has in general a variable topology, is composed of heterogeneous resources and operates in highly unstructured environments. An efficient centralized control policy has to guarantee efficient and automatic responses to changes in the operating conditions (adaptability) and to changes in the dimension of the mobile WSN (scalability). In particular, to accomplish mission goals and to guarantee optimal performances, the supervisory controller must be able to automatically reschedule missions, reassign resources to tasks and redefine the mission priorities as long as the WSN topology and the surrounding environment evolve. In the following, we will show how these issues can be efficiently tackled using our matrix-based DEC.

4.1 Adaptability

Adaptability is the ability of an agent team to change its behavior according to the dynamical evolution of the environment. In the following, we provide some ideas to make the system adaptable using our DEC.

Implementation of distributed algorithms

In related literature, coordination of teams of robots through the implementation of distributed algorithm is a common strategy, because it guarantees robustness to environment uncertainties and disturbances. Every robotic agent performs a certain task relying on local information only (e.g. keep a certain distance from the neighbors), in such a way that a predefined aggregate objective (e.g. complete environment coverage) is achieved. In the framework of the DEC, these operations can be considered as a generic (fully decentralized) mission *i* (or part of it) with a certain goal composed of simultaneous tasks. Enhanced adaptability can be obtained deciding, at the supervisor level (on the ground of the present situation), which decentralized mission (optimal sensor placement for environmental monitoring, search and recovery operations etc.) has the priority (changing F_{ud}^{i}) and which resources should be used (changing F_r^i and S_r^i).

Combining multiple plans for the same mission

In certain circumstances, different sequences of tasks can be used to implement the same mission. A computationally efficient algorithm has been recently proposed to combine the plans together and derive one single compact matrix representation for the DEC [9]. In this way, the DEC automatically sequences the most suitable succession of tasks to achieve a certain goal, depending on the current available resources.

Dynamic reallocation of resources

A dynamic reallocation of the agents to missions can be performed by rearranging the '1' relative to similar resources in the matrices F_r and S_r when new missions (or new agents) are added. Due to the matrix representation of the mission plans, these objectives can be pursued using computationally efficient algorithms.

In figure 3 we have reported an example of reallocation of resources (*r*1, *r*2 and *r*3) to the tasks of two missions. The number of tasks each resource has to perform is equal to the number of '1' in the corresponding column. After the reallocation, the resources have a more balanced workload, i.e. they perform a similar number of tasks.

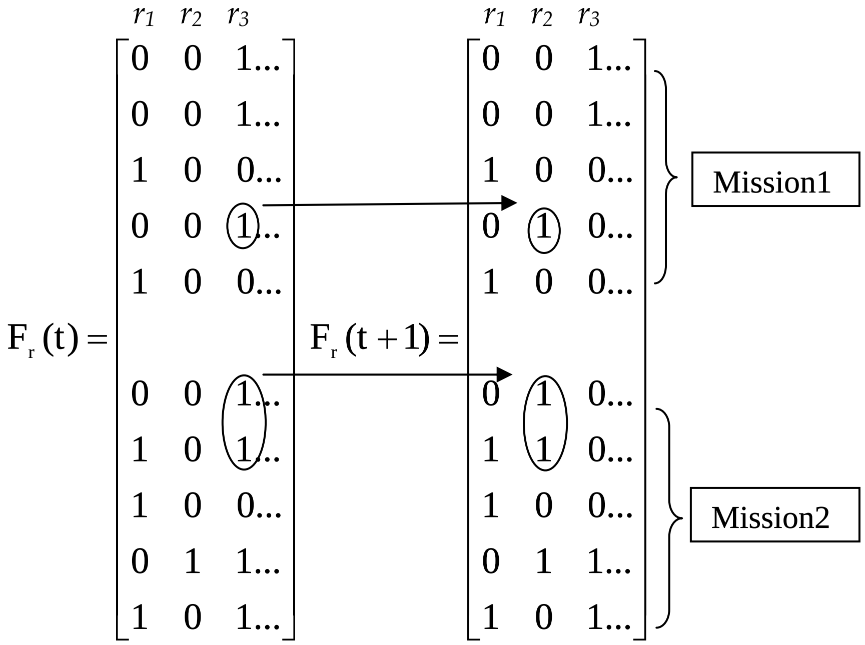


Figure 3. Reallocation of resources through matrix operations

Priority among missions

In order to further tailor the WSN to the present control scenario, the set of mission priority rules can be made adaptable.

For example, suppose that resource *r1* is shared among three different missions whose priority rank is *3, 1, 2*. After defining the conflict resolution matrix of *r1* for each mission ($F_{ud}^{1}(r_1)$, $F_{ud}^{2}(r_1)$, $F_{ud}^{3}(r_1)$), the overall conflict resolution matrix of *r1* ($F_{ud}(r_1)$) is built as

$$F_{ud}(r_1) = \begin{array}{c} \\ \text{mission}^1 \\ \text{mission}^2 \\ \text{mission}^3 \end{array} \begin{array}{c} \begin{array}{ccc} \textit{priority1} & \textit{priority2} & \textit{priority3} \end{array} \\ \begin{bmatrix} 0 & F_{ud}^{1}(r1) & 0 \\ 0 & 0 & F_{ud}^{2}(r1) \\ F_{ud}^{3}(r1) & 0 & 0 \end{bmatrix} \end{array}$$

If the priority of the missions change in *2, 3, 1* then we have

$$F_{ud}(r_1) = \begin{array}{c} \\ \text{mission}^1 \\ \text{mission}^2 \\ \text{mission}^3 \end{array} \begin{array}{c} \begin{array}{ccc} \textit{priority1} & \textit{priority2} & \textit{priority3} \end{array} \\ \begin{bmatrix} 0 & 0 & F_{ud}^{1}(r_1) \\ F_{ud}^{2}(r_1) & 0 & 0 \\ 0 & F_{ud}^{3}(r_1) & 0 \end{bmatrix} \end{array}$$

Thus, a change of priority results in a simple permutation of the block matrices F_{ud}^{i} for each resource.

4.2 Scalability

Scalability is the ability of an agent team to reorganize its overall behavior in response to a change in the number of the agents. We can use the DEC to tackle scalability at the supervisor level, updating the matrix based representation of the missions to take into account the failure of agents as well as the adding of new ones.

If a new agent is added to the system, a new column is added in the matrices F_r and $S_r^{'}$ (S_r transpose). Then, dispatching algorithms (based on matrix operations) can be applied to rearrange the tasks among resources. In a similar fashion, an agent failure can be tackled rearranging the tasks among the resources so that the columns relative to the failed resources in F_r and S_r' are null. Just to give an idea, in the following example, we describe a simple algorithm for reallocating (off-line, i.e. when no missions are in progress) resources after agent failure. The mission planning is revised in such a way that predefined back-up agents execute the tasks of the failed agents. In the matrix formulation this is equivalent to move the elements equal to one in the matrices F_r^{i} and S_r^{i} from the column of the failed resource to the column of the back-up resource. This can be achieved through a simple linear combination of the columns of F_r^{i} and S_r^{i} respectively. We have

$$F_r^{i,new} = F_r^{i,old} \cdot B^{i}$$
$$S_r^{i,new} = S_r^{i,old} \cdot B^{i}$$

where B^i is a square matrix of dimension equal to the number of the resources of the system. The diagonal elements of B^i (a_j) are parameters which are equal to 1 if resource r_j is working properly and 0 otherwise. On row *j* the element *(j,j)* is equal to a_j and the element *(j,k)* is equal to *1-* a_j, where *k* is the column of matrix F_r^i corresponding to the back-up resource of agent *j* for mission *i*. If a_j =0, the *jth* column of $F_r^{i,new}$ will be null (meaning that agent *j* is not supposed to perform any task) and the *kth* column will have '1's in correspondence of the tasks for which resource *j* was required. If no failure occurs, B^i is the identity matrix and mission plans are not changed. Clearly, in the definition of the matrix B^i we have to make sure that each back-up agent does not perform any simultaneous task with the resource they are supposed to substitute. For example, suppose we have three agents, and that, for a certain mission *i,* the resource requirement matrix and the back-up matrix B^i are

$$F_r^{i,old} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} \quad B^i = \begin{pmatrix} a_1 & 1-a_1 & 0 \\ 0 & a_2 & 0 \\ 1-a_3 & 0 & a_3 \end{pmatrix}$$

The B^i matrix corresponds to the case where, in mission *i*, agent 2 is the backup of agent 1, agent 2 has no back-up and agent 1 is the back up of agent 3. If agent 1 fails (a_1=0) whereas agent 2 and 3 work properly (a_2=a_3=1), we get

$$B^i = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad F_r^{i,new} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

i.e., in mission *i*, agent 1 has been replaced by agent 2.

A more effective alternative to cope with agent failure is to introduce routing resources [17], which allow one to define a priori a set of multiple resource choices for certain critical tasks. The routing resources automatically assign to the task the first available resource of the corresponding set providing redundancy and robustness against agent failures. In the case of WSN, they are logical, fictitious resources and can be considered as local dispatchers. Our matrix formulation supports task routing efficiently, since there is no need to distinguish between physical and logical resources.

4.3 Complete Dynamical Description

It is well known that a matrix approach can be used to describe the marking transitions of a Petri Net [18] using the PN transition equation

$$m(t+1) = m(t) + (S - F') \cdot x(t) \tag{5}$$

where *S* and *F* are the output and input incidence matrix respectively. This equation gives a useful insight on the dynamics of discrete event systems but does not provide a complete dynamical description of DE systems.

If we observe that the vector *x* in equation (5) is the same as in equation (1), then we may identify *x* as the vector associated with the PN transitions and *u, v, r,* u_d as associated with the places. Then it follows that [23]

$$m(t) = [u(t)', v(t)', r(t)', u_d(t)']'$$
$$S = [S_u', S_v', S_r', S_{u_d}', S_y']$$
$$F = [F_u', F_v', F_r', F_{u_d}', F_y']$$

Therefore, we can use equation (1) to generate the allowable firing vector to trigger transitions in equation (5). That is, the combination of the DEC (1) and the PN marking transition equation provides a complete dynamical description of the system.

In order to take into account the time durations of the tasks and the time required for resource releases, we can split *m(t)* into two vectors, one representing available resources and current finished tasks ($m_a(t)$) and the other representing the tasks in progress and idle resources ($m_p(t)$)

$$m(t) = m_a(t) + m_p(t) \tag{6}$$

This is equivalent to introducing timed places in a Petri net and to dividing each place into two parts, one relative to the pending states (task in progress, resource idle) and the other relative to the steady states (task completed and resource available). As a consequence, we can also split equation (5) into two equations

$$m_a(t+1) = m_a(t) - F' \cdot x(t) \tag{7}$$
$$m_p(t+1) = m_p(t) + S \cdot x(t) \tag{8}$$

When a transition fires a token is moved from $m_p(t)$ to $m_a(t)$ where it may be used to fire subsequent transitions. Therefore equations (1), (7) and (8) represent a complete description of the dynamical behavior of the discrete event system and can be implemented for the purposes of computer simulations using any programming language (e.g. Matlab® or C). In the case of a mobile wireless sensor network, where experiments on wide and hostile areas can be really complex and challenging, it allows one to perform extensive simulations of the control strategies and then test experimentally only those which guarantee the most promising results.

A network consisting of two mobile robots and five wireless sensors is considered as experimental scenario. Two different missions have been implemented to show the potentialities of the proposed DEC. In the first mission, after one of the sensors launches a chemical alert, the network automatically reconfigures its topology to further investigate the phenomenon. In the second mission, considering that power constraints are a very serious concern in a WSN, one of the mobile robots is employed to charge the batteries of one of the UGSs.

The procedure for implementing the supervisory control policy consists of three different steps.

First of all we define the vector of resources *r* present in the system and the tasks they can perform. In our test-bed we have two robots (R_1 and R_2), each one able of performing certain number of tasks, and five stationary sensors (UGS_1, UGS_2, UGS_3, UGS_4, UGS_5), each one able of performing one task (i.e. taking measurement). The resource vector is $r=[R_1, R_2, UGS_1, UGS_2, UGS_3, UGS_4, UGS_5]$.

Then for each mission *i*, we define the vector of inputs u^i, of outputs y^i and of tasks v^i, and the task sequence of each mission (refer table I and II for mission 1 and mission 2), and write down the if-then rules representing the supervisory coordination strategy to sequence the programmed missions (table III and table IV). In the definition of the rule bases particular attention has to be devoted to the definition of consecutive tasks performed by the same resources. If the consecutive tasks are interdependent (e.g. *go to sensor 2* and *retrieve sensor2*), the corresponding resource should be released just at the end of the last of the consecutive tasks. This is the case of the task groups (*R1gS2, R1rS2*) and (*R1gS1, R1dS2, R1m*) in mission1 and of (*R1gS3* and *R1cS3***)** in mission2. Instead, if the tasks are not interdependent, before starting the new consecutive task, the DEC releases the corresponding resource and makes sure that no other missions are waiting for it. If after a predetermined period of time no other missions request that resource, the previous mission can continue. This is the function of the *"Robot 1 listens for interrupts*" task in mission 1.

Finally we translate the linguistic description of the coordination rules into a more convenient matrix representation, suitable for mathematical analysis and computer implementation.

As an example, in the following we derive the matrix formulation of mission 1 from the rule-base reported in table III. We can easily write down the F_v^1 and F_r^1 matrices considering that $F_v^1(i, j)$ is '1' if task *j* is required as an immediate precursor to rule *i* and $F_r^1(i, j)$ is '1' if resource *j* is required as an immediate precursor to rule *i* (see figure 4). For example, the conditions for firing rule 6 (x_6^1), are the completion of tasks *R2m*^{1} and *R1gS1*^{1} (task 7 and 8 respectively, see table I). We therefore have two '1' entries in position (6,7) and (6,8) in matrix F_v^1. Resource 1 is required for the execution of the two macro-tasks defined previously, which start when rules 2 and 5 respectively are triggered. Therefore, as shown in figure 4b, we have two '1' in position (2,1) and (5,1) of F_r^1.

An analysis of matrix F_r^1 reveals that only robot 1 and robot 2 are shared resources (due to multiple '1's in the corresponding column of F_r^1), therefore we need to calculate just $F_{ud}^1(R1)$ and $F_{ud}^1(R2)$ (figure 4c).

mission1	notation	description
Input 1	*u*^{1}	*UGS1* launches chemical alert
Task 1	*S4m*^{1}	*UGS4* takes measurement
Task 2	*S5m*^{1}	*UGS5* takes measurement
Task 3	*R1gS2*^{1}	*R1* goes to *UGS2*
Task 4	*R2gA*^{1}	*R2* goes to location A
Task 5	*R1rS2*^{1}	*R1* retrieves *UGS2*
Task 6	*R1lis*^{1}	*R1* listens for interrupts
Task 7	*R1gS1*^{1}	*R1* gores to *UGS1*
Task 8	*R2m*^{1}	*R2* takes measurement
Task 9	*R1dS2*^{1}	*R1* deploys *UGS2*
Task 10	*R1m*^{1}	*R1* takes measurement
Task 11	*S2m*^{1}	*S2* takes measurement
output	*y*^{1}	Mission 1 completed

Table 1. Mission1- Task sequence

Mission2	**notation**	**description**
input	u^2	*UGS3* batteries are low
Task 1	$S1m^2$	*UGS1* takes measurement
Task 2	$R1g\ S3^2$	*R1* goes to *UGS3*
Task 3	$R1cS3^2$	*R1* charges *UGS3*
Task 4	$S3m^2$	*UGS3* takes measurement
Task 5	$R1dC^2$	*R1* docks the charger
output	y^2	Mission 2 completed

Table 2. Mission 2-Task sequence

Mission1-operation sequence	
Rule1 x_1^1	If *u*[1]*occurs and S4 available and S5available then start S4m*[1] *and S5m*[1]
Rule2 x_2^1	If *S4m*[1] *and S5m*[1] *completed* and *R1 and R2 available* then *start R1gS2*[1] *and R2gA*[1] and *release S4 and S5*
Rule3 x_3^1	If *R1gs2*[1] *and R2gA*[1] *competed* then *start R1rS2*[1] and *release R2*
Rule4 x_4^1	If *R1rS2*[1] *completed* then *start R1lis*[1] and *release R1*
Rule5 x_5^1	If *R1lis*[1] *completed* and *R1 and R2 available* then *start R2m*[1] *and R1gS1*[1]
Rule6 x_6^1	If *R1gS1*[1] and *R2m*[1] *completed* then *start R1dS2*[1] and *release R2*
Rule7 x_7^1	If *R1dS2*[1] *completed* then *start R1m*[1]
Rule8 x_8^1	If *R1m*[1] *completed* and *S2 available* then *start S2m*[1] and *release R1*
Rule9 x_9^1	If *S2m*[1] *completed* then *release S2* and *terminate mission1 y*[1]

Table 3. Mission 1-Rule-base

The S_v^1 matrix (figure 5a) is built considering which tasks should be executed after a rule fires. For example, after robot1 listens for interrupts (this fires rule 5), R_1 should go to UGS_1 (*R1gS1*, task 7) and R_2 should start taking measurements with its onboard sensors (*R2m*, task 8). Accordingly, the elements of S_v^1 in position (7,5) and (8,5) are equal to 1.

The S_r^1 matrix (figure 5b) is built considering that $S_r^1(i,j)$ is 1 if resource *i* has to be released after rule *j* has been fired. For example, since the firing of rules 4 and 8 releases robot1 (resource1), we have entries of 1 in positions (1,4), and (1,8) in matrix S_r^1.

Mission2- operation sequence	
Rule1 x_1^2	If *u*^{2} *occurs* and *S1 available* then *start S1m*^{2}
Rule2 x_2^2	If *S1m*^{2} *completed* and *R1 available* then *start R1gS3*^{2} and *release S1*
Rule3 x_3^2	If *R1gS3*^{2} *completed* then *start R1cS3*^{2}
Rule4 x_4^2	If *R1cS3*^{2} *completed* and *S3 available* then *start S3m*^{2}and *release R1*
Rule5 x_5^2	If *S3m*^{2} *completed* and *R1 available* then *start R1dC*^{2} and *release S3*
Rule6 x_6^2	If *R1dC*^{2} *completed* then release *R1* and *terminate mission2 y*^{2}

Table 4. Mission 2-Rule-base

$$F_v^1 = \begin{array}{c|ccccccccccc|} & \text{S4m} & \text{S5m} & \text{R1gS2} & \text{R2gA} & \text{R1lis} & \text{R1rS2} & \text{R1gS1} & \text{R2m} & \text{R1dS2} & \text{S2m} & \text{R1m} \\ x_1^1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ x_2^1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ x_3^1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ x_4^1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ x_5^1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ x_6^1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ x_7^1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ x_8^1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ x_9^1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array} \text{(a)} \quad F_r^1 = \begin{array}{c|ccccccc|} & \text{R1} & \text{R2} & \text{S1} & \text{S2} & \text{S3} & \text{S4} & \text{S5} \\ x_1^1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ x_2^1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ x_3^1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ x_4^1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ x_5^1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ x_6^1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ x_7^1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ x_8^1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ x_9^1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array} \text{(b)}$$

$$F_{ud}^1(\text{R1}) = F_{ud}^1(\text{R2}) = \begin{array}{c|cc|} x_1^1 & 0 & 0 \\ x_2^1 & 1 & 0 \\ x_3^1 & 0 & 0 \\ x_4^1 & 0 & 0 \\ x_5^1 & 0 & 1 \\ x_6^1 & 0 & 0 \\ x_7^1 & 0 & 0 \\ x_8^1 & 0 & 0 \\ x_9^1 & 0 & 0 \end{array} \text{(c)}$$

Figure 4. Mission1- job sequencing matrix F_v^1 (a), resource requirement matrix F_r^1 (b) and conflict resolution matrix F_{ud}^1 (c)

$$S_v^1 = \begin{array}{c|ccccccccc|} & x_1^1 & x_2^1 & x_3^1 & x_4^1 & x_5^1 & x_6^1 & x_7^1 & x_8^1 & x_9^1 \\ S4m & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ S5m & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ R1gS2 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ R2gX & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ R1lis & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ R1rS2 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ R1gS1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ R2m & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ R1dS2 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ S2m & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ R1m & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{array} \text{(a)} \quad S_r^1 = \begin{array}{c|ccccccccc|} & x_1^1 & x_2^1 & x_3^1 & x_4^1 & x_5^1 & x_6^1 & x_7^1 & x_8^1 & x_9^1 \\ R1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ R2 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ S1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ S2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ S3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ S4 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ S5 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array} \text{(b)}$$

Figure 5. Mission1- Task start matrix S_v^1 (a) and resource release matrix S_r^1 (b)

$$F_v^2 = \begin{array}{c|ccccc|} & S1m & R1gS3 & R1cS3 & S3m & R1dC \\ x_1^2 & 0 & 0 & 0 & 0 & 0 \\ x_2^2 & 1 & 0 & 0 & 0 & 0 \\ x_3^2 & 0 & 1 & 0 & 0 & 0 \\ x_4^2 & 0 & 0 & 1 & 0 & 0 \\ x_5^2 & 0 & 0 & 0 & 1 & 0 \\ x_6^2 & 0 & 0 & 0 & 0 & 1 \end{array} \text{a)} \quad F_r^2 = \begin{array}{c|ccccccc|} & R1 & R2 & S1 & S2 & S3 & S4 & S5 \\ x_1^2 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ x_2^2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ x_3^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ x_4^2 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ x_5^2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ x_6^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array} \text{b)}$$

$$F_{ud}^2(R1) = \begin{array}{c|cc|} x_1^2 & 0 & 0 \\ x_2^2 & 1 & 0 \\ x_3^2 & 0 & 0 \\ x_4^2 & 0 & 1 \\ x_5^2 & 0 & 0 \\ x_6^2 & 0 & 0 \end{array} \text{c)}$$

Figure 6. Mission2- Task sequencing matrix F_v^2 (a), resource requirement matrix F_r^2 (b) and conflict resolution matrix $F_{ud}^2(R1)$ (c)

$$S_v^2 = \begin{array}{c|cccccc|} & x_1^2 & x_2^2 & x_3^2 & x_4^2 & x_5^2 & x_6^2 \\ S1m & 1 & 0 & 0 & 0 & 0 & 0 \\ R1gS3 & 0 & 1 & 0 & 0 & 0 & 0 \\ R1cS3 & 0 & 0 & 1 & 0 & 0 & 0 \\ S3m & 0 & 0 & 0 & 1 & 0 & 0 \\ R1dC & 0 & 0 & 0 & 0 & 1 & 0 \end{array} \text{a)} \quad S_r^2 = \begin{array}{c|cccccc|} & x_1^2 & x_2^2 & x_3^2 & x_4^2 & x_5^2 & x_6^2 \\ R1 & 0 & 0 & 0 & 1 & 0 & 1 \\ R2 & 0 & 0 & 0 & 0 & 0 & 0 \\ S1 & 0 & 1 & 0 & 0 & 0 & 0 \\ S2 & 0 & 0 & 0 & 0 & 0 & 0 \\ S3 & 0 & 0 & 0 & 0 & 1 & 0 \\ S4 & 0 & 0 & 0 & 0 & 0 & 0 \\ S5 & 0 & 0 & 0 & 0 & 0 & 0 \end{array} \text{b)}$$

Figure 7. Mission2- Task start matrix S_v^2 (a) and resource release matrix S_r^2 (b)

$$F_v = \begin{array}{c} x^1 \\ x^2 \end{array}\begin{pmatrix} F_v^1 & 0 \\ 0 & F_v^2 \end{pmatrix} \quad F_r = \begin{array}{c} x^1 \\ x^2 \end{array}\begin{pmatrix} F_r^1 \\ F_r^2 \end{pmatrix}$$

$$S_v = \begin{array}{c} v^1 \\ v^2 \end{array}\begin{pmatrix} S_v^1 & 0 \\ 0 & S_v^2 \end{pmatrix} \quad S_r = r\begin{pmatrix} S_r^1 & S_r^2 \end{pmatrix}$$

$$F_{ud} = \begin{bmatrix} \overbrace{0 \qquad F_{ud}^1(R_1)}^{R1} & \overbrace{F_{ud}^1(R_2)}^{R2} \\ F_{ud}^2(R_1) \qquad 0 & 0 \end{bmatrix}$$

Figure 8. Overall monitoring operation- Matrix formulation

In the same way, the set of matrices relative to mission 2 can be built (figure 6 and 7). Then the matrix formulation of the overall monitoring operation is easily obtained by stacking together the matrices of mission 1 and mission 2 (figure 8). Also, as stated before, this matrix formulation can be used to represent a Petri Net. Therefore, for the sake of clarity, in figure 9 we have reported the Petri Net which corresponds to the matrix description of the two implemented missions.

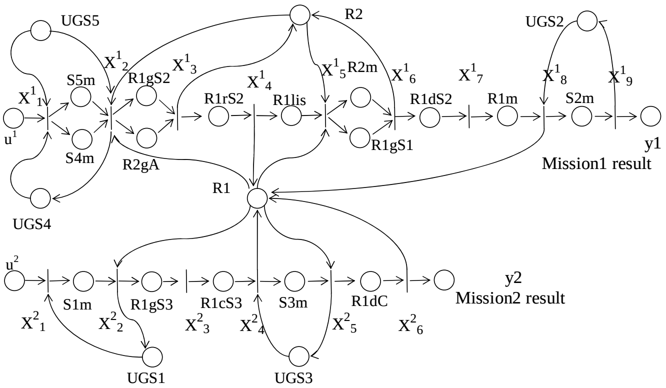


Figure 9. Petri net representation of the implemented missions, corresponding to the matrixes Fv, Fr, Sv, Sr

5. Simulation Results

Simulation of this system for the given missions using equations 1, 7 and 8 can be done using Matlab®. Figure 10, shows the utilization time trace of the resources and the execution time trace of the tasks. In these time traces, busy resources and tasks not in progress are denoted by low level, whereas idle resources and tasks in progress are denoted by high level. At time instants 5s and 15s the events u^2 (UGS_3 has low batteries) and u^1 (UGS_1 detects a chemical agent) occur respectively and mission 2 and mission 1 are triggered. Since mission 1 has a lower priority, after the first two tasks are completed ($S4m^1$ and $S5m^1$), the mission is temporarily interrupted because resource R_1 is assigned to mission2 for the execution of task *R1gS3*. At time instant 75 s, R_1, after listening for interrupts (task 6, mission1), is reassigned to mission 2 which can then come to an end with the execution of task $R1dC^2$. Finally, mission 1 can be successfully terminated. From the time traces of figure 10, it is interesting to note that whenever possible, the missions are executed simultaneously, and the shared resources are alternatively assigned to the two missions.

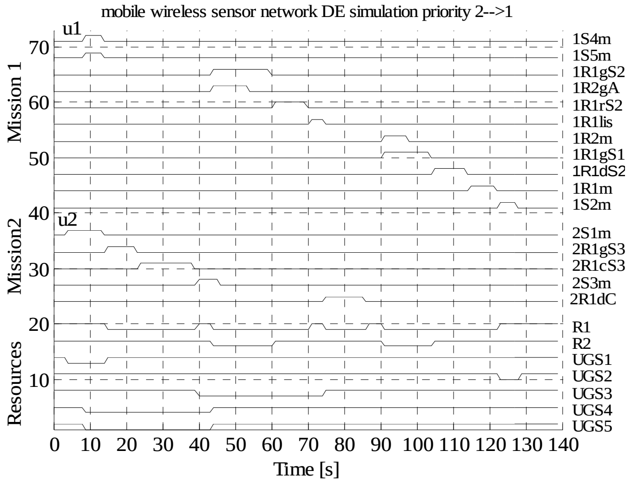


Figure 10. Event time trace simulation results of the WSN

If we change the priority order of the two missions, permuting the block matrices F_{ud}^{1} and F_{ud}^{2}, we get the utilization time trace of figure 11. Mission 2 is executed more fragmentarily, since resource 1 is preferentially assigned to mission1. To further increase the priority of mission1 we could eliminate the listening task *R1lis*, reducing the possibilities of mission 2 to get resource 1 while mission1 is active.

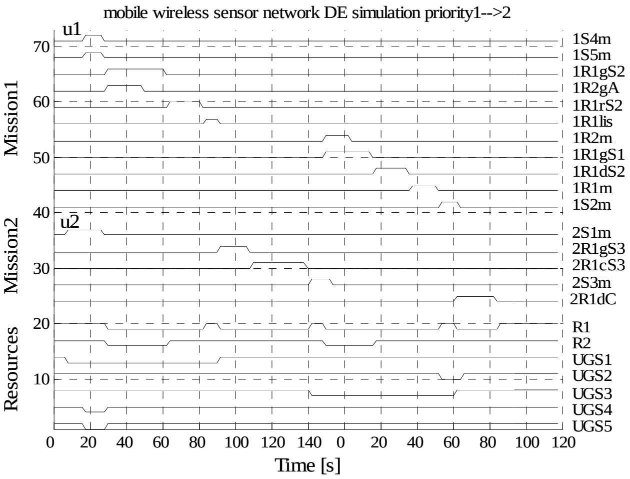


Figure 11. Event time trace simulation results of the WSN after changing the priority order between the two missions

6. Experimental Results

After performing extensive simulations, we can implement the control system directly on the WSN test-bed. Figure 12 shows the actual experimental utilization time trace of the agents, assigning higher priority to mission 2. Notice that the time duration of the real WSN runs in terms of discrete-event intervals, whereas the simulation results shown in figure 10 is in terms of time. It is interesting to note the similarity and fidelity of the dispatching sequences in both the simulation and experimental cases. This is a key result since it shows that the DEC allows one to perform a "simulate and experiment" approach for a WSN, with noticeable benefits in terms of cost, time and performance.

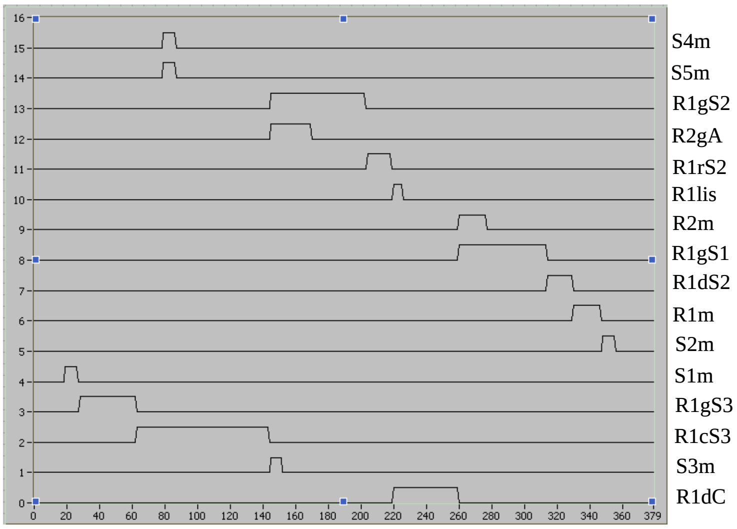


Figure 12. Utilization time trace of the WSN- Experimental results

In figure 13, for the sake of clarity, we have depicted the configuration of the agents in the environment, showing how the topology of the network evolves along time (13a) and finally reaches the configuration shown in figure (13b). In figure 14 and 15, the same results are proposed showing a panoramic view of the mobile WSN.

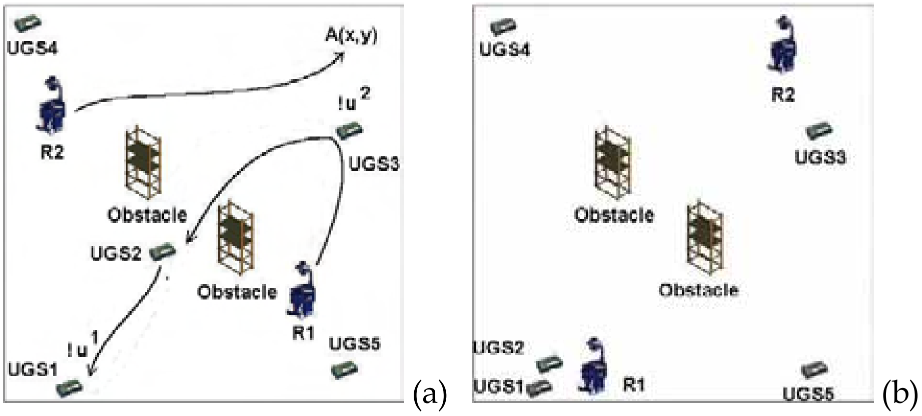


Figure 13. Starting configuration and trajectory followed by the mobile robots (a) - Final configuration after execution of mission 1 and 2 (b)

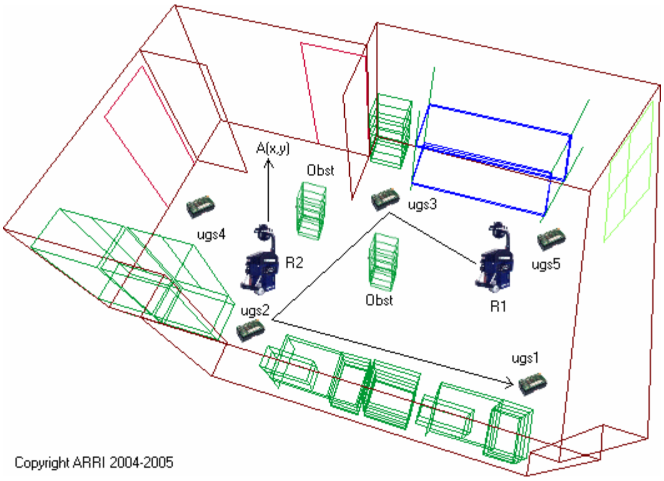


Figure 14. Panoramic view of the configuration of the mobile WSN during real-world experiments

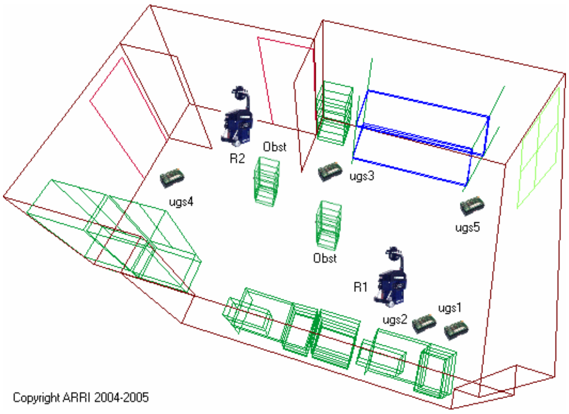


Figure 15. Panoramic view of the final configuration of the mobile WSN

Reassignment of resources

Since the topology of a mobile WSN evolves with time, it is necessary to adapt the mission planning to the changed operating conditions. We want to show now how the matrix formulation of the DEC allows one to easily reconfigure the dispatching rules of the system. Suppose that robot 2 has the same functionalities of robot1 and that mission 2 is triggered after the completion of mission 1 (figure 16a). Since at the end of mission1, robot 2 is actually closer to UGS3, we might consider employing robot2 for mission 2 instead of robot 1 (figure 16b). This change in mission planning would result in simple matrix operations. The '1's relative to the tasks of mission2 are shifted from column 1 to column 2 of matrix F_r^2 and from row 1 to row 2 of matrix S_r^2 (i.e. tasks are shifted from resource1 to resource2). F_{ud} is changed redefining the conflict resolution matrix of each resource for each mission (in this case we have to define $F_{ud}^2(R_2)$ whereas $F_{ud}^2(R_1)$becomes null) and accordingly reassembling the block matrices in F_{ud} (figure 17).

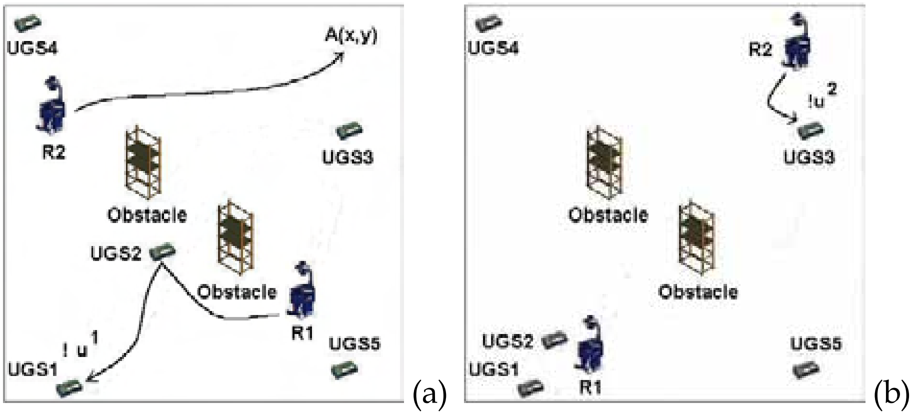


Figure 16. Execution of mission1 (a), change of plans and execution of mission2 b)

$$F_{ud}=\left[\begin{array}{ccc}\multicolumn{2}{c}{\overbrace{\qquad\qquad\qquad\qquad}^{R1}} & \overbrace{\qquad\quad}^{R2}\\ 0 & F_{ud}^{1}(R_1) & F_{ud}^{1}(R_2)\\ F_{ud}^{2}(R_1) & 0 & 0\end{array}\right]\text{a)}$$

$$F_{ud}=\left[\begin{array}{ccc}\overbrace{\qquad\quad}^{R1} & \multicolumn{2}{c}{\overbrace{\qquad\qquad\qquad\qquad}^{R2}}\\ F_{ud}^{1}(R_1) & 0 & F_{ud}^{1}(R_2)\\ 0 & F_{ud}^{2}(R_2) & 0\end{array}\right]\text{b)}$$

Figure 17. Conflict resolution matrix when resource1 (a) or resource 2 (b) are assigned to execute mission2

7. Conclusions

In this chapter we have presented a discrete-event coordination scheme for sensor networks composed of both mobile and stationary nodes. This architecture supports high-level planning for multiple heterogeneous agents with multiple concurrent goals in dynamic environment. The proposed formulation of the DEC represents a complete dynamical description that allows efficient computer simulation of the WSN prior to implementing a given DE coordination scheme on the actual system. The similarity between simulation and experimental results shows the effectiveness of the DEC for simulation analysis. The obtained results also prove the striking potentialities of the matrix formulation of the DEC, namely: straightforward implementation of missions on the ground of intuitive linguistic descriptions; possibility to tackle adaptability and scalability issues at a centralized level using simple matrix operations; guaranteed performances, since the DEC is a mathematical framework which constraints the behaviour of the single agents in a predictable way. Future research will be devoted to the development of high-level decision making algorithms for the dynamic updates of the matrices of the DEC, in order to automatically reformulate the missions on-line according to the current topology of the network and the current perception of the environment.

8. References

Akyldiz I., Su W., Sankarasubramaniam Y, Cayirci E., "A survey on sensor networks", IEEE Communications magazine, August 2002

Balch T., Hybinette M., "Social potentials for scalable multi-robot formations", Proceedings of the International Conference of Robotics and Automation, April 2000

Balch T., Arkin R., "Behavior-based formation control for multirobot teams", IEEE Transactions on Robotics and Automation, vol. 14, no.6, December 1998

Burgard, W.; Moors, M.; Fox, D.; Simmons, R.; Thrun, S.; "Collaborative multi-robot exploration", Proceedings of the IEEE International Conference on Robotics and Automation, 2000, vol. 1 , 24-28 April 2000 Pages:476 - 481

Butler Z., Rus D., "Event-based motion control for mobile-sensor network", IEEE Transactions. on Pervasive Computing, vol.2 issue 4, October-December 2003

Christensen T., Noergaard M., Madsen C., Hoover A., "Sensor networked mobile robotics", Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition, June 2000

Cortes J., Martinez S., Karatas T., Bullo F., "Coverage control for mobile sensing network", IEEE Trans. on Robotics and Automation, vol.20, no.2, April 2004

Gordon-Spears A., Kiriakidis K., "Reconfigurable robot teams: modeling and supervisory control", IEEE Transactions on control system technology, vol. 12, no. 5, September 2004

Harris B., Lewis F., Cook D., "Machine planning for manufacturing: dynamic resource allocation and on-line supervisory control", Journal of Intelligent Manufacturing, pp. 413-430, vol. 9, 1998

Howard, A.; Mataric, M.J.; Sukhatme, G.S.; "An incremental deployment algorithm for mobile robot teams", Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and System, vol.30, Pages:2849 – 2854, October 2002

King J., Pretty R., Gosine R., "Coordinated execution of tasks in multiagent environment", IEEE transactions on Systems, man and cybernetics- Part A: Systems and Humans, vol. 33, no.5, September 2003

Kusiak A. Intelligent scheduling of automated machining systems. In Intelligent design and Manufacturing. A. Kusiak (ed.) Wiley, New York (1992)

Lee J., Hashimoto H, "Controlling mobile robots in distributed intelligent sensor network", IEEE Transactions on Industrial Electronics, vol.50, no.5, October 2003

Lewis F., "Wireless sensor networks", Smart environments: technologies, protocols, and applications, ed. D. J. Cook and S. K. Das, John Wiley, New York, 2004.

McMickell M., Goodwine B., Montestruque L., "MICAbot: a robotic platform for large-scale distributed robotics", Proceedings of the International conference of robotics and automation, September 2003

Mireles J., Lewis F., "Intelligent material handling: development and implementation of a matrix-based discrete event controller IEEE Transactions on Industrial Electronics, , vol. 48 , Issue: 6 , Dec. 2001 Pages:1087 – 1097

Mireles J., Lewis F., "Deadlock analysis and routing on free-choice multipart reentrant flow lines using a matrix-based discrete event controller" Proceedings of the IEEE International conference on Decision and Control, December 2002

Murata, T. "Petri nets: properties, analysis and applications." Proceedings of the IEEE, vol.77, no.4, April 1989, pp.541-80

Petriu E., Whalen T.,Abielmona R., Stewart A., "Robotic sensor agents: a new generation of intelligent agents for complex environment monitoring", IEEE Magazine on Instrumentation and Measurement, vol.7 issue 3, September 2004

Saridis G., "Intelligent robotic control", IEEE Transactions on Robotics & Automation, vol. 28, no.5, May 1983

Sibley G., Rahimi M., Sukhatme G., "Robomote: a tiny mobile platform for large-scale ad-hoc sensor networks", Proceedings of the International conference of robotics and automation, May 2002

Steward D. V., "The design structure system: a method for managing the design of complex systems", IEEE Transactions on Engineering Management, pp. 45-54, Aug. 1981

Tacconi D., Lewis F., "A new matrix model for discrete event systems: application to simulation", IEEE Control System Magazine

Tilak S., Abu-Ghazaleh N., Heinzelman W., "A taxonomy of wireless micro-sensor network models," ACM Mobile Computing and Communications Review, Vol. 6, No. 2,pp. 28-36, 2002

Wu H., Siegel M., Stiefelhagen R., Yang J., "Sensor fusion using Dempster-Shafer theory", Proceedings of IEEE Instrumentation and Measurement Technology Conference,May

Design of a Generic, Vectorised, Machine-Vision library

Bing-Chang Lai & Phillip John McKerrow

1. Introduction

Generic programming is a mechanism by which generic algorithms can be implemented. These generic algorithms are obtained by abstracting concrete algorithms (Musser & Stepanov, 1989). Generic programming implements generic algorithms with comparable runtime performance to hand-coded programs (Geraud *et al.*, 2000). The most widely known and used library based on generic programming is the Standard Template Library (STL), which is now part of the C++ standard. Other libraries based on generic programming are available. For image processing, one currently available generic library is Vision with Generic Algorithms (VIGRA) (Köethe, 2001; Köethe, 1999; Köethe, 2000*c*; Köethe, 1998; Köethe, 2000*a*; Köethe, 2000*b*). VIGRA does not use the vector processor.

A vector processing unit (VPU) applies instructions to vectors. A vector is an array of scalars. Desktop VPUs usually have fixed sized vectors, and all vectors are of the same overall size. Since the number of bits in a vector remains constant, the number of scalars in a vector varies across types. Examples of desktop vector technologies include MMX, 3DNow!, SSE, and AltiVec. VPUs are suitable for applications where the same instructions are applied to large amounts of data – Single Instruction Multiple Data (SIMD) problems. Examples of applications suitable for VPUs include video, image and sound processing.

Despite the VPU being ideally suited for image processing applications, and generic libraries having excellent runtime performance and being flexible, there are currently no generic, vectorised libraries for image processing. Unfortunately, adding VPU support to existing generic, image processing libraries, namely VIGRA, is non-trivial, requiring architectural changes.

This paper discusses the problems that occur when combining vector processing with generic programming, and proposes a solution. The problems associated with vectorisation, and the reasons why existing generic libraries cannot be vectorised directly are covered first. This is followed by detailed descriptions of the more important facets of the proposed solution. The performance of the solution is then compared to hand-coded programs.

In the interest of clarification, this paper will use the following terminology.

VVIS: An abbreviation for Vectorised Vision. VVIS is the generic, vectorised, machine-vision library discussed in this paper.

VVM: An abbreviation for Virtual Vector Machine. VVM is the abstract vector processor

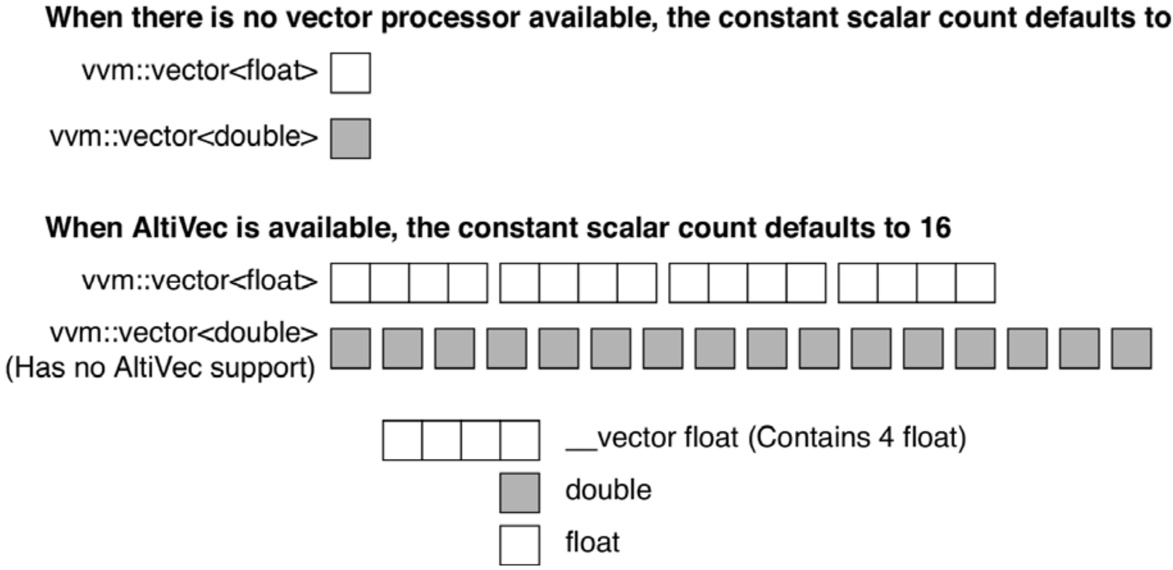

When there is no vector processor available, the constant scalar count defaults to1
vvm::vector<float>
vvm::vector<double>
When AltiVec is available, the constant scalar count defaults to 16
vvm::vector<float>
vvm::vector<double>
(Has no AltiVec support)
__vector float (Contains 4 float)
double
float

Figure 1. Layout of vvm::vector<float> and vvm::vector<double>

(see Section 3), which is used by VVIS.

vp::vector: This refers to a real VPU's native vector type. A vp::vector contains a number of scalars. An example of vp::vectors in AltiVec is __vector unsigned char.

element: In scalar mode, element refers to scalars. When a VPU is available, element refers to vp::vectors.

vvm::vector<T>: This refers to VVM's vector type which contains scalars of type T. A vvm::vector contains a number of elements. These elements are either vp::vectors (which contains a number of scalars) or scalars. See Figure 1 for a graphical overview of the relationship between vvm::vectors, vp::vectors and scalars.

vector: This is used to refer to all vectors, which includes both vp::vectors and vvm::vectors.

2. Problems with Vectorising Existing Generic Libraries

Existing generic libraries, like STL and VIGRA, are difficult to vectorise because the way pixels are arranged in memory is hidden from algorithms by iterators. While such decoupling increases flexibility, hiding how pixels are arranged from algorithms makes it difficult for algorithms to decide whether processing the pixels using the VPU will be beneficial to runtime performance. A vector program that operates on scattered data can be slower than a scalar program, because the scattered data has to be massaged into a form suitable for VPU consumption.

For efficient vectorisation, libraries need to consider:

1. How will algorithms load and write vectors efficiently? How will algorithms handle situations where it is impossible to load and write vectors efficiently?

   Contiguous aligned data are the easiest for the VPU to load efficiently, followed by contiguous unaligned data. Regardless of alignment, contiguous data can be prefetched effectively, which leads to faster loads and stores. When obtaining vectors efficiently is impossible, the easiest and safest solution is to process the data using the scalar processor.

2. How are unaligned data handled? In addition how will two images that have different alignments be handled?

   Since data are unlikely to be aligned in all circumstances, the library needs to consider how unaligned data should be handled. Even when an image is aligned, a region of interest within the image is likely to be unaligned. Apart from unaligned data, an operation that involves more than one input or output needs to consider situations where the alignments of the input and output are different.

3. How are edges handled?

   Edges can be processed using either the scalar processor or the VPU. Processing edges using the VPU is more difficult. When using the VPU, programs can trigger exceptions by accessing memory that does not belong to the current process if the edge is loaded directly. Moreover, using the VPU only produces correct output when the current operation is not affected by the CPU performing more work than necessary. In Section 4.3, we show that quantitative operations cannot have their edges processed using the VPU.

   The scalar processor can always be used to process edges. In addition, because scalar instructions can be executed at the same time as vector instructions, using the scalar processor to handle edges might actually be faster than using the VPU (Apple Computer Inc., 2002).

4. Who handles prefetching?

   Prefetching moves data from memory to the caches before they are used. (Lai *et al.*, 2002) shows that prefetching increased the performance of a vector program significantly on a PowerPC G4. Not all VPUs handle prefetching in the same manner. For example, while the PowerPC G5s also support AltiVec, they require less manual prefetching control because they have automatic prefetching and a larger bus. In fact, some prefetching instructions can be detrimental to speed on the G5, because they cause execution serialisation on the G5 (Apple Computer Inc., 2003). On the G4, such instructions do not cause execution serialisation, and can be used effective within a loop.

3. The Abstract Vector Processor

Generic libraries provide generic functors that can be used with any type that supports the functions used by the functor. For this to be feasible, all types must provide the same functionality through the same interface. Unfortunately, VPU instructions are typically non-uniform across types. An instruction might be available only for some types, and some operations require different instructions for different types. To solve this problem, our generic, vectorised library uses an abstract VPU called Virtual Vector Machine (VVM).

VVM is a templated abstract VPU designed to represent desktop VPUs, such as AltiVec, MMX and 3DNow! . It is a virtual VPU that represents a set of real VPUs with an idealised instruction set and common constraints (Lai *et al.*, 2005). Real VPUs can be abstracted into a virtual VPU when those real VPUs share constraints and have common functionality. For example, AltiVec, MMX and 3DNow! share constraints such as fixed vp::vector sizes and faster access to aligned memory addresses. They all provide operations such as saturated addition and subtraction.

The abstract VPU's idealised instruction set makes the abstract VPU easier to program and portable. It removes the need for programmers to consider instruction availability, and allows a programmer to express his logic using ideal VPU constructs, free from any real VPU's inadequacies. The abstract VPU's common constraints determine which real VPUs it can represent efficiently. Constraints ensure that abstract VPU programs are easier to convert to efficient real VPU programs.

VVM has three constraints common to desktop VPUs: short vvm::vectors, fixed vvm::vector sizes and fast access to aligned memory addresses only. Unlike desktop VPUs, all vvm::vectors have the same scalar count regardless of type. Constant scalar count is important for template programming, which is required for the creation of a generic, vectorised library. Other features that support the creation of a generic, vectorised library include traits, templated vvm::vectors, and consistent functions for both scalar and vvm::vector operations. Due to the high cost of converting types in vector programs, VVM does not provide automatic type conversion; it only supports explicit type conversions.

Constant scalar count: Unlike real VPUs, such as AltiVec, MMX and 3DNow! , which have vp::vectors that are all the same size, and therefore have different scalar counts for different vp::vector types (Motorola Inc., 1999; Mittal *et al.*, 1997; Weiser, 1996), all vvm::vectors consist of the same number of scalars. While the value of this fixed scalar count is not specified by VVM to allow more VPUs to be represented, it is expected to be a small number. A sensible value for this fixed scalar count is the largest number of scalars in a vp::vector. For example, sensible values for scalar counts are 16 for AltiVec, 8 for MMX and 1 for the scalar processor. Constant scalar count is important for template programming, and simplifies type conversions.

Fast access to aligned memory addresses only: VVM can only access aligned memory address quickly. Like SSE and SSE2, VVM also provides slower access to unaligned addresses. In AltiVec, unaligned memory can be accessed by loading aligned memory addresses and performing some transformations (Lai & McKerrow, 2001; Ollmann, 2001; Apple Computer Inc., 2004).

Traits: Traits provide information about types at compile time. Traits are important when automating the generation of more complicated generic algorithms (Köethe, 1999). Trait information can be used in the implementation of VVM itself. They are important for deriving the appropriate boolean type. Unlike scalar programs, vector programs have more than one boolean type. For example, AltiVec has boolean vp::vector types of differing sizes (Motorola Inc., 1999). Promotion traits are important for writing templated code where promotion is necessary, such as in convolutions.

Templated vvm::vectors: Vvm::vectors are templated to support the easy creation of templated vector programs. Templated vector programs are required for a generic, vectorised library implementation in C++.

Consistent functions where applicable: VVM has consistent functions for both scalar and vvm::vector operations where applicable. Consistent functions make VVM easier to use, because programmers can apply their knowledge of scalar operations directly to vvm::vectors. Because VVM has consistent functions, it is often possible to write templated code that performs correctly when instantiated with either a scalar or a vvm::vector. For example, many Standard C++ functors, such as std::plus and std::minus, can be instantiated with either scalars or vvm::vectors.

Comparison operators in VVM however do not return a single boolean, because each vector comparison returns a vector of booleans. In addition, unlike scalar code, true is converted to one's complement of 0 (a value with all bits set to 1, or ~0 in C++) and not 1. VVM maps true to ~0 because in vector programs, the results of comparisons are used as a mask. AltiVec comparison functions also return ~0 for true (Motorola Inc., 1999). Because VVM comparisons return a vvm::vector of booleans and true is converted to ~0, template functions that use comparison operators cannot be instantiated for both scalars and vvm::vectors.

Explicit type conversions only: Type conversions are discouraged in vector programs because type conversions have a pronounced effect on a vector program's performance. Because real vp::vectors typically have different scalar counts, type conversions can change the maximum theoretical speedup. For example, in AltiVec, a __vector unsigned char has 16 scalars, and therefore has a theoretical 16-fold maximum speedup over unsigned char. A __vector unsigned int on the other hand only has 4 scalars and therefore AltiVec has only a 4-fold theoretical maximum speedup. Converting a vvm::vector's type can lead to changes in the theoretical maximum speedup.

The overheads of the VVM implementation used in this paper when compiled with Apple GCC 3.1 20031003 with the -Os switch, were at worst 3.6% slower than hand-coded programs in scalar mode and 0.9% slower in AltiVec mode when operating on char vvm::vectors (Lai *et al.*, 2005; Lai, 2004). For other vvm::vectors in AltiVec mode, the VVM implementation was at worst 23.0% slower than a hand-coded AltiVec program.

4. Categorising Operations Based on their Input-to-Output Correlation

Vector programs are usually difficult to implement efficiently due to memory bottlenecks and non-uniform instructions across types. To minimise this problem, a categorisation scheme based on input-to-output correlation was introduced to reduce the number of algorithms required. Implementing several operations with a common algorithm allows efficiency problems to be solved once.

Operations are categorised by characteristics of the input they require, the output they produce from the input, and how the output is produced from the input. Useful characteristics for categorising image processing operations are the number of input elements required to produce the output, the number of output elements produced from the input, the number of input and output sets, and the types of the input and output elements. The term element is used instead of pixels because separating the type reduces the number of distinct algorithms. For example, since rotation is a one input element (of type coordinate) to one output element (of type coordinate) operation, it can use the same algorithm as threshold, which is a one input element (pixel) to one output element (pixel) operation. The term set is used to refer to a collection of elements.

Number of input and output elements: The number of input and output elements refers to the number of input elements per input set, and number of output elements per output set produced respectively. For example, a threshold operation requires one input element (pixel) per input set (image) to produce one output element (pixel) per output set (image). Arithmetic operations like addition and subtraction also require one input element per input set to produce one output element per output set, but require two input sets.

Number of input and output sets: The number of input and output sets refer to the number of input sets required and the number of output sets produced respectively. In image processing, input sets are typically images, while output sets are images, histograms or statistics. A threshold operation has one input set (an image) and one output set (an image). Binary arithmetic operations, such as addition and subtraction, have two input sets (both images) and one output set (an image). A histogram operation has one input set (an image) and one output set (a histogram).

Input and output element types: Input and output types refer to the type of the input and output elements. All elements have a single type. Most image processing operations have input and output types of pixels. Other possible output types in image processing are histograms, and statistics. In geometric transformations, input and output types are coordinates.

Number of Elements		Number of Sets		Type of Elements		
Input	Output	Input	Output	Input	Output	Examples
1	1	1	1	Pixels	Pixels	Lookup transformations. Colour conversions. Eg. threshold, equalise, reverse and invert.
1	1	2	1	Pixels	Pixels	Arithmetic and logical operations. Eg. addition and subtraction.
Rectangular (eg. 3x3 pixels windows)	1	1	1	Pixels	Pixels	Spatial filters. Eg. convolution filters, edge extraction and edge thickness. Also includes some morphological analysis. Eg. erosion and dilation.
1	0 or more	1	1	Pixels	Number	Quantitative analysis. Eg. perimeter and area.
1	1	1	1	Coordinates	Coordinates	Geometric operations.
M	N	1	1	Coordinates	Coordinates	Scale operations.

Table 1. Some image processing algorithms categorised using input-to-output correlation

Table 1 illustrates how image processing operations can be categorised using the six criteria discussed. M and N refer to the total number of input and output elements respectively. Rectangular refers to a rectangle of input, e.g. a 3x3 pixel window. Spatial filters, like Sobel filters, typically produce a single pixel from a square of pixels centred around a pixel; so they have rectangular input elements. A rectangle was used instead of a square to make the group more general. Since a single pixel is also a rectangle, operations accepting one input element per input set also fall under rectangular.

Whether all input elements are always processed is not part of the criteria, because all input elements are always processed for the operations considered. All the groups shown in Table 1 assume all input elements are processed. An example of an operation that does not always process all input elements is “find first”.

When selecting the correct number of input or output elements per input set, it is useful to visualise how the operation would be implemented using generic programming. For example, threshold operations use one input pixel to decide one output pixel for each iteration. Apart from considering how an operation is implemented using generic programming, it is also helpful to separate the input requirements from the output requirements. For example, histogram operations have one input element per input set, and produce zero or more output elements per output set. They are categorised as one input element per input set because they use each input pixel independently. They produce zero or more elements per output set because the number of elements in the output set is independent of the number of pixels in the image. Histogram operations process each input element one at a time, producing zero output elements until the last pixel is processed, after which many output elements are produced.

4.1 Applying the Categorisation Scheme to Generic Programming

Different criteria are handled by different concepts in a generic library. For example, in the Standard Template Library (STL), the number of input elements, number of output elements, number of input sets and number of output sets are handled by the algorithm. The input element type and the output element type are handled by the iterator. In STL, rotations can be expressed as a std::transform call that takes one input set and one output set, and operates on iterators that return coordinates. In VIGRA, the number of input elements, number of output elements, number of input sets and number of output sets are all handled by the algorithm. However, the input element type and output element type can be handled either by the iterator or the accessor.

Some criteria have more impact on the implementation of the algorithms than others. The most important criteria for categorising operations to reduce the number of algorithms required is the number of input elements because the other characteristics can either be handled by other concepts or have little impact on the algorithm’s implementation. For example, while an algorithm that accepts two input sets instead of one uses more iterators, and has more arguments to the algorithm and to the functors, the structure of the algorithm remains unchanged. The number of output sets, and number of output elements per output set can be handled by either the functor or the accessor. Examples of functors that write output are vigra::FindAverage, vigra::FindBoundingRectangle and vigra::FindMinMax. Input and output element types are handled by iterators and accessors and thus have no impact on the implementation of algorithms.

When applied to generic programming, each defined category will have one main algorithm, with variations for each combination of input and output sets. Once an operation satisfies the criteria for a category, the category’s algorithm can be used, even though it might not be the most efficient. For example, “find first” can be implemented using the same algorithm as histogram’s, since they have the same six criteria. However, “find first” would be faster if it gave up searching after finding the answer. This example shows that more criteria can be added to further narrow the groups. More criteria were not used in this paper, because the objective, a generic, vectorised, machine-vision library, did not require any other criteria.

Under the input-to-output correlation categorisation proposed, an operation can belong to many categories simultaneously. For example, a convolution algorithm can also be used to perform thresholding because one input element is also rectangular input (one pixel is also a 1x1 rectangle). Since all categories are actually subsets of a set whose input element to output element correlation is M to N, only one main algorithm is needed. Such an algorithm would delegate all processing to the functor; it passes all the input to the functor, and waits for all the output. Generally, when an operation cannot be placed into a group smaller than M to N, it is prudent to explicitly implement an algorithm for that operation.

4.2 Inferences

Some of the characteristics discussed have implications for how an operation can be implemented. These characteristics are discussed below.

One input element per input set: This implies that only a 1-D iteration is needed. Hence the algorithm can be written to process images of any shape.

One output element: Having one output element implies that the algorithm can perform more work than necessary, as long as it ensures that the extra results are discarded. Since extra results can be computed, edges can be handled using the VPU (see (Apple Computer Inc., 2002) for more information about edges).

One input element per input set produces one output element per output set: This characteristic suggests that elements can be computed out of turn efficiently. The output order still has to match the input order.

Rectangular input elements per input set: This characteristic indicates that a spatial iterator is required. For vector programs, it is faster to compute the data from left to right, top to bottom, since data loaded from the last iteration can be used in the next.

Zero or more output elements: Zero or more output elements suggest that the algorithm cannot handle the output because the output is unknown. Functors cannot simply return a list of output elements because the answers might be unknown until all elements are processed, and functors generally do not know which element is the last element. While it is possible for the algorithm to inform the functor when the last element is reached, it is easier to let the functor handle the output. In addition, zero or more output elements suggests that performing more work can lead to the wrong output, since the algorithm cannot discard unwanted output. The algorithm cannot discard unwanted output because it does not handle the output.

4.3 Categories for a Generic, Vectorised, Machine-Vision Library

Because the main reason for using this categorisation scheme was to reduce the number of algorithms required while retaining efficiency, the requirements of the algorithms form the basis for defining categories for a generic, vectorised, machine-vision library. From Section 4.1, the most important criteria for categorising operations to reduce the number of algorithms required is the number of input elements. Using only this criterion results in two categories, indicating that only two main algorithms are required to handle the image processing operations considered in Table 1. However, because output characteristics were not used, the functor would be always responsible for the output. Because functors

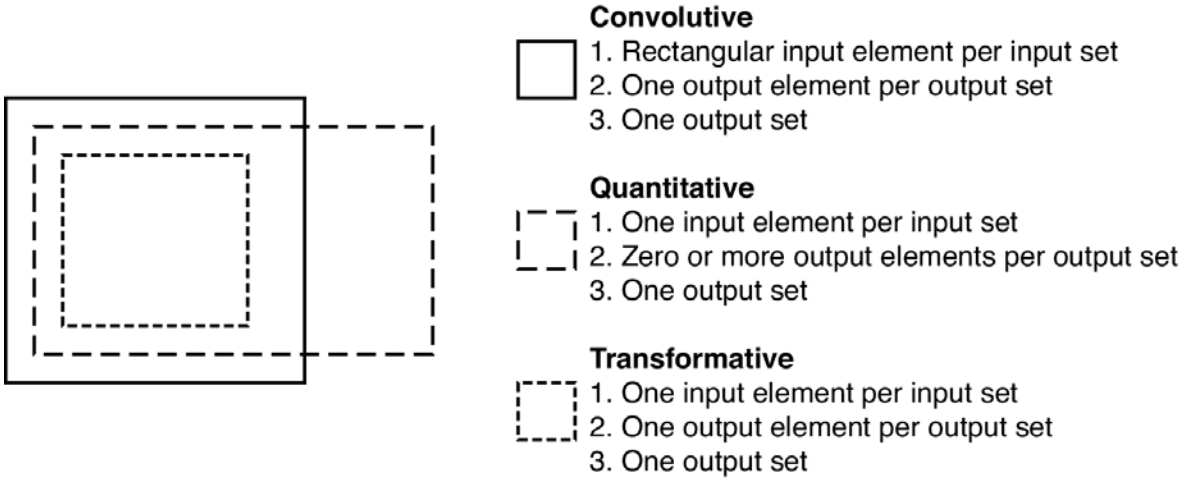

Convolutive
1. Rectangular input element per input set
2. One output element per output set
3. One output set
Quantitative
1. One input element per input set
2. Zero or more output elements per output set
3. One output set
Transformative
1. One input element per input set
2. One output element per output set
3. One output set

Figure 2. Categorisation for a generic, vectorised, machine-vision library

that handle output are more difficult to implement, are not consistent with existing generic libraries, and most image processing operations produce a single image as their output, consequently, the number of output sets was also used to derive the categories for the generic, vectorised, machine-vision library. Since functors are the most common concept created by users, making functors more difficult to implement would make the library more difficult to use.

Number of input elements per input set, number of output elements per output set and number of output sets were used to divide the image processing operations considered in Table 1 into three categories: quantitative, transformative and convolutive. Figure 2 shows how the three categories are related to each other.

Quantitative operations: Quantitative operations have one input element per input set, zero or more output elements per output set and a single output set. Because they have zero or more output elements, the output is handled by the functor. An example of a quantitative operation is the histogram.

Transformative operations: Transformative operations accept one input element per input set. In image processing, transformative algorithms require one or two input sets, and produce one output image set. Since transformative operations involve either one or two input sets, two algorithms are needed. Since they have one output element per output set, edges can be processed using the VPU. In addition, because they have one input element per input set, they are easy to parallelise and can be computed out of order without problems. Examples of transformative operations are thresholds, additions and subtractions.

Convolutive operations: Convolutive operations accept a rectangle of elements per input set, from which they produce one output element for one output set. Convolutive algorithms are named after the operations that mostly fall under it – convolutions. Because they have one output element per input set, edges can be processed using the VPU. Because they have rectangular input, convolutive algorithms require spatial iterators. While convolutions can be computed out of order, there is little reason for vector programs to do this, because most of the input already loaded for the current output is required for computing the next output. Examples of convolutive operations are linear and non-linear filters like Sobel filters, and Gaussian filters.

5. Storages

As mentioned previously, one of the main reasons why existing generic libraries cannot be vectorised easily is because iterators provide no information on the arrangement of pixels in memory to algorithms. To address this problem, the storage concept is introduced.

Storages specify constraints on how the data contained within them are arranged. Instead of passing iterators to algorithms, storages are passed. Storages allow algorithms to make informed decisions regarding the use of the VPU. Since some algorithms operate on specific geometric shapes, it is important to associate a shape with a storage. Because iterators can move and thereby break the constraints guaranteed by a storage, passing iterators to the algorithm makes it difficult to decide whether or not to use the VPU at compile time.

Three storage types are specified: contiguous, unknown and illife. Contiguous and unknown storages are one-dimensional storages, while illife storages are n-dimensional storages. Contiguous storages are processed using the VPU, while unknown storages are processed using the scalar processor. Contiguous storages are actually a subset of unknown storages, and as a result, contiguous storages support the unknown storage's interface. This is important when processing multiple storages together. When a single storage is unknown, all the storages are processed as unknown storages.

5.1 Contiguous Storages

Contiguous storages are one-dimensional storages where components of pixels are adjacent to each other in memory. Originally, components were required to be adjacent from the beginning of the storage to the end. However, this restriction was later eased to allow for chunky vector storages, which are storages where components are interleaved one vector at a time. Contiguous storages require components of pixels to be adjacent for a single vvm::vector. Table 2 shows the interface that all contiguous storages must provide.

Two kinds of contiguous storages are specified: contiguous aligned storages and contiguous unaligned storages. Contiguous unaligned storages must be convertible to contiguous aligned storages. The corresponding contiguous aligned storage type is specified by the contiguous_aligned_storage template metafunction, The inverse conversion is not necessary, because a contiguous aligned storage fulfills the criteria for a contiguous unaligned storage already.

The iterator returned by begin(), points to the first element, and must be aligned for contiguous aligned storages, but may be unaligned for contiguous unaligned storages. end() returns an iterator that points to the first past-the-end scalar, and can be unaligned for both types of contiguous storages. While end() can return an unaligned iterator, contiguous storages must ensure that performing a vvm::vector load from the last aligned position is valid. Contiguous storages are expected to pad the number of bytes allocated, so that the allocation ends on an aligned position. Figure 3, which assumes that there are four scalars in a vvm::vector, illustrates this requirement. Three extra scalars were allocated at the end to allow the last aligned position to be loaded as a vvm::vector.

5.1.1 Iterator Requirements

An iterator for a contiguous storage must be able to traverse forwards and backwards by taking scalar, vvm::vector, or pixel steps. Advancing an iterator by scalar or vvm::vector steps changes the position represented by the iterator by a scalar or vvm::vector respectively. Scalar steps are only valid when the step does not leave the current vvm::vector. Pixel steps are the same as scalar steps, except that they can cross vvm::vector boundaries. Pixel steps are used by convolutive algorithms to adjust the end iterator. Figure 4 illustrates the differences between scalar, vvm::vector and pixel steps when iterating through a chunky vector storage.

Both scalar and pixel steps are required despite both advancing by scalars because pixel steps may require more processing, and crossing vvm::vector boundaries is only required occasionally. For planar storages, scalar and pixel steps have the same implementations. However, for chunky vector storages, a scalar step is easier to implement than a pixel step.

The iterator is expected to treat scalar and vvm::vector components orthogonally. This allows the ! = operator to be applied to each component separately, and removes the need to calculate the location of the right edge. There is no need to calculate the position of the right edge because once the vvm::vector steps are completed, only the right edge remains. While keeping two orthogonal components can increase the cost of creating iterators, since both scalar and vvm::vector components have to be calculated, it simplifies algorithm implementation and reduces processing requirements during execution.

The required interface for an iterator for a contiguous storage is shown in Table 3. Note that there is no operation to read data from or write data to the current location. The required interface for reading and writing data is determined by what accessors the iterator has to be compatible with. Accessors are discussed later in Section 7.

Expression	Return Type	Notes
X::component_tl	ct::typelist of T1, T2, ...	
X::iterator	iterator type pointing to T	Convertible to X::const_iterator
X::const_iterator	iterator type pointing to const T	
a.begin()	iterator; const_iterator for constant a	
a.end()	iterator; const_iterator for constant a	
contiguous_aligned_storage <X>::type	Contiguous aligned storage	
contiguous_aligned_storage <X>::const_type	const contiguous aligned storage	

Table 2. Contiguous storage's required interface

When designing the iterator for a contiguous storage, the following ideas were considered: for iterators to keep a single location, to advance the iterator by adding constants, and to have different iterator types for each step type. These ideas were discarded for reasons discussed below.

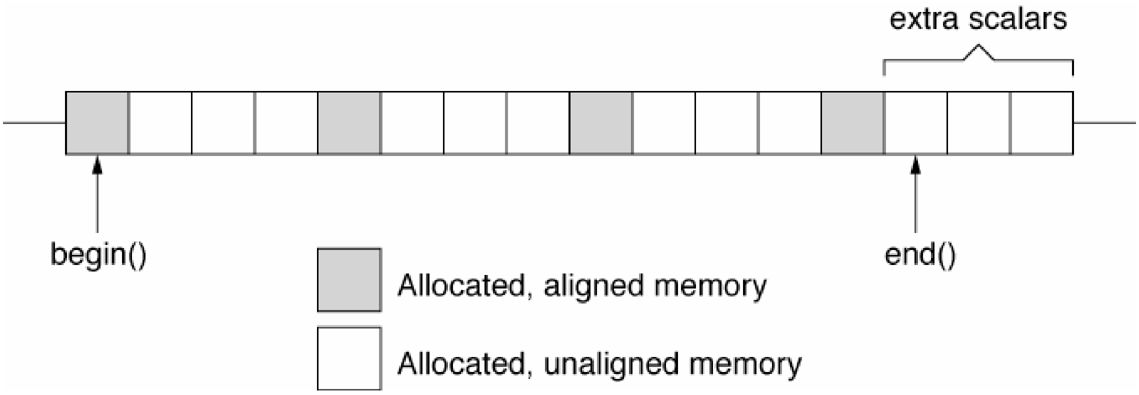


Figure 3. Expected allocation requirements of a contiguous storage

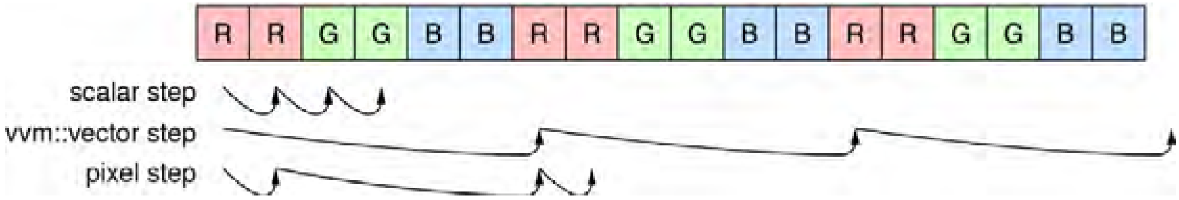


Figure 4. Differences between scalar, vvm::vector and pixel steps when iterating through a chunky vector storage

Keeping a single location in the iterator: The iterator originally kept a single location, and provided methods to move the position by different step types. Keeping a single location forces the algorithm to calculate the position of the right edge, and might increase the execution cost of the != operator, if the operator needs to calculate the real position first.

Advancing the iterator by adding constants: Advancing the iterator by scalar and vvm::vector steps by adding constants was also considered. For example, assuming i is an iterator, i += scalar_step would be a scalar step, and i += vector_step would be a vvm::vector step. For planar storages, scalar_step and vector_step would be equal to one and the VVM scalar count respectively. This method would have allowed iterators to be implemented as pointers, thereby producing code that was probably easier for compilers to optimise.

While this method can iterate through planar storages easily, it cannot iterate through some multi-channel chunky storages with differing channel types. In such cases, vector_step cannot be specified in terms of number of scalars. For example, a multi-channel chunky scalar storage, which is a chunky storage where components are interleaved one scalar at a time, with int RGB channels and a char alpha channel needs to have a scalar_step of 1 int and a vector_step of 3.5 ints. If we use different object types to represent scalar, vvm::vector, and pixel steps, we can overcome this problem. (Weihe 2002) provides some ideas on how such syntax can be implemented correctly. This idea was not investigated further because it seemed likely to introduce performance overheads.

Operation	Result	Notes
difference_type	Type	Returns type of i - j
i = j	iterator&	After operation, i != j is false
i != j	bool	After operation, i == j is false
i.vector != j.vector	bool	
i.scalar != j.scalar	bool	
i - j	difference_type	Returns number of pixels between i and j
++i.vector; i.vector++	iterator&	Move forward by 1 vector
i.vector += j	iterator&	Move forward by j vectors
++i.scalar; i.scalar++	iterator&	Move forward by 1 scalar
i.scalar += j	iterator&	Move forward by j scalars
++i; i++	iterator&	Move forward by 1 pixel
i += j	iterator	Returns an iterator that has moved forward by j pixels
--i.vector; i.vector--	iterator&	Move backward by 1 vvm::vector
i.vector -= j	iterator&	Move backward by j vvm::vectors
--i.scalar; i.scalar--	iterator&	Move backward by 1 scalar
i.scalar -= j	iterator&	Move backward by j scalars
--i; i--	iterator&	Move backward by 1 pixel
i -= j	iterator&	Returns an iterator that has moved backward by j pixels

Table 3. Required interface for an iterator for a contiguous storage

Operation	Result	Semantics
difference_type	Type	Returns type of i - j
i = j	iterator&	After operation, i != j is false
i ! = j	bool	Checks if i and j are pointing to the same location
i - j	difference_type	Returns number of pixels between i and j
++i; i++	iterator&	Move forward by 1 pixel
i += j	iterator&	Move forward by j pixels
--i; i--	iterator&	Move backward by 1 pixel
i -= j	iterator&	Move backward by j pixels

Table 5. Unknown storage's required iterator interface

Using different iterator types for different step types: Using different iterator types for each step type was also considered, because it was thought that this method would allow pointers to be used as iterators with both planar and chunky storages. Unfortunately, this is not possible, because there would have been only two pointer types (scalar and vvm::vector) and three different step types. In addition, the different iterators need to be convertible between one another, because we want to be able to advance by vvm::vector steps when we were using the VPU and switch to scalar steps when we came to the edges. The only way to convert pointers is to use reinterpret_cast; it is not possible to perform automatic conversions between pointer types. reinterpret_cast is not suitable when the iterators are not pointers.

5.2 Unknown Storages

Unknown storages are one-dimensional storages where there are no restrictions on how components of a pixel are arranged. Since there are no restrictions, all one-dimensional storages are unknown storages. The interface of the unknown storages is a subset of the interface of contiguous storages. begin() and end() can both return unaligned iterators. begin() returns an iterator that points to the first element. end() returns an iterator that points to the past-the-end element. Table 4 summarises the interface that unknown storages must implement.

5.2.1 Iterator Requirements

An iterator for an unknown storage is expected to be able to traverse forwards and backwards by taking pixel steps. Scalar and vvm::vector steps are not required, because unknown storages are expected to be processed by the scalar processor. The iterator's interface is shown in Table 4. The interface is the same as the interface of a contiguous storage's iterator, exception there are no .scalar and .vector operations.

Expression	Return Type	Notes
X::component_tl	ct::typelist of T1, T2, ...	
X::iterator	iterator type pointing to T	Convertible to X::const_iterator
X::const_iterator	iterator type pointing to const T	
a.begin()	iterator; const_iterator for constant a	
a.end()	iterator; const_iterator for constant a	

Table 4. Unknown storage's required interface

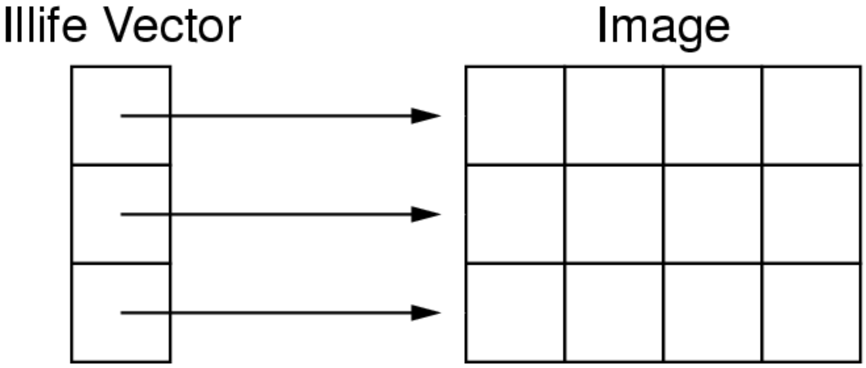


Figure 5. An illife vector

5.3 Illife Storages

Illife storages are storages that contain other storages. Illife storages are named after illife vectors, which are vectors that store row pointers in an image. Figure 5 illustrates an illife vector. Illife storages are expected to be row-major. Pixels adjacent horizontally are adjacent in memory, while pixels adjacent vertically are not. Row-major was chosen because this is the usual major used in the implementation language C++.

Unlike contiguous or unknown storages, the illife storage represents an n-dimensional storage. It is n-dimensional because the storage type of the rows can also be an illife storage. However, some algorithms, like convolutions, require the illife storages to contain only unknown or contiguous storages, that is to be two-dimensional.

The interface of illife storages is shown in Table 6. Since the illife storage's interface is quite similar to STL containers, it should be straightforward to use STL containers to house each row's storage. begin() returns an iterator referring to the first row. end() returns an iterator that refers to the first past-the-end row.

Expression	Return Type	Notes
X::value_type	T	T is assignable
X::iterator	iterator type pointing to T	Convertible to X::const_iterator
X::const_iterator	iterator type pointing to const T	
a.begin()	iterator; const_iterator for constant a	
a.end()	iterator; const_iterator for constant a	

Table 6. Illife storage's required interface

Expression	Return Type	Notes
i = j	iterator&	After operation, i != j is false
i ! = j	bool	
++i; i++	iterator&	Moves down by 1 row
i += j	iterator&	Moves down by j rows
i + j	iterator	Returns an iterator j rows down
--i; i--	iterator&	Moves up by 1 row
i -= j	iterator&	Moves up by j rows
i - j	iterator	Returns an iterator j rows up
*i	value_type	Returns the current row
i[n]	value_type	Returns the row n rows down

Table 7. Illife storages' required iterator interface

The illife storage's interface has functions with the same names as the contiguous storage's interface, which perform different operations. For example, in contiguous and unknown storages, begin() returns an iterator that points to the beginning of image data while in illife, begin() returns an iterator pointing to the first row's storage. Because of this, a storage that supports the illife storage interface cannot support the contiguous or unknown storage interfaces.

In initial designs, the illife storage had an interface that was compatible with unknown and contiguous storages. This allowed the same storage to implement both the interfaces of both the illife storage and a one-dimensional storage. begin and end functions accepted an int, which specifies a row, and returned iterators to image data. However, such an interface makes it more difficult to call the algorithms for one-dimensional storages directly for a row, because algorithms accept storages and not iterators. Since the final design returns iterators to storages instead of to image data, storages can be easily extracted from the iterators.

5.3.1 Iterator Requirements

The iterator's requirements are shown in Table 7. The iterator supports the +, - and [] operators. These operators allow algorithms, such as the convolutive algorithm, to access rows offset from the current row easily.

Expression	Return Type	Notes
A::prefetch_channel_count	int	Returns number of prefetch channels required
A::scalar_type	Type	Scalar Type
A::vector_type	Type	Vvm::vector Type
a.prefetch_read<ch>(i)	void	Prefetch from iterator i for read using channel ch
a.get_scalar(i)	scalar	Returns the scalar at iterator i
a.get_scalar(i, o)	scalar	Returns the scalar at iterator i+o scalar steps
a.get_vector(i)	vector	Returns the vector at iterator i
a.get_vector(i, o)	vector	Returns the vector at iterator i+o vector steps
a.get_vector(a, b, o)	vector	Returns a vector from a and b. The first scalar of the vector is a[o].

For unknown storages, A::prefetch_channel_count, A::vector_type, a.prefetch_read<ch>(i), a.get_vector(i), and a.get_vector(i, o) do not need to be implemented.

Table 8. Read accessor's required interface

6. Regions

Regions are storages that represent regions of interest in other storages. Since regions are storages, regions can be passed to algorithms. In existing generic libraries, iterators mark the region of interest to algorithms. Regions allow for the same region of interest marking in a generic, vectorised library.

Regions are expected to have an admit/release phase like VSIPL (Georgia Tech Research Corporation, 2001; *VSIPL website*, 2001). Like VSIPL, access to a portion of the storage that has been admitted to the region, must be made through that region. A storage can have more than one portion admitted to different regions at one time, as long as the portions do not overlap. This admit/release phase is important in allowing regions to perform pre- and post- processing that will be helpful to runtime performance.

7. Accessors

In STL, data elements are accessed via references to the original data returned by operator* and operator[]. A problem with this approach is that, for multi-channel planar images, there is no efficient way of returning a reference to a pixel. To access a pixel that contains all the channels of a multi-channel planar image, a proxy object is required. This proxy object reduces the efficiency of accessing the data elements (Kühl & Weihe, 1997). Accessors were introduced by (Kühl & Weihe, 1997) to solve this problem by providing an extra level of indirection.

Unlike accessors in VIGRA, or those described in (Kühl & Weihe, 1997), that provide access to a single type, accessors in VVIS provide access to two types – scalars and vvm::vectors. In addition, apart from being responsible for retrieving and writing data, and performing any necessary type conversions, accessors in VVIS are also responsible for prefetching. Read and write accessor requirements for contiguous storage are shown in Tables 8 and 9 respectively.

Expression	Return Type	Notes
A::scalar_type	Type	Scalar Type
A::vector_type	Type	Vvm::vector Type
A::prefetch_channel_count	int	Returns number of prefetch channels required
a.prefetch_write<ch>(i)	void	Prefetch from iterator i for writing using channel ch
a.set_scalar(s, i)	void	Writes scalar s to iterator i
a.set_vector(v, i)	void	Writes vector v to iterator i

For unknown storages, A::prefetch_channel_count, A::vector_type, a.prefetch_write<ch>(i), and a.set_vector(v, i) do not need to be implemented.

Table 9. Write accessor's required interface

Prefetching is handled by accessors because only accessors know exactly what data is being loaded and stored. For example, an accessor that provides access to a single channel only needs to prefetch that single channel instead of all the channels. Prefetching functions are only used for contiguous storages.

Accessors provide functions to get and set both scalars and vvm::vectors at the current location, or at an offset from the current location. There is an extra function for loading a vvm::vector from two vvm::vectors, that can be used to perform unaligned loading.

scalar_type and vector_type refer to the scalar and vvm::vector type used by the accessor. This is useful when the algorithm wishes to keep a copy of the values returned by the accessor. prefetch_channel_count is an integer containing the number of prefetch channels required when prefetching. To prefetch from multiple storages, prefetch_channel_count of the previously prefetching storage should be passed to the next set of prefetch functions, to avoid using the same prefetch channels. The following example illustrates how to prefetch multiple storages.

```
// Assume there are three accessor types, A1, A2, A3
// Assume a1, a2, a3 are of types A1, A2, A3
// Assume there are three iterators i1, i2, i3
// Assume i1 and i2 are for reading, i3 is for writing
```

```
    a1.prefetch_read<0>(i1);
    a2.prefetch_read<A1::prefetch_channel_count>(i2);
    a3.prefetch_write<A1::prefetch_channel_count +
            A2::prefetch_channel_count>(i3);
```

8. Algorithms

Algorithms are responsible for coordinating how the other concepts are used to solve a problem. In Section 4.3, three categories were identified for use with VVIS: quantitative, transformative and convolutive. for_each, transform and convolute are the algorithms for quantitative, transformative and convolutive categories respectively.

Table 10 shows the algorithms that VVIS supports. The transform algorithm has two versions: one for one input set, and one for two input sets. Like VIGRA, algorithms accept an accessor for each input and output set. Without an accessor for each input and output set, it is more difficult to perform operations on images that involve different channels.

Algorithms are expected to process data differently depending on the storage type. Algorithms are expected to use the scalar processor and the VPU to process unknown and contiguous storages respectively. An algorithm in VVIS does not have to provide an implementation for all three storage types. For transformative and quantitative operations, algorithms must provide an implementation for unknown and illife storages. For convolutive operations, which require spatial iterators and thus two-dimensional storages, algorithms must provide only an implementation that processes illife storages which contain unknown row storages. Implementations for contiguous storages are not required because contiguous storages implement the unknown storage interface.

Expression	Return Type	Notes
vvis::convolute(in, ia, out, oa, f)	void	Convolutive algorithm
vvis::for_each(in, ia, f)	void	Quantitative algorithm
vvis::transform(in, ia, out, a, f)	void	Transformative algorithm
vvis::transform(in1, ia1, in2, ia2, out, oa, f)	void	Transformative algorithm

Table 10. VVIS algorithms

9. Functors

All VVIS functors must provide a scalar and a vvm::vector implementation, because data is processed using both the scalar processor and the VPU. Since VVM uses consistent functions for scalar and vvm::vector operations where applicable, it is possible to implement a single templated functor for both versions in many cases. Having two implementations for each functor prevents STL binders from working with VVIS functors. Thus VVIS also provides its own binders.

The three different categories have different functor requirements. Quantitative and transformative functors both accept input via operator(). Transformative functors are expected to return their answers, while quantitative functors keep their answer. Convolutive functors have a more complex interface because they accumulate input one pixel at a time, and then return the answer for that set of input. Tables 11, 12, and 13 summarises the functor requirements for the quantitative, transformative, and convolutive operations respectively.

Expression	Return Type	Notes
f(s)	Void	s is a scalar
f(v)	Void	v is a vector

Table 11. Quantitative functors' required functor interface

Expression	Return Type	Notes
f(s)	Output	s is a scalar
f(v)	Output	v is a vector

Table 12. Transformative functors' required functor interface

Expression	Return Type	Notes
A::kernel_type	Type	Type of a.kernel()
a.reset()	void	Signals start of new rectangle input
a.accumulate(s)	void	Accumulate a scalar
a.accumulate(v)	void	Accumulate a vector
a.scalar_result()	scalar	Returns the scalar answer
a.vector_result()	vector	Returns the vector answer
a.kernel_width()	const int	Returns the width of the kernel
a.kernel_height()	const int	Returns the height of the kernel

Table 13. Convolutive functors' required functor interface

10. Performance

To give readers some idea on the performance of the generic, vectorised, machine-vision library, VVIS, runtime performance results are presented in this section.

Figure 6 compares the runtime performance of VVIS's transformative algorithm to hand-coded programs, in scalar and in AltiVec mode. All results were collected on an Apple

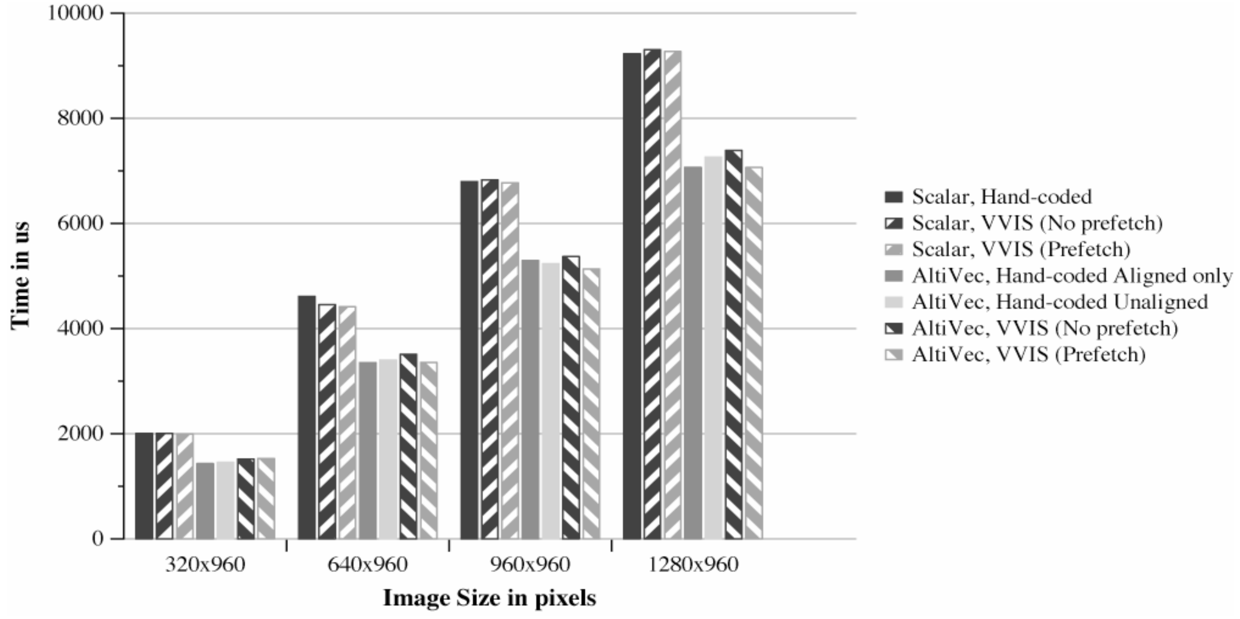


Figure 6. Performance of different vvis::transform implementations when the source and destination single-channel images are different, and the functor adds input values to themselves

PowerBook G4 with one PowerPC 7447A 1.3GHz with 32K L1 instruction cache, 32K L1 data cache, and 512K L2 cache, from programs that were compiled with the -Os (optimise for size) switch. Each result shown is the lowest time obtained after 20 runs. In each run, the source and destination images were allocated, and the source image was filled with data, after which vvis::transform was timed. The functor used added each input value to itself. All source and destination images used were aligned. The hand-coded scalar and AltiVec programs are based on the programs presented in (Lai *et al.*, 2002).

Figure 6 shows that for single-channel, unsigned char images, VVIS is comparable with both scalar and AltiVec hand-coded programs. For unsigned char operations, the VVM implementation used had no significant overheads.

11. Conclusion

Existing generic libraries, such as STL and VIGRA, are difficult to vectorise because iterators do not provide algorithms with information on how data are arranged in memory. Without this information, the algorithm cannot decide whether to use the scalar processor or the VPU to process the data. A generic, vectorised library needs to consider how functors invoke VPU instructions, how algorithms access vectors efficiently, and how edges, unaligned data, and prefetching are handled. The generic, vectorised, machine-vision library design presented in this paper addresses these issues.

The functors access the VPU through an abstract VPU. An abstract VPU is a virtual VPU that represents a set of real VPUs through an idealised instruction set and common constraints. The implementation used has no significant overheads in scalar mode, and for char types in AltiVec mode. Functors must also provide two implementations, one for the scalar processor and one for the VPU. This is necessary because the solution proposed uses both the scalar processor and the VPU to process data.

Since VPU programs are difficult to implement efficiently, a categorisation scheme based on input-to-output correlation was used to reduce the number of algorithms required. Three categories were specified for VVIS: quantitative, transformative and convolutive. Quantitative operations require one input element per input set to produce zero or more output elements per output set. Transformative operations are a subset of quantitative and convolutive operations, requiring one input element per input set to produce one output element per output set. Convolutive operations accept a rectangle of input elements per input set to produce one output element per output set.

Storages provide information on how data are arranged in memory to the algorithm, allowing the algorithm to automatically select appropriate implementations. Three main storage types were specified: contiguous, unknown or illife. Contiguous and unknown storages are one-dimensional while illife storages are n-dimensional storages. Only contiguous storages are expected to be processed using the VPU. Two types of contiguous storages were also specified: contiguous aligned storages, and contiguous unaligned storages. The iterator returned by begin() is always aligned for contiguous aligned storages, but may be unaligned for contiguous unaligned storages. Different algorithm implementations are required for different storage types. To support processing of different storage types simultaneously, storage types are designed to be subsets of one another. This allows an algorithm to gracefully degrade VPU usage and to provide efficient performance in the absence of VPUs.

Edges are handled by an iterator with orthogonal scalar and vvm::vector components. The contiguous storage's iterator allows traversal using scalar, vvm::vector or pixel steps. Scalar steps change only the scalar component and are only valid if the iterator does not cross a vvm::vector boundary. Vvm::vector steps change only the vvm::vector component. Pixel steps can change both the scalar and vvm::vector components. The orthogonal components make handling edges easy: once the vvm::vector component is exhausted, only the edge remains.

Contiguous unaligned storages allow algorithms to handle unaligned data directly. To facilitate such implementations, all contiguous storages ensure that the last edge can be loaded using the VPU without triggering memory exceptions. In cases where an algorithm that processes unaligned data is too difficult to implement, contiguous unaligned storages can be converted to contiguous aligned storages.

Prefetching is delegated to accessors. Accessors handle prefetching because only accessors know which data are actually going to be used. For example, an accessor might provide access to only the red channel of a RGB image, and thus it should only prefetch the red channel.

Runtime performance results from a transformative algorithm are presented. These results show that with an abstract VPU with no significant overheads, the solution is comparable in performance to hand-coded programs in both scalar and AltiVec mode.

12. References

Apple Computer Inc. (2002), *Code Optimization*. http://developer.apple.com/hardware/ve/code_optimization.html

Apple Computer Inc. (2003), *PowerPC G5 Performance Primer*. http://developer.apple.com/technotes/tn/tn2087.html

Apple Computer Inc. (2004), *Performance Issues: Memory Usage*. http://developer.apple.com/hardware/ve/performance_memory.html

Georgia Tech Research Corporation (2001), *VSIPL 1.01 API*. http://www.vsipl.org/PubInfo/vsiplv1p01_final1.pdf

Geraud, T., Fabre, Y., Duret-Lutz, A., Papadopoulos-Orfanos, D. & Mangin, J.-F. (2000), Obtaining genericity for image processing and pattern recognition algorithms, *in* 'Pattern Recognition, 2000. Proceedings. 15th International Conference on'.

Köethe, U. (1998), 'On data access via iterators (working draft)'. http://kogs-www.informatik.uni-hamburg.de/~koethe/vigra/doc/documents/DataAccessors.ps

Köethe, U. (1999), 'Reusable software in computer vision', *B. Jähne, H. Haussecker, P. Geissler: "Handbook on Computer Vision and Applications"* **3**. http://kogs-www.informatik.uni-hamburg.de/~koethe/papers/handbook.ps.gz

Köethe, U. (2000*a*), 'Expression templates for automated functor creation'. http://kogs-www.informatik.uni-hamburg.de/~koethe/vigra/doc/documents/FunctorFactory.ps

Köethe, U. (2000*b*), Generische Programmierung für die Bildverarbeitung, PhD thesis, Universität Hamburg. http://kogs-www.informatik.uni-hamburg.de/~koethe/papers/DissPrint.ps.gz

Köethe, U. (2000*c*), 'STL-style generic programming with images', *C++ Report Magazine* pp. 24–30.http://kogs-www.informatik.uni-hamburg.de/~koethe/papers/GenericProg2DC++Report.ps.gz

Köethe, U. (2001), *VIGRA Reference Manual for 1.1.2*. http://kogs-www.informatik.uni-hamburg.de/~koethe/vigra/doc/index.html

Kühl, D. & Weihe, K. (1997), 'Data access templates', *C++ Report* 9/7, 15, 18–21.

Lai, B.-C. (2004), Vector processor software for machine vision, PhD thesis, University of Wollongong. Under Examination.

Lai, B.-C. & McKerrow, P. J. (2001), Programming the velocity engine, *in* N. Smythe, ed., 'e-Xplore 2001: a face to face odyssey', Apple University Consortium.

Lai, B.-C., McKerrow, P. J. & Abrantes, J. (2002), Vectorized machine vision algorithms using AltiVec, Australasian Conference on Robotics & Automation.

Lai, B.-C., McKerrow, P. J. & Abrantes, J. (2005), 'The abstract vector processor', *Microprocessors and Microsystems* . Under Review.

Mittal, M., Peleg, A. & Weiser, U. (1997), 'MMX technology architecture overview', *Intel Technology Journal '97* .

Motorola Inc. (1999), *AltiVec Technology Programming Interface Manual*. http://e-www.motorola.com/brdata/pdfdb/microprocessors/32_bit/powerpc/altivec/altivecpim.pdf

Musser & Stepanov (1989), Generic programming, *in* 'ISSAC: Proceedings of the ACM SIGSAM International Symposium on Symbolic and Algebraic Computation. http://citeseer.nj.nec.com/musser88generic.html

Ollmann, I. (2001), Altivec.http://www.alienorb/AltiVec/Altivec.pdf

VSIPL website (2001). Maintained by Dr. Mark Richards of Georgia Tech Research Institue for DARPA and U.S. Navy.

Weihe, K. (2002), 'Towards improved static safety: Expressing meaning by type', *C/C++ Users Journal* pp. 32,34–36,38–40.

Weiser, U. (1996), 'MMX technology extension to the Intel architecture', *IEEE Micro* pp. 42–50.

An Active Stereo Vision-Based Learning Approach for Robotic Tracking, Fixating and Grasping Control

Nan-Feng Xiao & Saeid Nahavandi

1. Introduction

Vision-based robotic tracking, fixating and grasping control depends on many environmental factors in an unknown environment. The robot control systems lack robustness, and the calibration of the CCD cameras is very slow and tedious in the existing methods.

Although the binocular cameras can solve some of these problems, it is necessary to rely on the time consuming and complicated 3-D reconstruction algorithms (8)-(9). Therefore, it is necessary to develop a more effective vision-based robotic tracking, fixating and grasping method, and use the robotic learning ability to improve the tracking, fixating and grasping in the unknown environment.

This chapter presents an active stereo vision-based learning approach for robotic tracking, fixating and grasping. First, the many-to-one functional mapping relationships are derived to describe the spatial representations of the object in the workspace frame. Then, ART_NN and FF_NN are used to learn the mapping relationships, so that the active stereo vision system guides the end-effecter to track, fixate and grasp the object without the complicated coordinate transformation and calibration. Finally, the present approach is verified by simulation.

2. Visual Tracking, Fixating and Grasping

Active vision can easily realize selective attention and prevent an object to go out of the view fields of the cameras, therefore the active stereo vision-based robotic tracking, fixating and grasping can achieve greater flexibility in an unknown environment.

Figure 1 shows an active stereo vision-based robotic system for the tracking, fixating and grasping. The CCD cameras have 5 DOF, the robot has 6 DOF, which constitute an 11 DOF tracking, fixating and grasping system.

Because the active CCD cameras and the robot can move independently or together, the active CCD cameras can observe freely an object in Σ_0.

According to the visual feed back information, the robot can track, fixate and grasp the object autonomously.

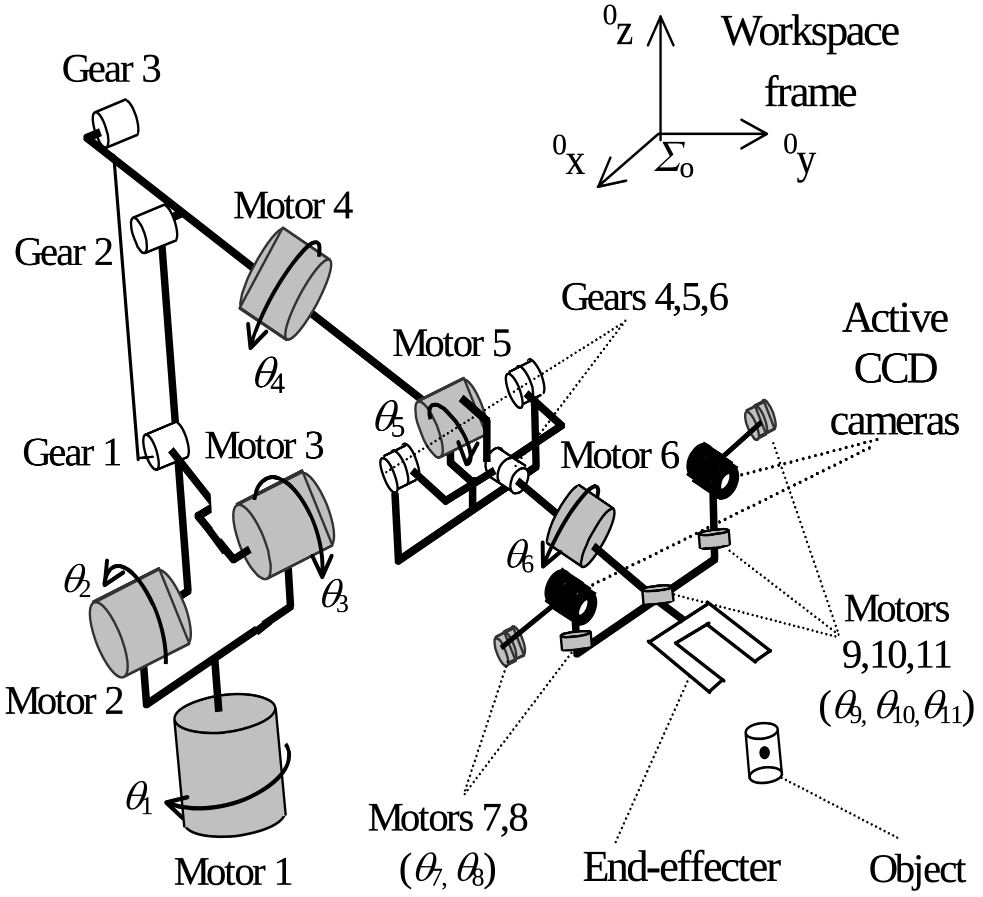


Figure 1. A robot system with active vision

3. Many-to-One Mapping Relationships

Figure 2 shows the projective relationships between the active stereo vision system and the object in Σ_o. Let p_i (i=1,2,3,…)be a feature point on the object. When the active stereo vision system tracks pi and its image coordinates are registered in the centers (o_l, o_r) of the left and right image planes of the two cameras respectively, the active stereo vision system is known as fixation on p_i.

Let $Q_i=[\theta_{i7},\theta_{i8,\dots,}\theta_{i11}]^T,$ $P_i=[x_{oi}, y_{oi}, z_{oi}]^T$ and $V_i=[^lu_{i1,}{}^lv_{i1,}{}^ru_{i1,}{}^rv_{i1}]^T$ be a joint angle vector of the active stereo vision system tracking p_i, a spatial representation vector of p_i in Σ_o and an image coordinate vector of p_i on the left and image planes, respectively. It is known from Fig. 2 that when p1 and p2 are visible to the CCD cameras, Q_2 and P_2 of p_2 identified by the active stereo vision system tracking on p_2 should be different from Q_1 and P_1. If another joint angle vector Q_3 is obtained by tracking p_3, p_1 and p_2 are still visible. P1 and P2 are not changed from that obtained by tracking on p_1 and p_2, respectively, despite the image coordinate vectors V1 and V2 change on the image planes of the CCD cameras. Therefore, there exist many combinations of Q_i and V_i which correspond to the same P_i, which means that P_i is invariant to the changing Q_i and V_i.

According to the projective geometry (7), Vi can be expressed as follows:

$$V_i=\varphi(Q_i,p_i), \quad (i=1,2,3,\dots), \tag{1}$$

where φ is a nonlinear projective function which maps the object and the joint angles on the left and right image planes of the CCD cameras. Therefore, Pi is specified as

$$P_i=\psi(V_i,Q_i), \quad (i=1,2,3,\dots), \tag{2}$$

where ψ is a nonlinear many-to-one mapping function, which denotes that the combinations of Qi and Vi correspond to Pi. On the other hand, it is known from Fig.2 that any combination of Qi and Vi should map to the same Pi, because pi is stationary feature.

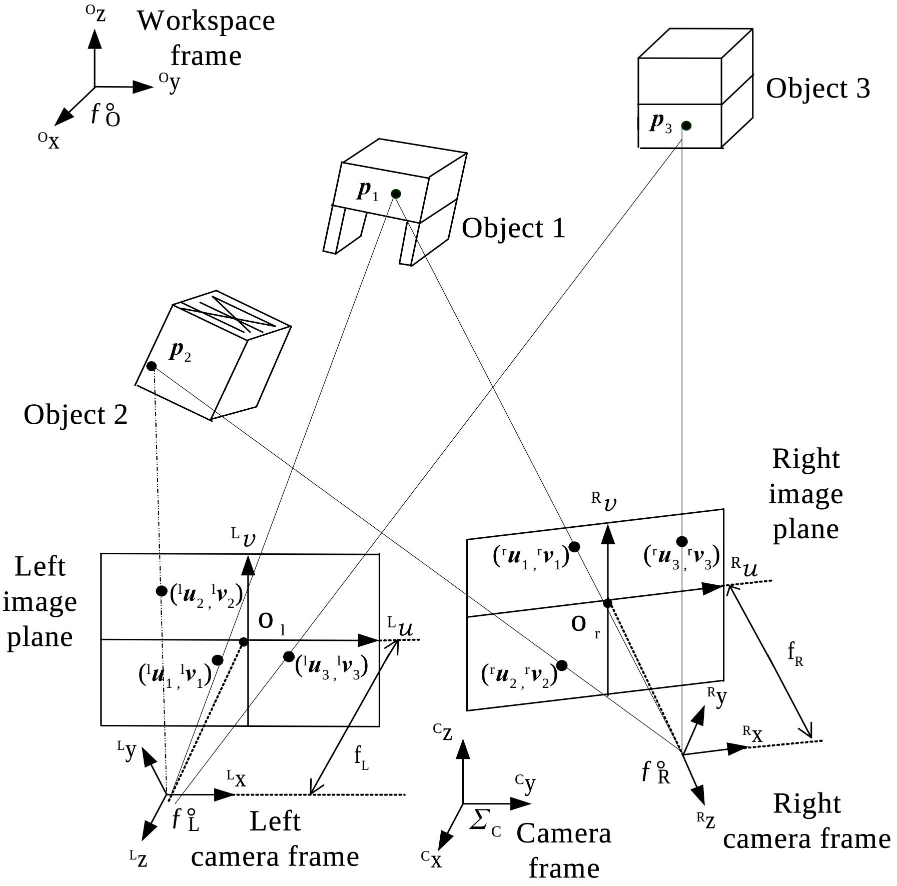


Figure 2. Projection and fixation relationships

4. Active Vision-Based Robot Control

4.1 Tracking and Fixating Control

It is known from Fig. 2 that when the active stereo vision system tracks pi, Vi is obtained for pi and Qi, we have from Equation (2),

$$P_i=\psi(V_i,Q_i), \tag{3}$$

where the active stereo vision system tracks p_i. When the active stereo vision system fixates p_i and the coordinate vector V_{Oi} of p_i corresponds to the centers (o_l,, o_r) of the image planes, then the desired joint angle vector Q_{Oi} which is necessary to bring V_{Oi} to V_{Oi} can be computed as follows:

$$P_i=\psi(V_{Oi},Q_{Oi}), \tag{4.a}$$

$$\text{or}\quad Q_{Oi}=\psi^{-1}(V_{Oi},P_i), \tag{4.b}$$

where ψ and ψ^{-1} are invertible functions which can be used to control the fixation on p_i, respectively. Because Pi has invariance, Q_{Oi} can be computed by combining Equation (3) with Equation (4),

$$Q_{Oi}=\psi^{-1}[V_{Oi},\psi(V_i,Q_i)], \tag{5}$$

therefore, Q_{Oi} is used to control the active vision system to fixate p_i.

4.2 Grasping Control

Figure 3 shows the configuration parameters of the active stereo vision system. Let d, s, l be the distance, height of the CCD cameras and diameter of the sphere coordinate system. Let $\theta_{i\alpha}$, $\theta_{i\beta}$, $\theta_{i\gamma}$ be the configuration angles of the active stereo vision system and the spatial coordinates of p_i in Σ_C be ${}^{C}P_i=[x_{ci}, y_{ci}, z_{ci}]^T$, respectively. When the active stereo vision system fixates p_i, ${}^{C}P_i$ can be computed by the triangular geometry relationships in Fig. 3

$$x_{ci} = l\cos(\theta_{i\gamma})\sin(\theta_{i\beta}), \tag{6.a}$$

$$y_{ci} = l\cos(\theta_{i\gamma})\cos(\theta_{i\beta}), \tag{6.b}$$

$$z_{ci} = l\sin(\theta_{i\gamma}). \tag{6.c}$$

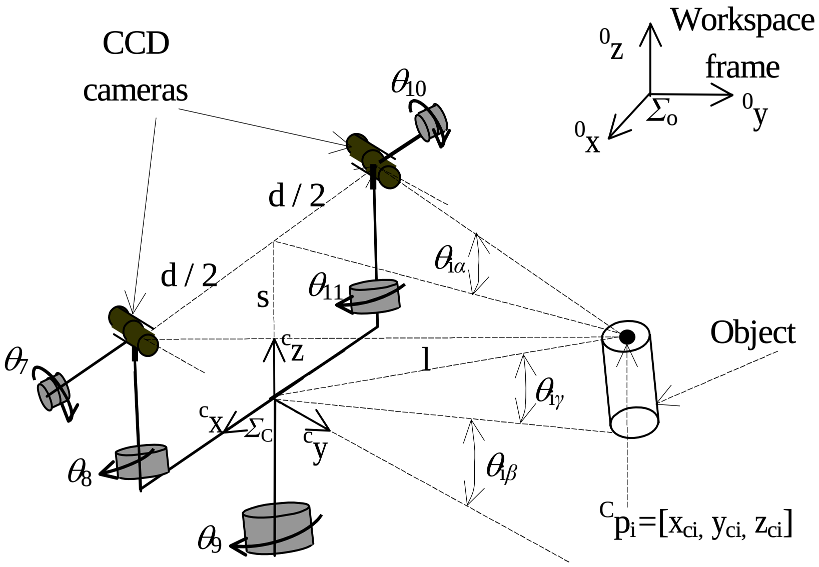


Figure 3. Configuration of the vision system

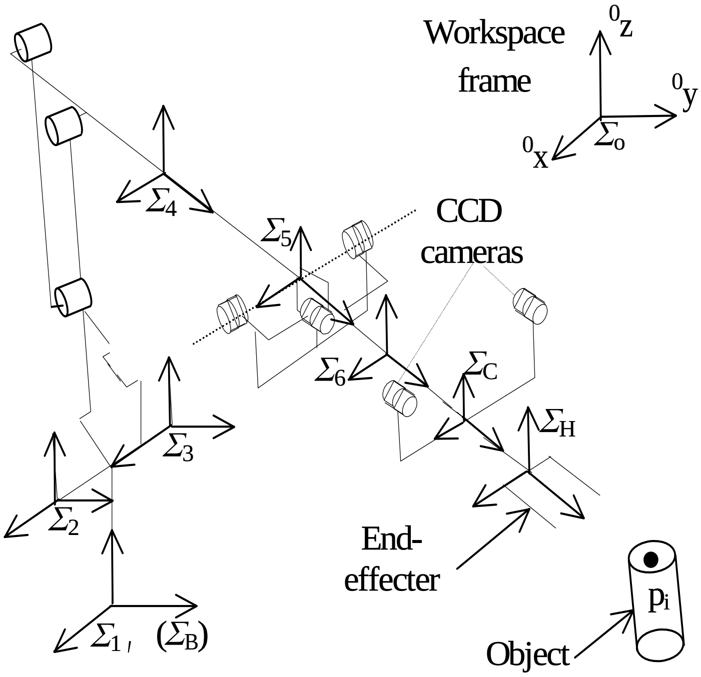


Figure 4. Frames of the active vision system

Figure 4 gives the joint coordinate frames of the robot joint system. In Fig. 4, let Σ_j(j= 1,2,…,6), Σ_B, Σ_H, Σ_C be a coordinate frame of ith robotic joint, base frame, coordinate frame of the end-effecter, camera frame, ${}^BP_i=[x_{bi}, y_{bi}, z_{bi}]^T$, ${}^HP_i=[x_{hi}, y_{hi}, z_{hi}]^T$, $P_i=[x_{oi}, y_{oi}, z_{oi}]^T$ be an Euclidian coordinate vector of p_i in Σ_B, Σ_H, Σ_O, respectively.

By homogeneous transformation relationship, ${}^Bp_i$ can be specified by

$$ {}^HP_i = {}^HH_C \bullet {}^CP_i, \tag{7.a} $$

$$ {}^BP_i={}^BH_6 \bullet {}^6H_H \bullet {}^HP_i, \tag{7.b} $$

$$ {}^OP_i={}^OH_B \bullet {}^BP_i. \tag{7.c} $$

where ${}^HH_C$, ${}^6H_H$, ${}^BH_6$, ${}^OH_B$ are the homogeneous matrixes from Σ_C to Σ_H, Σ_H to Σ_6, Σ_6 to Σ_B, Σ_B to Σ_O, respectively.

According to the robotic forward kinematics $\Lambda[\theta_r(t)]\quad R^{6\times1}$, we obtain

$$ {}^OP_{Hi}=\Lambda[\theta_r(t)], \tag{8} $$

$$ J[\theta_r(t)]=\partial\Lambda[\theta_r(t)]/\partial\theta_r(t), \tag{9} $$

$$ {}^O\dot{P}_{Hi} =J[\theta_r(t)]\dot{\theta}_{r(t)}, \tag{10} $$

where ${}^OP_{Hi}$ is the original coordinates of Σ_H in Σ_O, $J[\theta_r(t)]\quad R^{6\times1}$ is a Jacobian matrix of the end-effecter, $\theta_r(t)\quad R^{6\times1}$ is a reference joint angle vector of the end-effecter. Therefore, we have

$$ \dot{\theta}_{r(t)} =J^{-1}[\theta_r(t)] \bullet {}^O\dot{P}_{Hi}\, . \tag{11} $$

When the sampling period of the robot joint control system T is every minute, it is suitable that using $\dot{\theta}_{r(k)}=[\theta_r(k+1)–\theta_r(k)]/T$ to replace $\dot{\theta}_{r(k)}$ at time t=kT. Therefore, Equation (11) is discreted by

$$ [\theta_r(k+1)–\theta_r(k)]/T = J^{-1}[\theta_r(k)] \bullet {}^O\dot{P}_{Hi}\, , \tag{12.a} $$

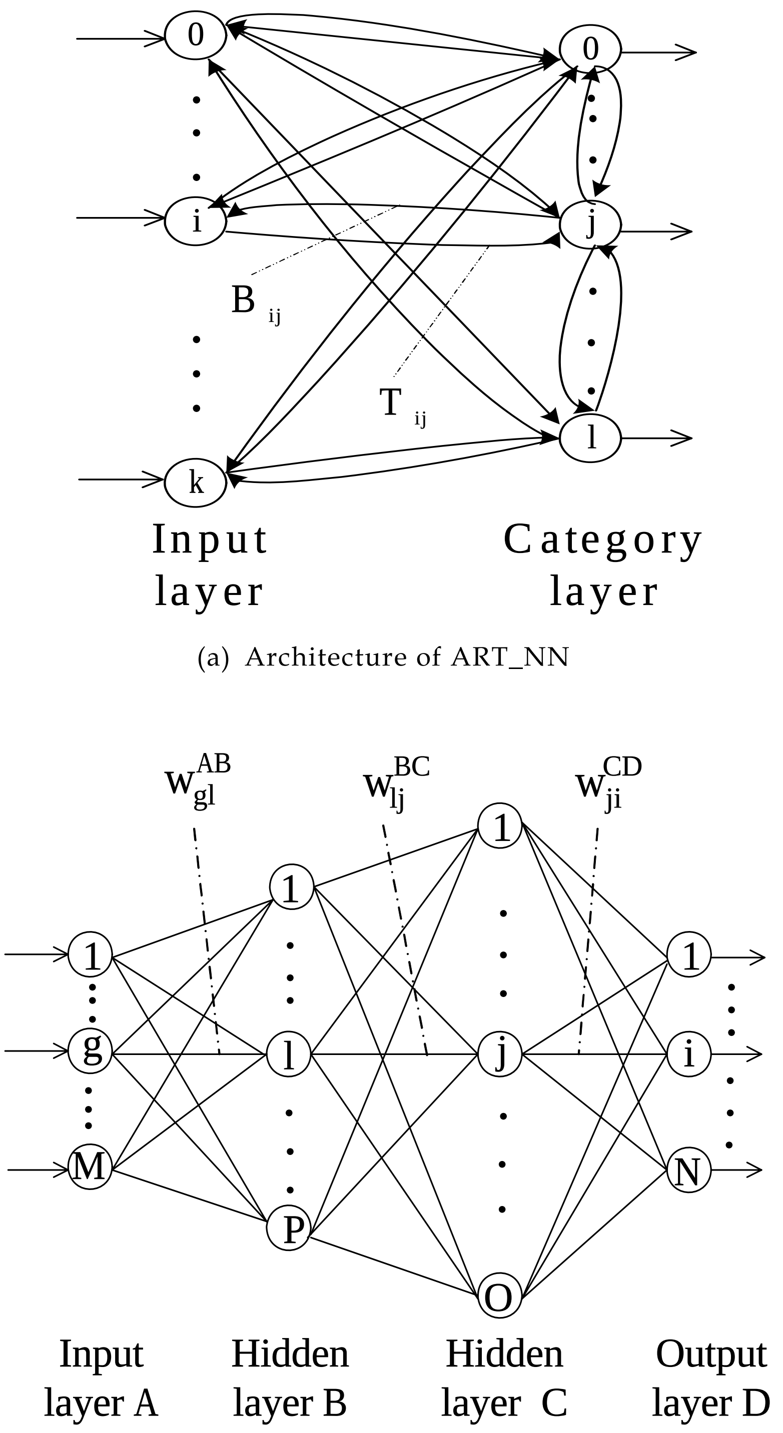
$$ \text{or } \theta_r(k+1)= \theta_r(k)+TJ^{-1}[\theta_r(k)] \bullet {}^O\dot{P}_{Hi}\, , \tag{12.b} $$

where ${}^O\dot{P}_{Hi}(k)=[{}^oP_{Hi}(k)–{}^oP_{Hi}(k-1)]/T$, $\theta_r(k+1)$ can be used to control the robot joint angles. When ${}^OP_{Hi}={}^OP_i$, the end-effecter can grasp the object.

5. Visual Robot Learning Control System

5.1 Visual Learning Control System

In order to obtain the nonlinear many-to-one mapping function, ART_NN are combined with FF_NN to learn ψ defined in Equation (3). The architecture of ART_NN, FF_NN and the vision-based robot control system based on ART_NN and FF_NN are showed in Figs. 5 and 6, respectively.



(a) Architecture of ART_NN

(b) Architecture of FF_NN

Figure 5. Architectures of ART_NN and FF_NN

5.2 Learning of ART_NN and FF_NN

In Fig. 5, T_{ij}, B_{ij}, w_{gl}^{AB}, w_{lj}^{BC} and w_{ji}^{CD} are weights. The self-adaptive resonance algorithms for ART_NN and the learning algorithms for FF _NN are omitted. In Fig. 6, the ART_NN require two types of inputs V_i and Q_i, where V_i corresponds to the image coordinates of p_i on the image planes of the CCD cameras, and Q_i is the joint angle coordinates corresponding to the CCD cameras tracking p_i. The ART_NN clusters V_i into classes within the category layer. The class number in each category layer depends on a vigilant parameter which is a real number between zero and one.

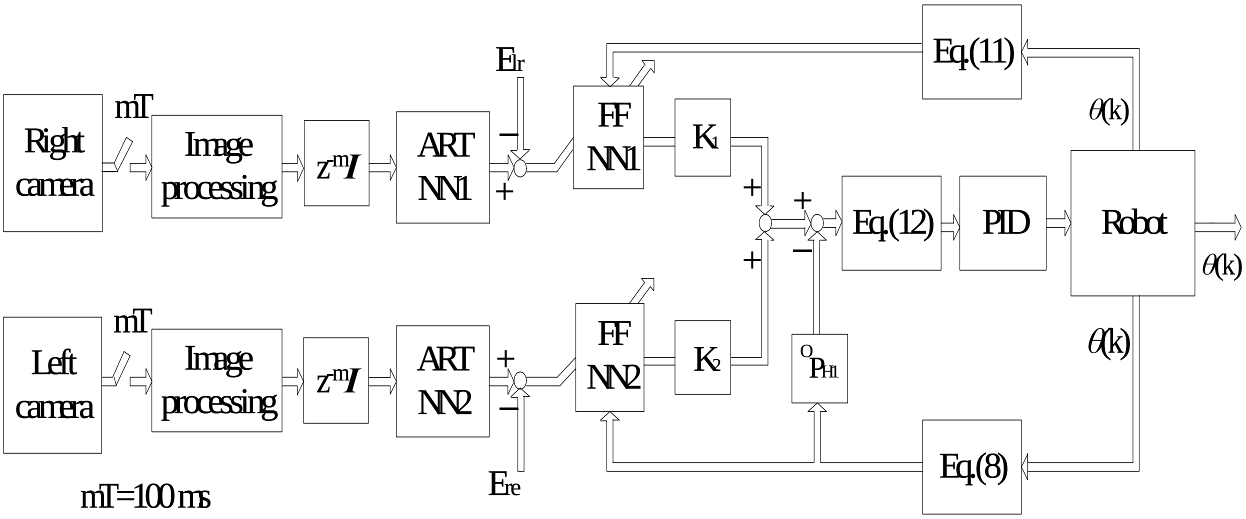


Figure 6. A Robotic Learning Control System

In Fig. 6, $\boldsymbol{K}_1$ and $\boldsymbol{K}_2$?R$^{6\times6}$ are the coefficient matrixes which were specified empirically. $\boldsymbol{E}_{le}$ and $\boldsymbol{E}_{re}$?R$^{6\times6}$ are the differences between the two learning trials, respectively, and the PID controller is used to obtain the joint servoing control with high accuracy.

6. Simulations

To evaluate the validity of the active stereo vision-based robotic tracking, fixating and grasping in the unknown environment, the simulations are carried out using the models of the active stereo vision system installed in the end-effecter.

For controlling the robot to track, fixate and grasp the object, first, ART_NN1 and ART_ NN2, FF_NN1 and FF_NN2 learn the many- to-one functional mapping relationships by generating 10000 random pairs of V_i and Q_i signals corresponding to p_i. The ART_NN1 created 500 classes for the inputs from the right CCD camera, and the ART_NN2 also created 500 classes for the inputs from the left CCD camera.

The spatial coordinates of pi are computed by using in the tracking, fixating and grasping control loop.

The simulation results denote that the errors for all of the three components of the spatial representation converged to within 2% of its dynamic range. These results show that the learning of ART_NN and FF_NN is fast, convergent and the end-effecter can also arrive at the position of the object.

7. Conclusions

The following conclusions are drawn from the above experiments:

(1) There exist many-to-one mapping relationships between the joint angles of the active stereo vision system and the spatial representations of the object in the workspace frame.

(2) ART_NN and FF_NN can learn the mapping relationships in an invariant manner to the changing joint angles. The vision and joint angle signals of the active vision system corresponding to the object correspond to the same spatial representation of the object.

(3) The present approach was evaluated by simulation using the models of an 11 DOF active stereo vision system and the simulation confirms that the present approach has high robustness and stability.

8. Acknowledgement

This research was funded by the initial foundation of China Education Ministry for Scholars Returned China from abroad (project no.: 2002247) and the Intelligent Systems research initiative of Deakin university.

9. References

Bernardino A., Santos-Victor, Bin- ocular Tracking: Integrating Perception and Control, IEEE Transactions on Robotics and Automation, (1999-12), Vol. 15, No. 6, pp. 1080 – 1093.

Sumi, Kawai, Yoshimi, 3-D Object Recognition in Cluttered Environments by Segment-Based Stereo Vision, International Journal of Computer Vision, (2002-1), Vol. 46, No. 1, pp. 5-23.

Srinivasa N., Rajeev S., SOIM: A Self-Organizing Invertible Map with Applications in Active Vision, IEEE Transactions on Neural Net- works, (1997-5), Vol. 8, No. 3, pp. 758-773.

Barnes N.M., Liu Z.Q., Vision guided circumnavigating autonomous robots, *International Journal of Pattern Recognition and Artificial Intelligence*, (2000), Vol. 14, No. 6, pp. 689-714.

-III-

Navigation

Managing Limited Sensing Resources for Mobile Robots Obstacle Avoidance

Juan Carlos Alvarez, Rafael C. Gonzalez, Diego Alvarez & Antonio M. Lopez

1. Introduction

Obstacle avoidance is a sensor-based task designed to steer a mobile robot towards intermediate goals while avoiding unexpected local obstacles in the robot's surroundings. Responsiveness requires that the task is computed in short time intervals, fast enough to deal with the robot inertia and dynamics limitations. Simultaneously it has to guarantee detection by gathering enough information to allow the robot to react adequately to any potential danger.

However, throughput and detection guarantee are opposite demands. If we try to assure the latter by augmenting the scanned area in the robot's surroundings, it will increase the time taken to process the sensor data, proportionally decreasing the reactivity to the external world. On the other hand, scanning smaller areas can endanger the robot if some possible trajectories of motion are not swept by the sensors (e.g. the tunnel–vision problem).

The previous trade-off can be addressed in different ways. Some obstacle avoidance algorithms are designed supposing that fixed sensorial information of the robot's surroundings will be available when needed, e.g. range measures in a circular or rectangular robot-centered area (Lumelsky and Skewis, 1990), (Ulrich and Borenstein, 2001). Furthermore, others restrict the required information to the limits imposed by the robot's actual motion, its dynamics and kinematics (Brock and Khatib, 2000).

But the assumption that certain sensorial information will be available when required is not always realistic. For example, the particular field of view of conventional range sensors such as stereovision, sonar or laser, cannot adjust adequately to the specific perception requirements (Laubach and Burdick, 2000). Or the time needed for sensor data processing can have an impact on the real-time motion performance, making some sense modalities more recommendable than others (Kelly and Stentz, 1998). Finally, the availability of each specific sensing device can be an issue whenever the robot's task solicitations exceed the robot's computing capabilities, for example in robots with complex missions or with massive sensorial information. For these cases, treating sensing as an isolated phenomena leads to bottlenecks both oinn computational demand and in real-time performance.

Our approach to the problem of making the required information available when needed is based on two ideas: 1) to define what the (minimum) information required is in order to guarantee the safety of motion for a given motion strategy, and 2) to use sensor management to obtain only the required info. Scanning only the essential area around the robot helps us to adapt to the real time demands about motion reactivity, and managing the sensor allows us to obtain only the required information, shortening the processing time, by combining the actual usable sensors. The proposed solution has been applied to a real robot motion problem, whose formal formulation is discussed in Section 2. Section 3 is devoted to the analysis of the sensor information requirements for a given robot and motion control policy, in relation to motion safety and performance. Then, and for a given sensor model, we design a sensor management strategy able to gather the required information. In Section 4 we present experiments designed to focus attention on specific points of the problem. Experimental results in realistic environments, in Section 5, show the feasibility of the proposed strategy, allowing a mobile robot to move at high average speeds and with high reactivity (ten cycles per second).

2. Problem Definition

The problem stated in the previous section has two main components: the robot and sensors capabilities and the intended motion control strategy. In the following section both aspects are explained, leading to a specific problem formulation.

2.1 Definitions: Robot and Sensors

We address the problem of a robot moving in a planar Euclidean environment. The robot has a circular shape with radius rr, and its configuration is represented by the coordinates of its center $C_i = (x, y)$ and its orientation θ, relative to a fixed coordinate system. The robot orientation θ is collinear with its instantaneous velocity vector.

The robot is equipped with range sensors, which can sweep an area in front of it called Field of Regard, see Figure 1-left. It is defined by the aperture angle (field of view, ρ), and its maximum range (depth of field, d_f). The sensor is able to pan, represented by a rotation angle γ relative to the mobile frame. The "frontal swept line" (FSL) is chosen to be perpendicular and symmetrical with respect to the mobile frame, and it is defined by parameters (d_f, ρ, γ). This model is applicable for commonly used sensors such as ultrasonic and stereovision cameras. Rotating ultrasonic sensors and stereovision systems are usually mounted over pan and tilt devices permitting the panning operation; for fixed rings of ultrasonic sensors we have a "discrete panning" result of firing only a certain group of sensors.

Notice that, depending on the robot's motion policy and the sensor field of regard, possible motion trajectories are not swept by the sensors, see Figure 1-right. This is known as a tunnel–vision problem, and from that point of view, we are interested in the real-time selection of these three parameters (d_f , ρ, γ) in order to avoid such danger without increasing the sensing effort, while maintaining good reactivity and motion performance.

2.2 Safety, Control Policy and Robot Dynamics

Two robot characteristics will affect the design of the sensing strategy: its mobility limitations and its motion control strategy. Mobility limitations come from the robot's mechanical configuration, dynamics, and other physical limits. Usual steering mechanisms impose a nonholonomic restriction,

$$x = v \cos\theta \qquad y = v \sin\theta \qquad \theta = \omega \tag{1}$$

with v(t) being the robot's forward velocity, and ω(t) its rate of change in orientation or turn velocity. Robot actuator dynamics limit robot velocities (v, ω) both in magnitude [vM, ωM] and in rate of change or acceleration, [aM, αM]. A convenient model for robots with a synchronous steering mechanism is (Alvarez, et al., 2001):

$$v + kpv = uv \qquad \omega + kh\omega = u\omega \tag{2}$$

kp and kh are constants and (uv, uω) velocity references to the robot actuators. The robot's trajectory for a given control command (uv, uω) can be computed by integrating equations (2) and (1).

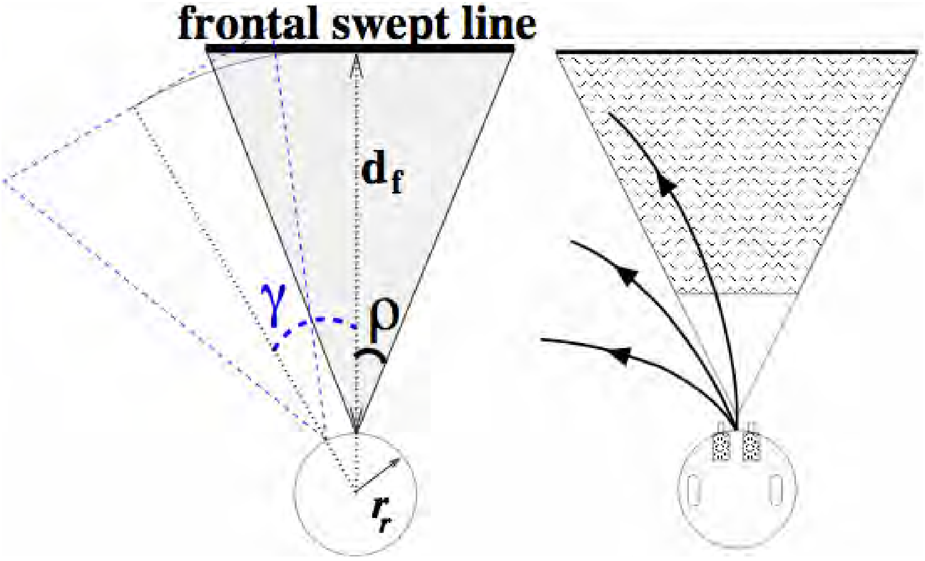


Figure 1. Problem formulation. (Left) The sensor Field of Regard is defined by three parameters: the field of view ρ, the depth of field d_f , and the field orientation γ. (Right) Tunnel vision problem: a fixed Field of Regard can be insufficient to sweep all the possible robot trajectories, with the corresponding safety risk

The motion control strategy is responsible for how the references (u_v, u_ω) are selected from an initial robot position C_i and state (v_0, ω_0), in order to reach an intermediate goal Ti with some optimal criteria (such as minimum time). We are supposing that robot motion decisions are computed in short and fixed time periods T_{cyc}. At every period T_{cyc} a new intermediate goal to reach T_i is provided to the robot (we can think of T_i as the result of a motion planning module). For instance, a reasonable strategy for an "emergency stop" is (u_v, u_ω) = (0, 0). For the rest of the operations we will assume that controls are obtained with a "maximum turn" strategy:

$$u_v = K_{vdobs} \qquad u_\omega = K_\omega(T_i - C_i) \tag{3}$$

with constants K_v and K_ω tuned to maximize velocities, and d_{obs} the sensed distance to the closest obstacle in the intended robot trajectory.

2.3 Overcoming Limited Sensing with Sensor Management

Sensor Management deals with multisensor systems operating in real time, and "how to manage, co-ordinate and integrate the sensor usage to accomplish specific and often dynamic mission objectives" (Luo, et al., 2002). It includes the individual control of each sensor such as direction, pointing, change in frequency, power level, etc. The goal is to reduce unnecessary use of sensors through adequate use of the available resources.

The expected benefit, in our problem, is to be able to deliver the required information when needed (just-in-time). Precisely, the sensor field of regard will be extended in order to maintain motion performance with safety guarantee, by means of a sensor selection strategy. In order to minimize time data processing, such strategy will depend on the actual robot motion conditions at every moment. The solution will be the outcome of an analysis aimed to calculate the minimum sensorial information (field of regard size) needed to assure safety with the proposed control policy.

3. Strategies for Just-In-Time Obstacle Detection

In this section the algorithmic foundation of the proposed solution will be explained. It leads to the definition of a "virtual sensor" able to fulfill the requirements of obstacle avoidance in real time. The detail of the following mathematical analysis has been reported in (Alvarez, et al., 2002).

3.1 Motion at Constant Velocities

Let us consider the safety of motion at constant velocities, that is, for straight and circular motion. When the robot moves along a straight line, the relation between sensor depth and aperture (d_f , ρ) has to be sufficient to guarantee its safety. That is accomplished if the whole area traversed by the robot is previously swept by the FSL. In other words, for a circular robot of radius r_r and a sensor of aperture ρ, the field of regard depth satisfies:

$$d_{f1} = r_r / \tan\rho \tag{4}$$

If d_f is smaller than d_{f1}, safety is not guaranteed. If it is greater, obstacles out of the robot's path will be detected as dangerous, and control (3) will cause an unnecessary reduction of velocity.

A second condition is that sensor field depth has to be large enough to let the robot stop safely –fast enough– if an obstacle is detected in front of it. This distance will be greater as the robot's inertia is increased. Let us call an "emergency stop" the maneuver aimed to completely stop the robot, whatever its initial state $(v, \omega) = (v_0, \omega_0)$ is. Such maneuver will be implemented by sending the command references $(u_v, u_\omega) = (0, 0)$.

The distance traversed before the control command halts the robot, rd, depends on its dynamics and the response time of the robot's control system, t_r,

$$r_d = v_0 t_r + x(\infty) = v_0 t_r + v_0/k_p \tag{5}$$

being $x(\infty) = v_0/k_p$ the consequence of dynamics, calculated by integrating equations (2) and (1) (Alvarez, et al., 2001). This magnitude establishes a lower limit to the depth of field, $d_f \geq {}_{rd}$, which depends on the robot current status v_0. The worst case analysis $v_0 = v_{max}$ allows us to compute, off–line, a secure distance to stop, for every initial state $(v_0, \omega = 0)_k$:

$$d_{f2} \geq r_{dmax} \tag{6}$$

In conclusion, when moving in a straight line, robot safety and smooth motion is guaranteed by equations (4) and (6). This analysis is condensed in Figure 2.

For constant velocity references (u_v, u_ω) the robot moves along a circular trajectory of radius $r_g = u_v/u_\omega$. following the model defined by (1) and (2). Thus, for a sensor field of view fulfilling conditions (4) and (6), we can identify potentially risky situations, see Figure 3.

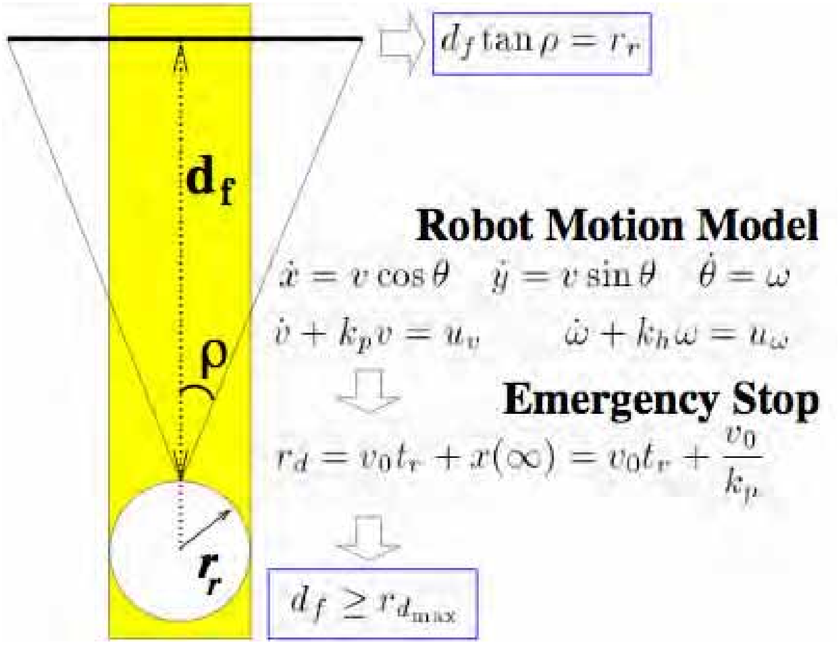

$d_f \tan\rho = r_r$
$\mathbf{d_f}$
Robot Motion Model
$\dot{x} = v\cos\theta \quad \dot{y} = v\sin\theta \quad \dot{\theta} = \omega$
$\dot{v} + k_p v = u_v \quad \dot{\omega} + k_h\omega = u_\omega$
ρ
Emergency Stop
$r_d = v_0 t_r + x(\infty) = v_0 t_r + \frac{v_0}{k_p}$
r_r
$d_f \geq r_{d_{\max}}$

Figure 2. Minimum sensory requirements when a robot moves in straight line. They take into account the robot's dimensions, equation (6), and its dynamics, equation (9)

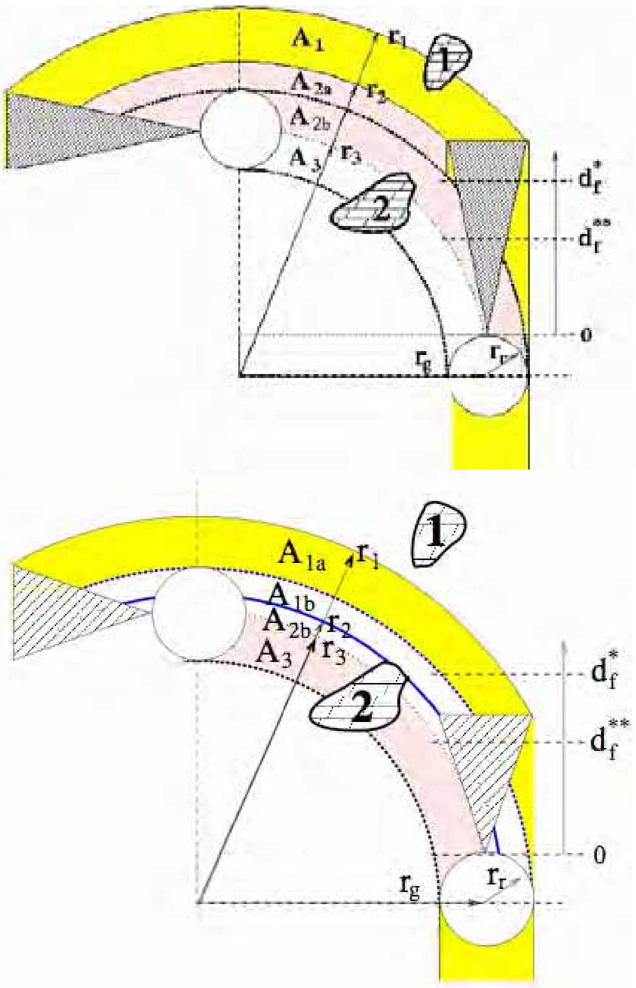


Figure 3. Robot turning with constant velocities. Its sensor Field of Regard fulfills conditions (4) and (6). (Left) Case when $d_f \succ d_f^*$ and the tunnel vision problem: the robot moves where the sensors did not reach. (Right) For $d_f^{**} \succ d_f \succ d_f^*$ the situation is better, but still there are blind zones

The FSL swept action causes three different situations: a zone A_1 of “unnecessary deceleration”, a zone of “lateral detection” comprised between sectors of radius r_3 and r_2 (A_2), and a “blind zone” A_3, where obstacles are not detected.

All of them have to be eliminated or reduced as much as possible, setting new conditions on the selection of the sensor field depth d_f . The objective is to sweep in advance with the FSL the exact zone that will be traversed by the robot, as we did in straight-line motion. Notice in Figure 3 that smaller depths d_f reduce them, in opposition to the demand of larger depths–at least fulfilling condition (6)–that increases the options to react to unexpected obstacles. This trade-off depends on two factors (Alvarez, et al., 2002):

1) There is a certain limit depth, d_f^* , which eliminates lateral detection of non-dangerous obstacles (zone A_{2a} in Figure 3-right),

$$d_f^* = 2\sqrt{(r_{rrg})} - r_r \qquad (7)$$

This value d_f^* is an upper limit to d_f .

2) Another limit value exists d_f^{**} such as smaller depths $d_f \leq d_f^{**}$ completely eliminates lateral detection (zone A_2),

$$d_f^{**} = \sqrt{2}\sqrt{(r_{rrg)}} - r_r \qquad (8)$$

A further reduction of blind zones A_3 is only possible by choosing $d_f < d_f^{**}$.

As illustration Figure 4 shows an experiment with a Nomad–200 where an obstacle in a “lateral detection” A_2 zone appears abruptly in the robot field of view. It triggers an emergency stop condition because the robot dynamics do not allow any other avoiding maneuver, and the overall motion performance is negatively affected (Alvarez, et al., 2001).

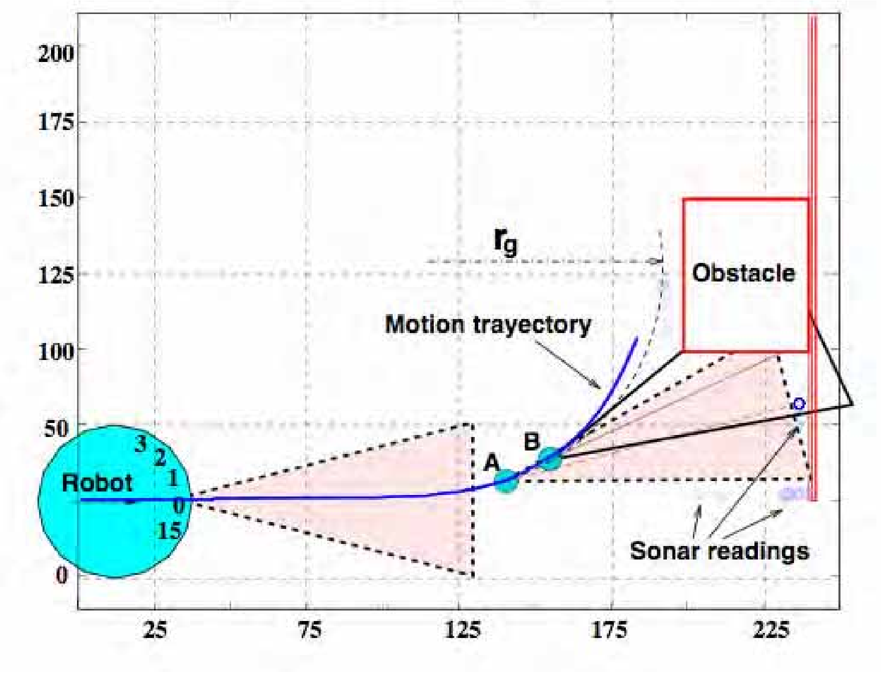


Figure 4. Stop maneuver triggered by an “A_2 zone” obstacle in a real experiment; top view. From point A to point B the robot is turning at maximum speed, and the sensor is measuring a maximum distance d_f ; in B the obstacle corner comes into view, and an abrupt range measure is read, leading to an emergency stop condition (and eventually a collision)

3.2 Improving Detection by Sensor Panning

It is clear that the previous static design is not enough to eliminate the risk of crossing areas of lateral detection or blind zones. An alternative to reduce such dangers is to dynamically accommodate the scanned area to the instantaneous robot motion state. Let us define the scanning swept line by the pair (d_f , ρ), and let us denote γ to the angle, measured from the main robot axis, to direct a new scan action.
We will calculate γ assuming that the scan line has the previously calculated shape d_f , and adding the panning (orientation selection) capability. A first solution consists in finding the sensor orientation γ so that the blind zone A_3 diminishes the maximum possible, or totally disappears. The idea is to make use of the whole FSL to cover the intended robot trajectory. It can be obtained by solving the equation:

$$A \sin \gamma + B \cos \gamma = 1 \qquad (9)$$

being A and B being constants for a given field of regard (d_f , ρ). Its solution reveals the greater blind-zone reduction that can be achieved by panning (Alvarez, et al., 2002). Notice that A_2 zones may still appear. To avoid them, an alternative design would be necessary which allows us to modify the three sensor parameters on-line, a much more complicated condition to apply in real sensors.

3.3 Improving Detection by Sensor Selection

Another option is to use the combination of sensors necessary to guarantee robot safety, that is, to reduce sensing to the areas traveled by the robot if a stop maneuver is initiated. We want to check whether the robot will hit an obstacle along that trajectory or not, subject to sensor availability at each cycle. First, we must determine which set of sensors, from the available ones, minimize the difference between the area covered by sensors and the area which has to be inspected. We propose a greedy algorithm to perform this task:

1) Calculate the trajectory followed in an emergency stop initiated after the next command has been executed.
2) Determine the set of available sensors to inspect the desired path and initialize the set of selected sensors to the empty set.
3) From the set of available sensors, select the sensor that covers the points farthest from the stopping trajectory with the best resolution and configure it so that it covers the maximum area.
4) Remove the selected sensor from the set of available ones and add it to the set of selected sensors.
5) Remove from the path to be inspected the area scanned by the selected sensor.
6) Go back to step 3 until the area to be inspected is empty, or no new sensor can be selected.
7) f there is an area that can not be inspected return a not safe condition,
8) Otherwise fire selected sensors and check if all readings are compatible with the minimum free area configured for each sensor.

Greedy algorithms are suboptimal, but in this case, they may give a good approximation in a short time period. To compute the area to be scanned with the sensors the robot motion equations can easily be integrated to obtain the trajectory of the robot until it stops. The stop commands sequence can be arbitrarily chosen every control cycle, and even different combinations of commands may be tested. Figure 5 shows an example of stopping trajectory. Figure 6 shows the result of this covering algorithm in the same environment where the experiments will be carried out (see following section).

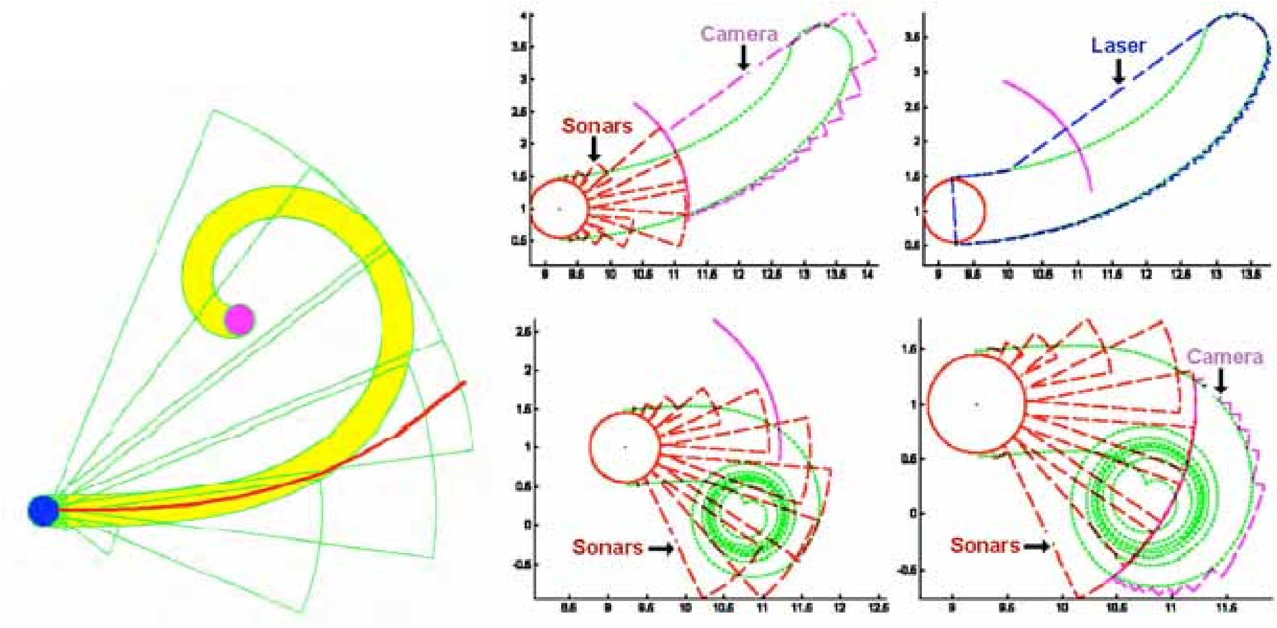


Figure 5. Example of a stopping trajectory and a set of sonars to inspect it. (Left) The continuous line represents the intended trajectory, while the thick path shows the area traversed by the robot for a given stopping procedure. (Right) Different runs of the covering algorithm, for different combinations of stopping trajectories and available sensors

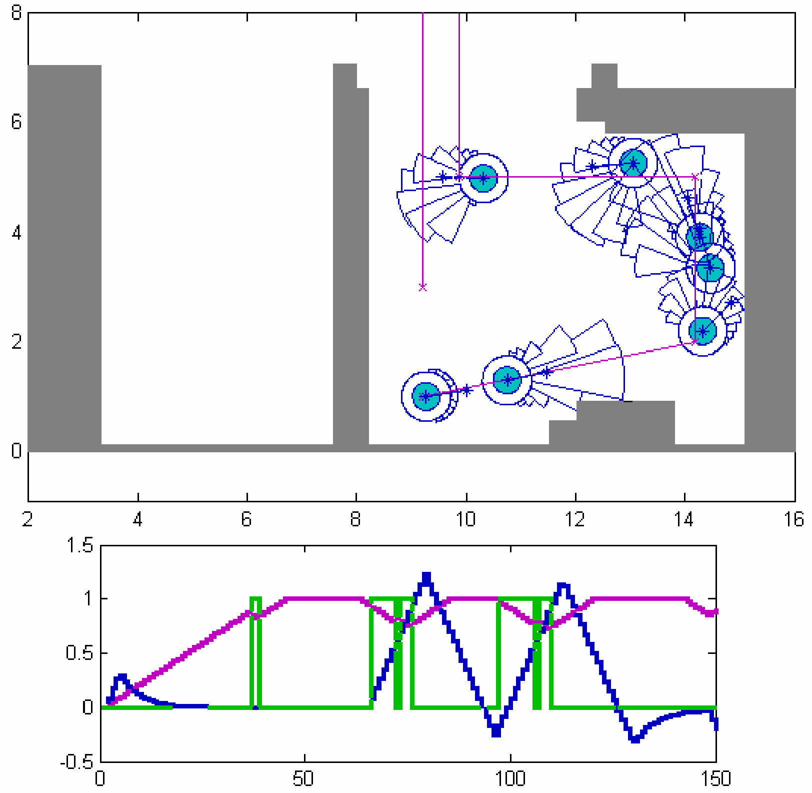


Figure 6. Sensor selection: the minimum set of individual sensors that are able to cover the emergency trajectory within a cycle time. (Above) Robot motion behavior and set of sensors fired in each iteration. (Below) Time velocity profiles: forward and turn velocities (m/s vs. s)

4. Experimental Setup

The previous analysis has been applied to the design of an obstacle avoidance module for a RWI B-21 mobile robot at the University of Oviedo. The local sensing strategy was implemented with a ring of ultrasonic range sensors of 24 equidistant units. Even though it limits the pan angle election to values 15 degrees, different combinations of simultaneous firing of contiguous sonar were used to implement different scan window shapes, as represented in Figure 7.

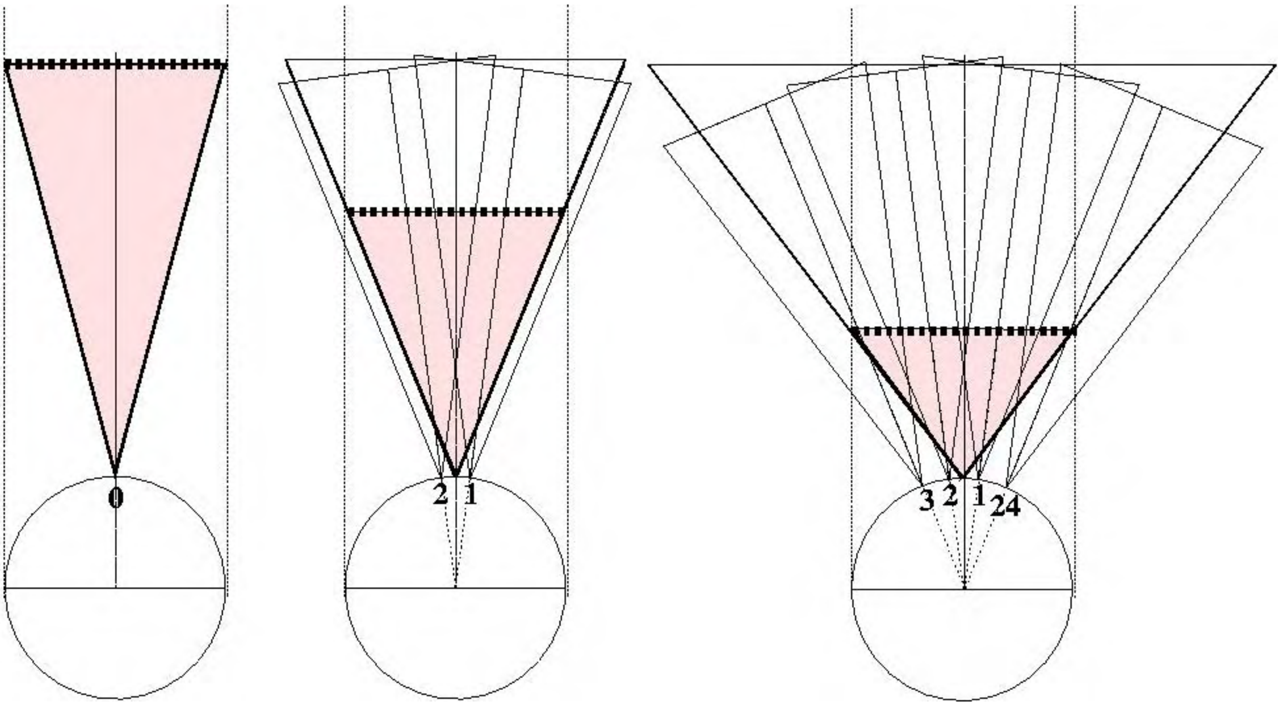


Figure 7. (Left) Virtual sensor implemented with a unique frontal ultrasonic sensor of aperture 15 deg. (Center) Virtual sensor implemented with two sonar sensors, located at ±7.5 degrees from the main robot axis, robot B-21, aperture 22.5 deg. (Right) Virtual sensor implemented with four sonar sensors, located at ±7.5 degrees from the main robot axis, robot B-21, aperture 37.5 deg

Similar experiments to those made with the Nomad–200, as reported in (Alvarez, et al., 2001), have been carried out with the faster and heavier B21 robot, applying the proposed panning strategy to avoid the previous inconvenient situations without loss of reactivity.

Every control cycle the robot is fed with another sub-goal to reach in its workspace, with the condition that the straight line connecting each sub-goal is supposed to be free of obstacles. In our first setup, the sub-goals happen to be located in the corners of a 2x2m square, see Figure 8. The robot does not know which the next sub-goal will be until it reaches the current one. The obstacles labeled in Figure 8 have to be detected and avoided in real time with minimum loss of motion performance.

The B21 robot has a maximum forward velocity of 1 m/s. With a 45 degree beam, the df to cover condition (4) is 64 cm. As this distance is similar to that needed to stop, $d_f \approx d_f^*$, the limiting values (7) and (8) are 59.5 cm and 34.3 cm respectively. The first conclusion is that, with the selected height of d_f =64 centimeters, the sensor will not present lateral detection zones.

When turning, the proposed sensor panning strategy, equation (9), suggests firing the sonar located at 32.5 degrees, which can be physically accomplished with sonar numbers 3 and 4. It eliminates blind zones, and the obstacle is detected with enough time to reduce speed and, eventually, plan a detour.

Figure 8 shows the result of the proposed motion strategy with sensor panning in a cluttered environment. The robot moves counter clockwise from the lower left corner point. Five unexpected obstacles lie in the sub-goals surroundings. When obstacle 3 is not present, obstacle 1 is especially prone to fire emergency stop conditions as described before. Such possibility was eradicated using the panning strategy: the 3-4 sonar pair was fired, allowing smooth obstacle detection and an equivalent velocity reduction proportional to the distance to the obstacle. It produces the flat velocities profiles in Figure 8-right, resulting in a good motion performance and high average velocities.

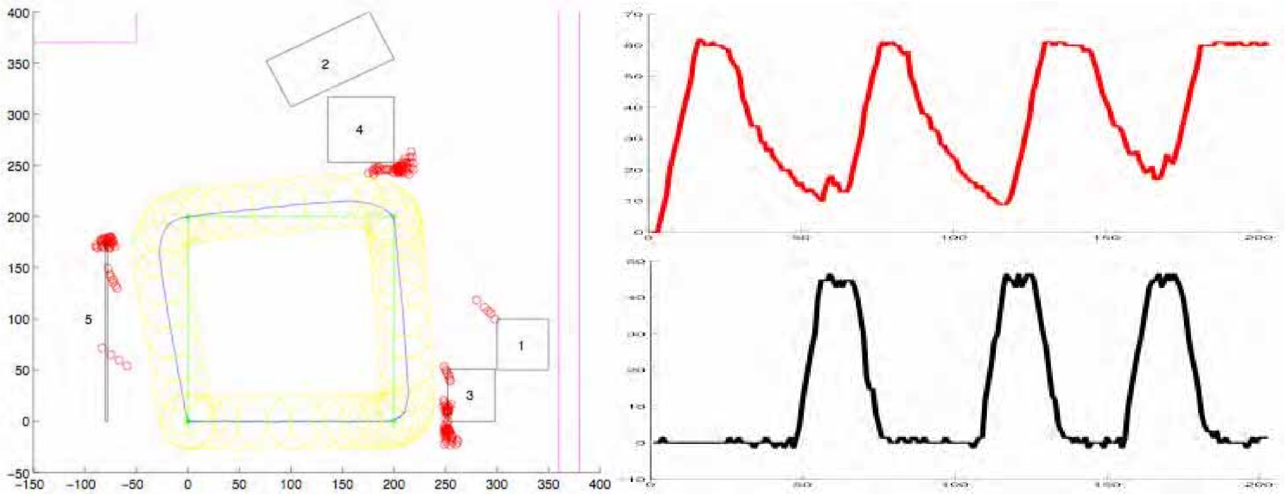


Figure 8. Experiment 4a, in a cluttered environment; top view. (Left) Unexpected obstacles detected during motion, as the sonar readings show, shorten the robot's trajectory. (Right) Robot forward and turn velocities

Table 1 summarizes the experimental results. The B-21 was programmed with the same dynamic parameters that the Nomad-200 in (Alvarez, et al., 2001), that is, 60cm/s and 45deg/s as maximum velocities, and 25cm/s2 and 50deg/s2 as maximum accelerations. Experiments labeled 4b and 4c correspond to the same setup as 4a, but removing obstacles 4 and 3 respectively. Experiment 1 implements a point-to-point stop-and-turn strategy (notice the total path length of 8 meters), that is safe but of low performance.

		Path Length (cm)	Time (s)	(vM, aM) (cm/s, cm/s2)	(ωM, αM) (deg/s, deg/s2)
Nomad	Exp 1	800	31.6	(60,25)	(45,50)
	Exp 4a	828	22.3	(60,25)	(45,50)
B21	Exp 4a	867	21.0	(60,25)	(45,50)
	Exp 4a	910	19.1	(60,25)	(45,50)
	Exp 4a	939	18.0	(60,25)	(45,50)

Table 1. Comparative experimental results for two mobile robots: a Nomad-200 of Nomadic and a B-21 of Real World Interface

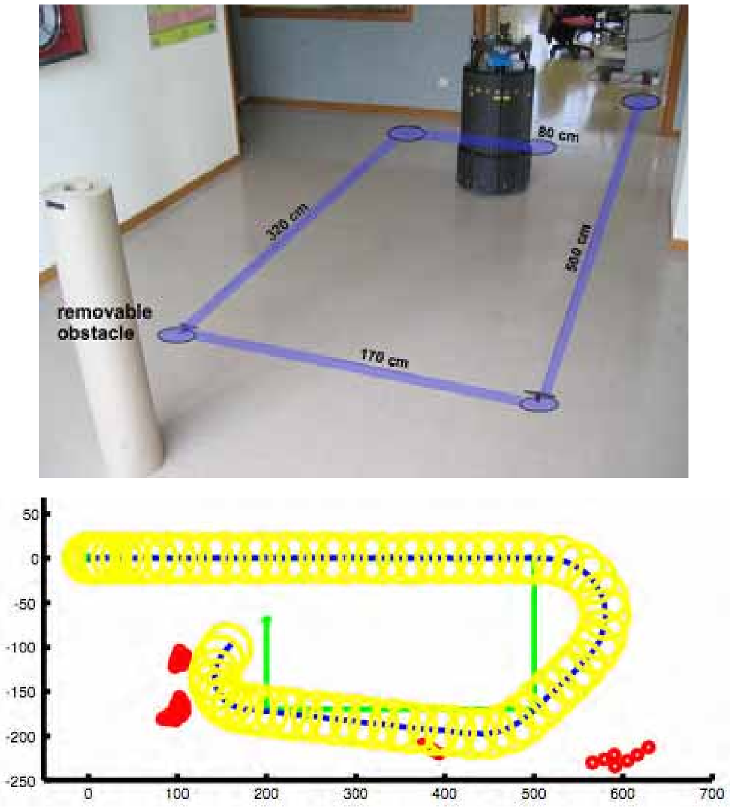


Figure 9. Experimental setup. (Above) The robot has to move as fast as possible between four consecutive goals in an unknown environment. Obstacle detection is implemented by means of two sonars, with a field of view, ρ=22.5 deg and d_f =64 cm, and a control period of 4 cycles/s. The removable obstacle will be located in different situations. (Below) Typical result, top view: the sensor readings inside the scanned window (dots) produce lineal robot decelerations following the strategy in equation (3) (axis in cms)

Figures 10 and 11 show more experimental results, with a different setup plotted on Figure 9. Figure 10 illustrates a situation of unnecessary deceleration, that is solved with the panning strategy. The movable obstacle is located in a zone of unnecessary deceleration A1a, which produces a velocity reduction in t = 12s. The same situation is produced by the wall on the opposite side. In the inferior figure the panning strategy is applied, and the sensors fired are those located at 15 degrees. The movable obstacle in not detected and no deceleration is commanded. A better operation also occurs in the vicinity of the mentioned wall. Figure 11 shows a situation of blind zone, with the consequent collision, and how it is solved by panning. The movable obstacle is located in a blind zone A3, which produces a collision in t = 15s. By firing sensors at γ=15 degrees, the obstacle is detected and a lineal deceleration is commanded. It changes the robot's trajectory enough to avoid the obstacle, and the robot is able to finish the mission.

In Figure 11-below, the covering algorithm is applied instead of the panning strategy. We defined a set of three possible stopping trajectories: to keep turning speed constant, and increasing turning speed to the left/right at maximum allowed rate. We have represented the scanned areas at different positions. In the beginning, the robot accelerates towards its first goal. At the point P_1, the robot decelerates to avoid a column close to the path. This situation is an unnecessary deceleration due to the use of sonar devices. The situation is repeated at the point P_4. As soon as a suitable stopping strategy is found (turning to the interior of the intended path), the robot begins to accelerate again. As the robot reaches P_3, the walls are relatively close. Due to the excessive area scanned by sonar because of its poor angular resolution some unnecessary decelerations are fired. Once the small passage formed by the obstacle and the wall is left behind, the robot began to accelerate. At point P_5 a wall is too close to the checkpoint, and a new deceleration is fired. The time used to complete the mission is similar to that achieved with the panning strategy, but execution is safer and the robot speed references are smoother.

5. Extended Experiments

The proposed strategy has been tested in longer and more realistic experimental setups. Figure 7 shows the Robotics Lab at the University of Oviedo and the point-to-point mission assigned. The lab area of interest is around 130 square meters, and the point-to-point total distance is 55.6 meters. Figure 8 shows several views of the checkpoints and the obstacles that surround them. Most of these obstacles are removable, in order to present a less cluttered challenge to test the performance degradation with the proposed algorithm. Six experiments are reported and their result compared. Experiments 1 to 3 were carried out removing the obstacles in the checkpoints vicinity, and experiments 4 to 6 were performed in the more cluttered setup in Figures 12 and 13.

Experiments differ from each other on the maximum robot velocities and accelerations applied, and in the length of the scan window d_f adopted (see Table II).

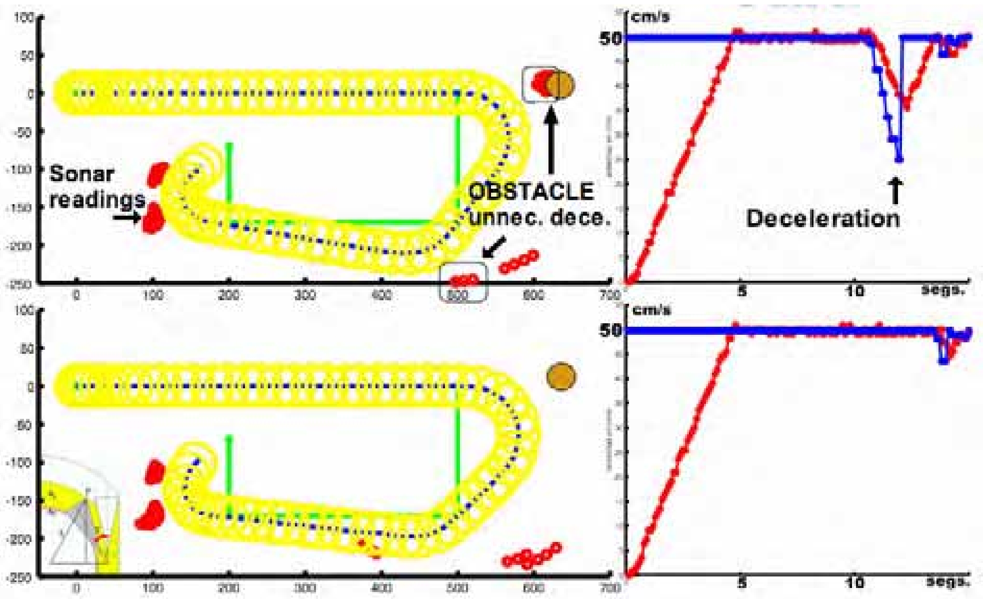


Figure 10. Avoiding situations of unnecessary deceleration by panning. (Above) The movable obstacle is located in a zone of unnecessary deceleration A1a, which produces a velocity reduction in t=12s. The same happens with the inferior wall. (Below) In the first turn the sensors fired are those located at 15 degrees, the obstacle in not detected and no deceleration is commanded. A better operation occurs in the vicinity of the wall too

Experiments	(vM, aM)	(ωM, αM)	df
1,4	(90,50)	(70,50)	170
2,5	(60,25)	(70,50)	170
3,6	(60,50)	(70,50)	144

Table 2. Extended experimental results conditions

Scan length is larger then the minimum, to allow smoother velocity reductions, but it increases the risk of unnecessary decelerations when moving in a straight line as discussed. Sweep aperture was always ρ=22.5 degrees implemented by firing the two frontal sonars. Table III summarizes the results of the experiments. The first line refers to a Stop-and-Turn control strategy that allows moving from point to point without local obstacle detection, supposing that there are no obstacles interfering in the straight-line paths.

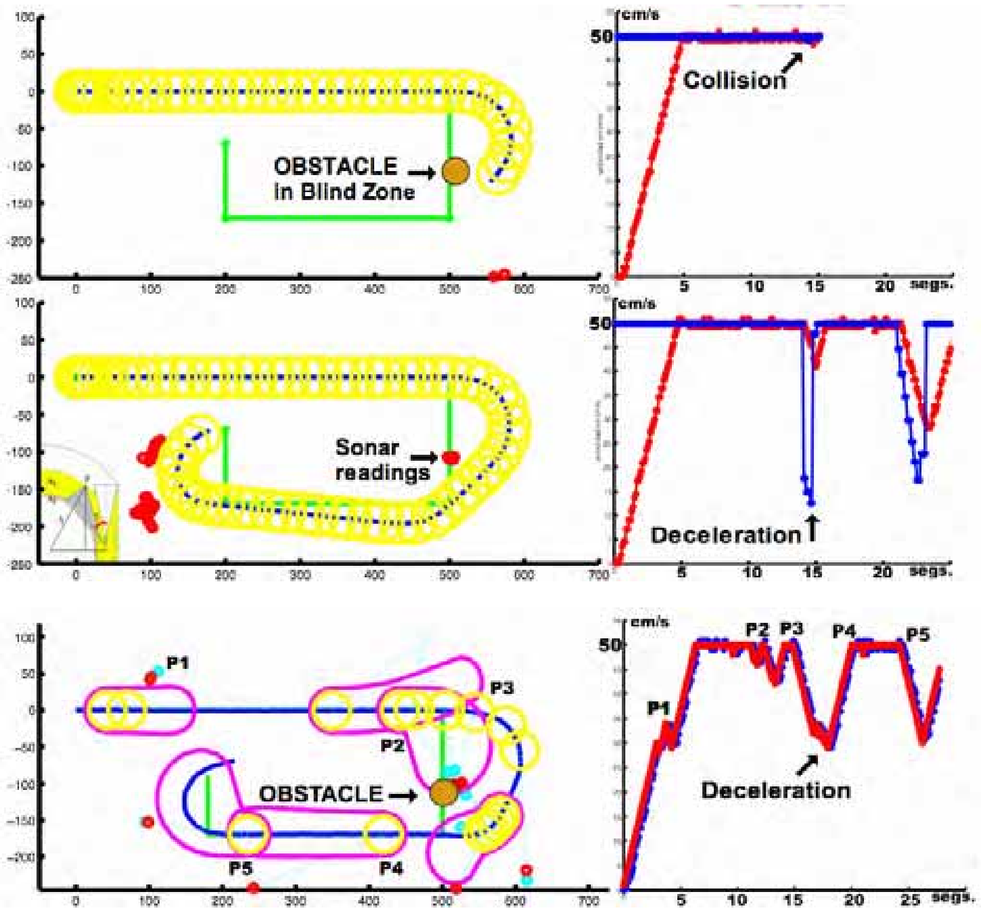


Figure 11. Avoiding blind zones by sensor selection. (Above) The movable obstacle is located in a blind zone A3, which produces a collision in t=15s. (Middle) By firing sensors at γ=15deg. the obstacle is detected and a lineal deceleration is commanded. It changes the robot's trajectory enough to avoid the obstacle, and the robot is able to finish the mission. (Below) Results obtained using the covering algorithm. We can see the real trajectory followed by the robot and some candidate stopping trajectories: straight, turning left and turning right. Notice that the algorithm generates references compatible with the robot dynamics

	dist. (cm)	time (s)	v_a (cm/s)
Stop & Turn	5560	144	38.6
1	6178	84	73.5
2	5899	116	50.6
3	6011	106	56.3
4	5784	89	64.1
5	5641	129	43.5
6	5756	111	51.6

Table 3. Results of the extended experiments

Average velocities v_a are lower in experiments 4-6 than in their corresponding 1-3 because the environment is more cluttered. Also final paths are shorter for the same reason (the robot has to negotiate more carefully at every checkpoint).

Fastest motion happens in Exp. 1 (v_a = 73.5) because the robot has higher velocities and acceleration capabilities to exploit in an easier environment. Experiment 3 has better performance than Experiment 2 only because the robot decelerates faster. Interestingly, being able to see 20% further than "robot 3" does not help "robot 2" to overcome such fact. The same reasoning applies to experiments 5 and 6.

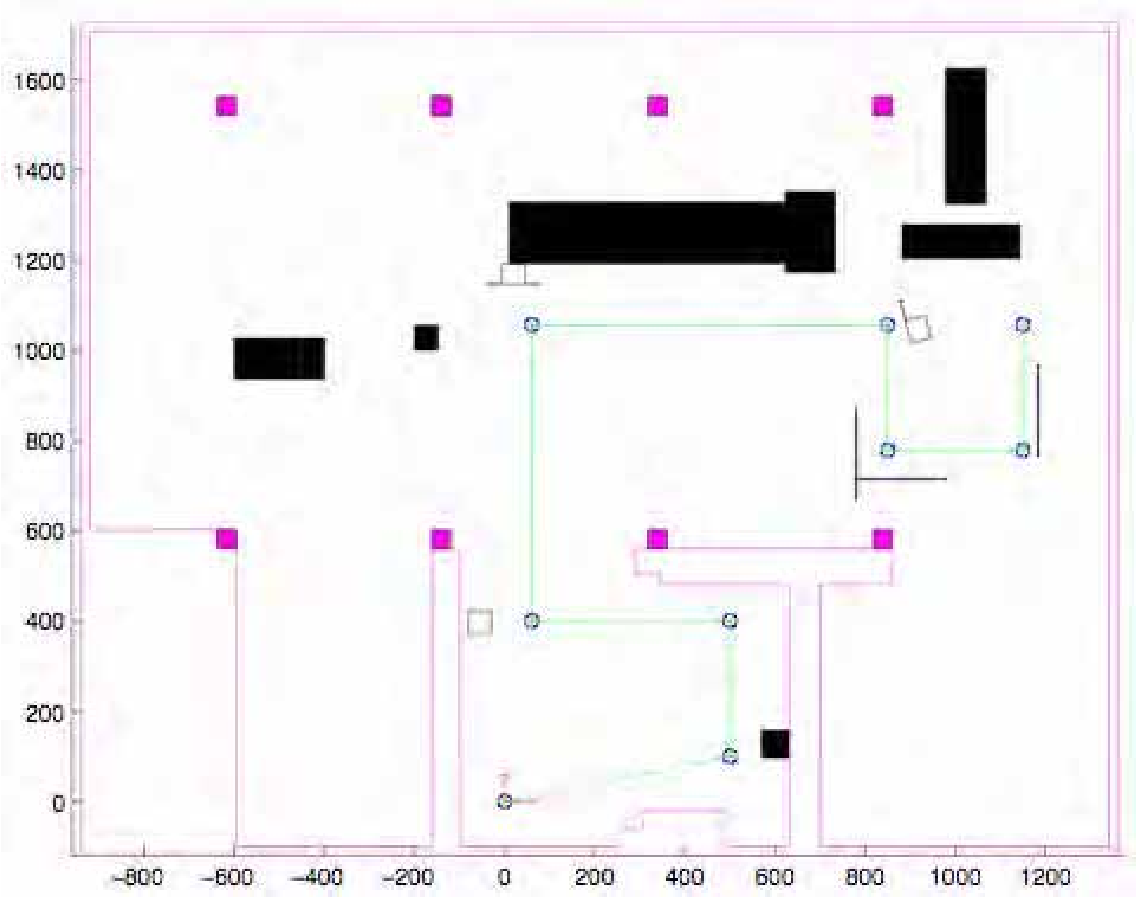
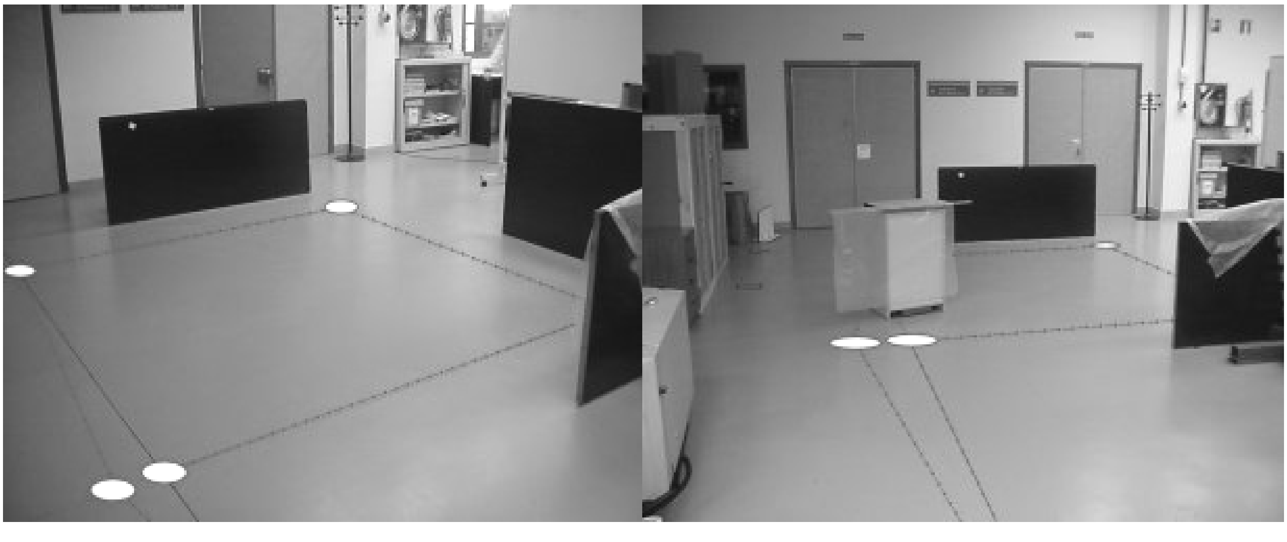


Figure 12. Lab workspace setup: an area of 130 square meters, and a point-to-point mission of a length of 55.6 meters with 12 checkpoints



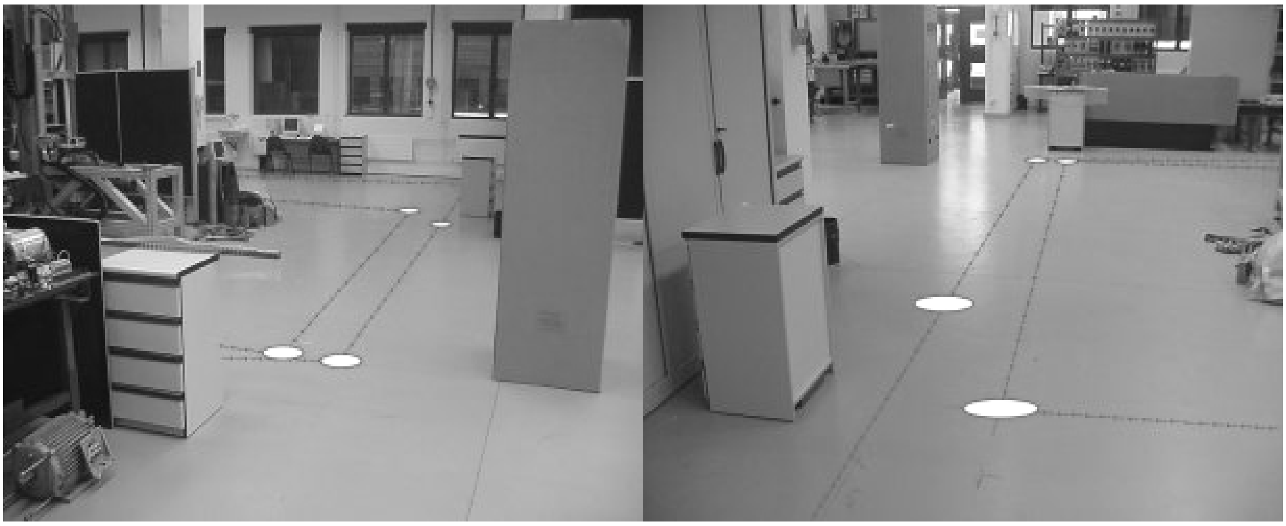


Figure 13. The experimental setup in Figure 12, and simulated in Figure 6. The images show details of the obstacles around each checkpoint

6. Conclusions

As the sensorial capacity of the robots increases, for example with the availability of MEMS sensors, it will be a requisite for good real-time performance to wisely select and manage them, in order to fulfill the strong time requirements of machines moving in human environments. This work represents a first approach towards solving this kind of problems. An analysis is presented which permits the selection of the sensor requirements for collision avoidance tasks of mobile robots. The design is compatible with motion in real time, as only the indispensable environment zones are explored, avoiding unnecessary velocity reductions. Perception system restrictions can be considered in the strategy. The analysis is deterministic, and the effect of sensor uncertainties is not discussed.

The sensor model presented is general enough to be implemented with various different sensing devices. In practice, some restrictions in the election of the parameters will depend on the specific sensor characteristics, e.g., with a sonar ring, transducers are arranged around the robot with fixed intervals γk, and only those pan angles are permitted. Using computer vision for range measurement gives us more flexibility, but restrictions will exist on the other two parameters. Notwithstanding, the ideas seem to be extensible to more complex models of robots or sensors, and to different motion control policies.

8. References

Alvarez, J. C., et al. (2002) Sensor management for local obstacle detection in mobile robots. New York, IEEE, pp. 2401-2406.

Alvarez, J. C., A. Shkel, and V. Lumelsky. "Accounting for Mobile Robot Dynamics in Sensor-Based Motion Planning." Int. Journal of Robotics and Automation 16, no. 3(2001): 132-142.

Brock, O., and O. Khatib. "Elastic strips: A framework for integrated planning and execution." Experimental Robotics VI 250(2000): 329-338.

Kelly, A., and a. Stentz. "Rough terrain autonomous mobility - Part 1: A theoretical analysis of requirements." Autonomous Robots 5, no. 2(1998): 129-161.

Laubach, S., and J. Burdick. "RoverBug: Long range navigation for Mars Rovers." Experimental Robotics Vi 250(2000): 339-348.

Lumelsky, V., and T. Skewis. "Incorporating Range Sensing in the Robot Navigation Function." IEEE Transactions on Systems Man and Cybernetics 20, no. 5(1990): 1058-1069.

Luo, R. C., C. C. Yih, and K. L. Su. "Multisensor fusion and integration: Approaches, applications, and future research directions." IEEE Sensors Journal 2, no. 2(2002): 107-119.

Ulrich, I., and J. Borenstein. "The GuideCane - Applying mobile robot technologies to assist the visually impaired." IEEE Transactions on Systems Man and Cybernetics Part a-Systems and Humans 31, no. 2(2001): 131-136.

Behaviour Based Mobile Robot Navigation with Dynamic Weighted Voting Technique

Shamsudin H.M. Amin, Rosbi Mamat & Tan Chee Kwong

1. Introduction

Designing a mobile robot that operates in our everyday environment is a challenging task. Complexity of the environment set strict requirements for both the hardware and software components of the robot. A robot needs sensors and sensor data processing to keep updating the environment. Furthermore, the robot has to integrate task execution with fast reaction to unexpected situations. These problems are fundamental to embodied autonomous systems that have to interact with unknown dynamic environments. To overcome this problem, various types of architectural framework of mobile robot have been introduced. These methods range from centralized sense-model-plan-act architectures to distributed behaviour based architectures.

Behaviour-based architecture has emerged as an alternative to traditional approaches for designing autonomous mobile robots (Maes, 1989). It consists of a collection of task-achieving modules or behaviours, which achieve goals and run independently. Each behaviour can take inputs from the robot sensors and send outputs to the robot actuators. Intelligence emerges from the interaction of the behaviours of the system. Thus the coordinator plays an important role to combine the outputs from several conflicting behaviours. This is known as the action selection problem (ASP) and following is the definition of ASP: "How can such an agent select 'the most appropriate' or 'the most relevant' next action to take at a particular moment, when facing a particular situation?" (Pirjanian, 1998).

A dynamic weighted voting technique is introduced to solve the problem in multiple behaviour coordination. It proposes a satisfactory mechanism for action selection that covers the three criteria, namely capability of dealing with multiple problems, multi-valued behaviour, and dynamic priority. The use of voting technique for command fusion allows the mobile robot to deal with multiple problems. It takes a shared control approach where each behaviour module concurrently shares control of the mobile robot by generating votes for every possible motor command.

A centre arbiter will then perform command fusion to choose the most appropriate action. Besides, the generated votes are between 0 and 1, with vote zero being the least desired action and vote one is the most desired action. It employs the concept of multi-value rather than simple binary value. The votes are generated in this manner to show the possibility for each action to achieve behaviour's goal. With the weight generation module, the behaviours' weights are generated based on the readings from various sensors. In different situations, the behaviours will have different weights. Therefore, the priority of each behaviour changes dynamically according to the situation by modifying their relative importance.

Various techniques have been proposed for behaviour coordination to solve the action selection problem. From the literature, the action selection mechanisms had been divided into two main groups, known as arbitration and command fusion respectively (Brooks, 1986). Behaviour arbitration is having the command of one behaviour completely overriding the commands of other behaviours. These include priority-based (Kosecka & Bajcsy, 1993), state-based (Arkin & Mackenjie, 1994), and winner-take-all (Pirjanian, 1998) approaches. Meanwhile command fusion mechanisms coordinate the activities of the set of behaviours that are active simultaneously by selecting the action that best satisfies the system's goal. These can further be divided into superposition (Khatib, 1986; Arkin, 1989; Saffiotti, et al., 1995), fuzzy (Tunstel, 1995; Pirjanian & Mataric, 1999) and voting (Brooks, 1986; Rosenblatt, 1995; Riekki & Roning, 1997) approaches.

In this paper, a dynamic weighted voting technique is proposed for mobile robot goal directed navigation in unknown dynamic indoor environment. Section 2 deals with the robot behaviour analysis. Section 3 discusses the implementation of this goal directed navigation. Section 4 describes the design of behaviour modules and weight manager. Section 5 presents the experimental results and discussion. Conclusion and suggestions for further work is given in section 6.

2. Robot Behaviour Analysis

The analysis of behaviour is an important aspect. The main goal of the mobile robot must be analyzed in detail. It is a top-down approach that involves decomposing the main objective into simpler ones, in such way that the main objective is achieved as a result from the execution of simpler behaviours and from their interaction. In short, a main objective like navigation can be decomposed into simple objectives like obstacle avoidance, and goal seeking. These simple objectives are going to be the basic behaviours for the mobile robot.

However, from an engineering point of view it is more appropriate to construct a system with a specified performance and functionality (Pirjanian, 1998). A top-down breakdown of the given task into a set of objectives identifies a set of relevant behaviours that when coordinated appropriately can achieve that given task.

Each objective module constructed from the value-tree is implemented as a single behaviour. Behaviour is actually a mapping from perception to action. However, it could be not only to define the mapping from perception to action but also to associate with each alterative action value that reflects its desirability. This is the key feature in voting technique, where behaviours vote for the desirability of the possible action set. The vote value is actually representing the preferences of each action in an interval of [0, 1].

Behaviour is treated as a specific objective function. Behaviours directly calculate the objective functions for particular objectives. The votes are generated in order to meet the behaviours' objectives. These votes generate assignments of preferences to action set. The assignments can be a fuzzy membership function, a probability distribution, or other pattern that suitable for the application. Each behaviour may use whichever representation and algorithm is most appropriate for that specific task. This representation does not exclude implementation using a look-up-table, a finite state machine, a neural network, or any other approach.

Meanwhile, the weight for each behaviour also needs to be defined. The weights will reflect the priority of the behaviours. Behaviour with higher priority will have bigger weight. The weight manager takes the sensor input and decides the weight based on the situation. The weight for each behaviour is chosen with the value to show their priority and normalised to 1. The exact value needs not to be accurate because only the relative values are important (Rosenblatt, 1997).

The calculation of objective functions and assignment of weights are heuristic processes. There is no general guideline on these design issues due to the constraint of the unpredictable dynamic environment that the robot needs to deal with. Because from an engineering point of view, the dynamic changes of the environment are uncontrollable factor. Therefore, the design is done empirically and experiments have to carry out to improve the quality. This is actually a key feature of the embodiment in behaviour-based approaches. The mobile robot needs to experience the world directly, thus the idea of designing a mobile robot can only be proved by having a real robot that deals with the real environment.

2.1 Performance Analysis

The performance of the robot should be evaluated in quantitative terms. However, it is quite complicated due to the non-determinism of the robot trajectory that navigates in unknown dynamic environments (Fabiani et al., 1998). For dynamic domains this is notoriously difficult as they themselves resist modelling and characterization (Arkin, 1995). Comparing two sets of behaviours, even within the same task, is complex and the domain-dependent nature of the solutions can cause these systems to be basically incommensurate – one may fail some times, one may fail at other times and comparison is difficult (Manikonda et al., 1995). This is due to the inherent limitation in the system that deals with unknown dynamic environment. Thus various evaluation parameters are used by different researchers in mobile robot experiments (Gat, 1995; Rosenblatt, 1997; Pirjanian, 1998; Fabiani et al., 1998). In the work presented here, two parameters are suggested to be taken into consideration. These are reliability index and time index.

Reliability index, I_R is the ability to complete the task. It represents the percentage of completed task on the population. The formula is,

$$I_R = \frac{S_{success}}{S_{total}} \tag{1}$$

$S_{success}$ number of success

S_{total} number of tests

Meanwhile, the time index I_T is the completed-task time. It is the average time necessary for the mobile robot to perform the task. It will only be measured when the robot reaches goal. The formula is,

$$I_T = \frac{\sum_n T}{N} \tag{2}$$

T time to complete the task

N total number of completed task

The performance is analysed and evaluated from the performance indices. If the performance is not satisfied, the designers should find the root cause and improve the design from the behaviour analysis stage.

3. Implementation: Goal-Directed Navigation

Research in the field of robotics is accompanied by experimental investigation of ideas, validation of theoretical results and so on. Simulations are not well suited for generating conclusive test data and results. The belief, among the opponents of simulation, is that it is only in the real world that certain phenomena manifest themselves and thus studies should be based on real-world experimentation. While simulations are quite useful for the proof-of-concept stage of research, when the feasibility of algorithms needs to be tested, they do not suffice as proof of algorithm functionality in the real world. If an algorithm fails in simulation it will certainly not work in the real world, but the opposite is not necessarily true. Robotics is an experimental science and must have realistic experiments as a central component for verifying hypotheses. In order to test the algorithm and avoid repeated work in simulation, a physical robot, AIBOT, was used for testing and debugging.

3.1 Mobile Robot Goal-Directed Navigation

The motivation behind the presented work was to build a mobile robot for indoor goal-directed navigation. The goal is to implement behaviour-based approach using dynamic weighted voting technique to achieve navigation as a result of a collection of interacting behaviour. The mobile robot must successfully navigate around obstacles, reach its goal and do so efficiently.

Goal-directed navigation problem is a classical problem in mobile robotics. In its original sense, the term navigation applies to the process of directing a ship to its destination. For a navigating mobile robot, the process can be seen as answering the following three question: 1) “Where am I?” 2) “Where are other places with respect to me?”, and 3) “How do I get to other places from here?” (Levitt & Lawton, 1990). Sometimes, it does not require knowledge of the starting position. Thus the most important question is “How do I reach the goal?” In short, the definition of navigation was taken as the process of determining and maintaining a course or trajectory to a goal location (Gallistel, 1990). However, to achieve a successful navigation, an appropriate navigation scheme is needed. A navigation scheme is needed to find the lowest cost path from the robot’s start state to the goal state. Cost can be defined to be distance travelled, energy expended, time exposed to danger and so on. In the past decade, various kinds of navigation schemes are introduced, ranging from simple reactive search navigation to complicated map-building navigation (Franz & Mallot, 2000). To perform an indoor goal-directed navigation, a simple navigation scheme with direction following is needed. In direction-following navigation, the mobile robot is required to follow a certain direction to find the goal. The goal itself needs not to be perceivable during approach. If direction following is coupled with distance information, then direction following becomes more efficient. Since the environment is unknown a priori, the mobile robot needs to navigate and store the information of the environment. A set of states of the environment is stored and a spatial map of the environment is built for further planning. Planning is required to avoid local minima problem and find optimal way while navigating in unknown environment.

3.2 The Mobile Robot - AIBOT

To test the proposed voting technique, a physical mobile robot, AIBOT (Autonomous Intelligent Mobile Robot) was used in the experiments. Two wheels mounted on a single axis are independently powered and controlled, thus providing both drive and steering for the mobile robot. Meanwhile, two additional passive wheels or castors are provided for support. It moves at an average speed of 0.3 m/s. AIBOT's computational hardware is located on the body. The processing is performed by a MIT Handy Board based on Motorola 68HC11 microprocessor that includes 32K of battery-backed static RAM, two L293D chips capable of driving four DC motors, both analog and digital inputs for a variety of sensors, and a 16x2 character LCD screen (Martin, 1999). An expansion board is added to the system, which provides additional inputs for sensors, digital outputs, and connector mount for Polaroid 6500 ultrasonic ranging system. The Handy Board runs Interactive C (IC in short), a multi-tasking version of the C programming language. It consists of a compiler and run-time machine language module. IC implements a subset of C including control structures, local and global variables, arrays, pointers, 16-bit and 32-bit integers, and 32-bit floating point numbers. The program is compiled to Motorola hex file and loaded into Handy Board's memory using 6811 downloader. AIBOT is equipped with various sensors. These are infrared sensors, sonar sensors, and odometer.

3.3 Sensor Modules

Three kinds of sensors are used in the navigation task. These are infrared sensors, sonar sensors, and odometer. Infrared sensors are a type of light sensors, which function in the infrared part of the frequency spectrum. However, they are preferable to visible light because it suffers a bit less from ambient interference since it can be easily modulated. They consist of an emitter and a receiver. In mobile robot's obstacle detection, infrared sensors are used as reflectance sensors. The emitter provides infrared signal and the signal will be reflected to the receiver if there are obstacles. Infrared sensors can only detect the presence of obstacle but not able to measure the distance of the obstacle. SUNX CX-22-PN infrared sensor is used in AIBOT. The sensing range is up to 0.8 meter. Seven infrared sensors are placed in front of AIBOT. They are directly connected to the digital input ports of Handy Board.

Another common sensor technique in robotics for proximity detection is time-of flight measurement (TOF). This is commonly achieved by using sonar (sound navigation and ranging). In sonar, the detection is based on the propagation of waves between the target and detector. The sensing is initiated by first creating a sonic ping at a specific frequency. These transitions are fed to the transducer at around 50 kHz. As this chirp falls well out of the range of human hearing, the ping is not audible. The chirp moves radially away from the transducer through the air at approximately 343.2 m/s, the speed of sound. When the chirp reaches an object, it is reflected. This reflected chirp then travels back towards the transducer, again at the speed of sound. As the reflected signal hits the transducer, a voltage is created which is fed to a stepped-gain amplifier. The Polaroid 6500 series ultrasonic ranging system is used in AIBOT. It is an economical sonar ranging module that is widely used in robotics researches. This module is able to measure distances from 6 inches to 35 feet. Three transducers are placed in front of AIBOT to sense the environment.

To move to a goal point, mobile robots need to know its relative position from the goal location. Dead reckoning (derived from "deduced reckoning" from sailing) is a simple mathematical procedure for determining the present location of a vehicle by advancing some previous position through known course and velocity information over a given length of time. The simplest form of dead reckoning is often termed as odometry. This implies that the vehicle displacement along the path of travel is directly derived from some on-board odometer. A common means of odometric measurement involves optical encoders directly coupled to wheel axles. In AIBOT, the odometer sensors are connected to digital input port of Handy Board. They provide the rotational count of the wheel and thus the vehicle velocity can be calculated.

3.4 Possible Action Set

If both drive wheels turn in tandem, the robot moves in a straight line. If one wheel turns faster than the other, the robot follows a curved path. In order to set the possible set as candidates, a discrete number of circular trajectories are chosen. Eight possible actions are as shown in Fig. 1 and named as "Hard Left", "Left", "Soft Left", "Forward", "Soft Right", "Right", "Hard Right", and "escape" respectively. "Escape" action is different from others. It is an action where one wheel move forward and the other wheel move backward. Thus the robot turns instantaneously on a point. It enables the robot to turn in a narrow space.

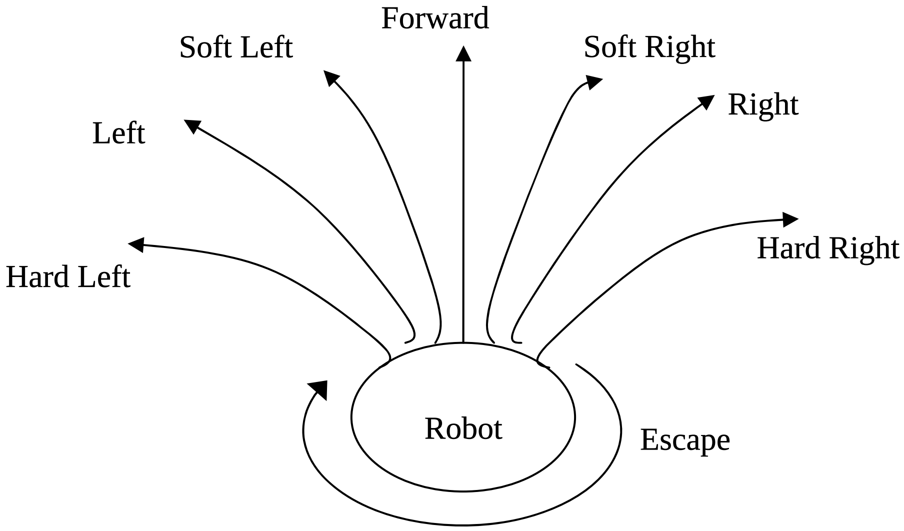


Figure 1. Possible actions for the robot

4. Design of Behaviour Modules and Weight Manager

In this stage, design of each behaviour modules is discussed. Each behaviour votes for every possible action base on the sensor reading. They vote in the pattern to achieve the behaviours' objective. Meanwhile, weight manager is designed to generate weight value for each behaviour.

4.1 Obstacle Avoidance Behaviour

The objective of this behaviour is to move the robot at a safe distance from the obstacles. Obstacle avoidance techniques range from primitive algorithms that detect an obstacle and stop the robot in order to avoid a collision, to sophisticated algorithms that enable the robot to detour obstacles. Sonar sensors and infrared sensors are used separately to meet the objective of avoiding obstacles.

Since infrared sensors are only detecting the presence of obstacles, the real distances of the obstacles are not available. Therefore, they only trigger upon detection of obstacles. For each possible path, three infrared sensors are responsible to determine the vote for that path. If all three infrared sensors detect the presence of obstacle, obstacle avoidance behaviour with infrared (b_{IR}) will vote 0 to that path and vote 1 if the opposite situation holds. If only one or two of the infrared sensors detect the obstacle, it will vote for a value in between 0 and 1. This can be shown by the equation below,

$$b_{IR}(x_m) = \begin{cases} 1.0 & \text{if no obstacle} \\ 0.7 & \text{if one sensor is triggered} \\ 0.3 & \text{if two sensors are triggered} \\ 0.0 & \text{if three sensors are triggered} \end{cases} \tag{3}$$

For “Escape” action, the voting process is different. The robot only needs to escape when it goes in to a space with a lot of obstacles, where the passage is too narrow and it is not allowed to move forward or turn in circular path. This condition is represented by the density of obstacles. When the number of infrared sensors that sense the presence of obstacle is increased, the obstacle density is high. Therefore, the vote is a function of number of sensors being triggered, as shown in the equation below,

$$b_{IR}(\text{Escape}) = \frac{\text{number of sensors being triggered}}{\text{total number of sensors}} \tag{4}$$

So, when the mobile robot goes into a narrow space, the escape action will receive higher average vote.

The obstacle avoidance behaviour with sonar (b_{sonar}) votes for the possibility to bring the mobile robot to a path free from obstacles,

$$b_{sonar}(x_m) = \begin{cases} 1 & \text{if } D_{obs} > D_{max} \\ \dfrac{(D_{obs} - D_{min})}{(D_{max} - D_{min})} & \text{if } D_{min} < D_{obs} \le D_{max} \\ 0 & \text{if } D_{obs} \le D_{min} \end{cases} \tag{5}$$

Dobs the detected obstacles distance

Dmax the maximum distance to detect

Dmin the minimum distance to detect

4.2 Goal-Seeking Behaviour

The task for goal-seeking behaviour is to look for a goal point and try to approach it. In a completely known environment, an optimal algorithm is used to search a state space to find the lowest cost path for the mobile robot. However, the work presented here focuses on the robot with no information about its environment before it begins its traverse. Thus the goal-seeking behaviour needs to direct the robot toward the goal point from an incomplete and uncertain model of the environment. “How to locate the goal-point” is a main issue in this behaviour. Two different techniques are used to achieve the target of this behaviour. These are goal-seeking behaviour with odometer and goal-seeking behaviour with planning.

This behaviour is designed to assign high values to actions that will cause the robot to face the target and low values to action that do the opposite. To use the odometer and dead-reckoning method, the coordinate system for the robot's world must be defined. The robot operates in a two-dimensional world. This means that at any given time its position can be defined in terms of 2-D Cartesian coordinates and the direction it is facing can be defined as an angle measured from one of the axes, as illustrated in Fig. 2. The orientation of the robot (theta) is measured counter clockwise from the x-axis. The position coordinates, x and y, are in meters and the orientation angle, theta, is in radians.

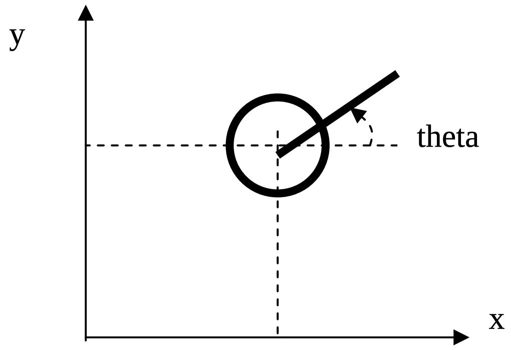


Figure 2. Coordinate System of the Mobile Robot

This behaviour generates the votes in three steps. First, the robot receives signals from the odometer. It counts the pulses from the odometer. These pulses are used to calculate the robot's position as well as its heading. The robot's current position and heading is calculated as below,

$$x(t) = x_o + \frac{d(V_R + V_L)}{2(V_R - V_L)}\left[\sin\left(\frac{t(V_R - V_L)}{d} + \theta_o\right) - \sin\theta_o\right] \tag{6}$$

$$y(t) = y_o + \frac{d(V_R + V_L)}{2(V_R - V_L)}\left[\cos\left(\frac{t(V_R - V_L)}{d} + \theta_o\right) - \cos\theta_o\right] \tag{7}$$

$$\theta(t) = \frac{t(V_R - V_L)}{d} + \theta_o \tag{8}$$

V_R	Velocity of right wheel
V_L	Velocity of left wheel
t	time
d	distance between wheels
θ_o	initial orientation of the robot

Secondly, with the robot's current position and heading, the relative position to the goal-point is calculated. The robot will then locate the target point direction. Finally, it votes for each possible action according to their angle to the goal point. A trajectory that will lead the robot toward the goal will get a higher vote and vice versa. This will give the robot more flexibility in navigation and take the uncertainty into consideration. The vote evaluation in this goal-seeking behaviour with odometer $(\mathrm{b_{odo}})$ is represented as an objective function of $\prod -$function (Pirjanian, 1998),

$$\mathrm{b_{odo}}(\mathrm{x_m}) = \frac{1}{1+\left(\frac{\theta_{\mathrm{m}} - \theta_{\mathrm{goa_1}}}{\beta}\right)^2} \tag{9}$$

θ_{m} angle of the candidate path

θ_{goal} relative angle to the goal

β width of window, which determines the value

Fig. 3 illustrates an example of the vote evaluation in this behaviour. In Fig. 3a, the mobile robot detects the goal on 60o to the right. It broadens the target direction and generates the votes as shown in Fig. 3b.

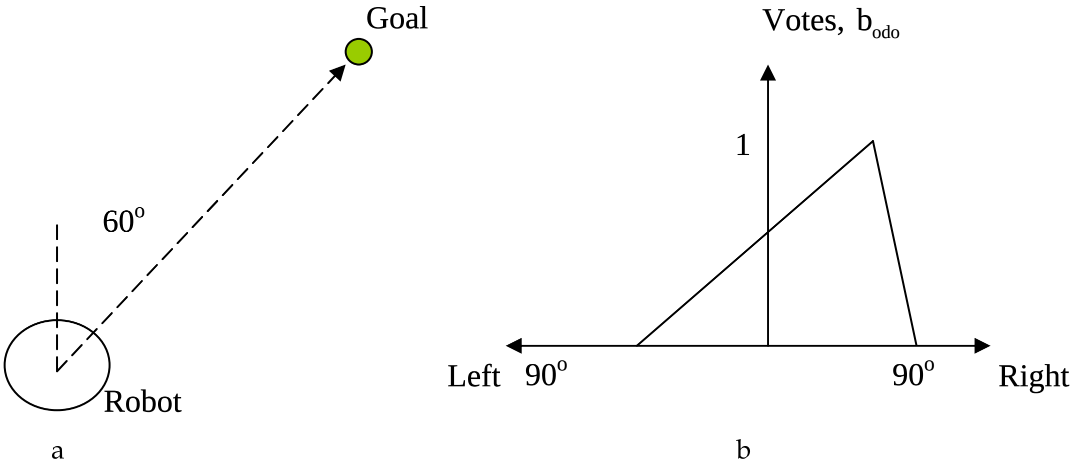


Figure 3. Vote evaluation in goal-seeking behaviour

In mobile robot navigation, the use of planning is a critical issue. While planning is useful for mobile robot to escape from the trap and avoid local minima problem, it is computationally expensive. This is mainly due to the use of knowledge representation. The knowledge could be a map, an analytical representation of surface features, or a semantic description of the world. Traditional systems build symbolic maps of the world for navigational reference. In the work presented here, knowledge is represented as a local spatial memory to minimize the use of memory. The addition of a local spatial memory allows the mobile robot to avoid areas that have been visited.

The concept of this local spatial memory is equivalent to leaving chemical trail in ants (Balch & Arkin, 1993). The memory is a two dimensional array of integers, which corresponds to the environment to be navigated. Each element of the grid records the number of times the corresponding square patch in the world has been visited. Every time the grid point is visited, it will be given a value of +1. The more often an area is visited, the larger the value. This recorded "trail" is used later for planning.

The planning function uses the memory with current positional information to generate the vote to run away from an area that has already been visited. It compares the mark point of the area around the robot's current position and finds the freest space, which is the least visited. The direction is recorded and set to be the temporary goal point. The planning function will vote for this temporary goal point in the same manner as the algorithm in goal-seeking behaviour with odometer. That is, it votes for each possible action according to their angle to the temporary goal point. The formula for goal-seeking behaviour with planning (b_{plan}) is,

$$b_{plan}(x_m) = \frac{1}{1 + \left(\frac{\theta_m - \theta_{goal}}{\beta} \right)^2} \tag{10}$$

θ_m angle of the candidate path

θ_{tem} relative angle to the temporary goal point

β width of window, which determines the value for half vote

4.3 Weight Manager

The weight manager plays an important role to change the behaviour's weight dynamically to enable the mobile robot to deal with the complexity of the environment. From the discussion above, it shows that there are two levels of arbitration in the mobile robot, that is the arbitration in behaviour team and the arbitration in centre arbiter. Therefore, the weight manager needs to generate three weight functions, one for arbitration in obstacle avoidance behaviour team, one for goal-seeking behaviour team, and one for centre arbiter. In obstacle avoidance behaviour team, there are two behaviours. Although the two behaviours use different sensors to achieve the same function, their priorities are the same in every situation of the environment. No matter what is the situation of the environment, both the behaviours will vote for the path to avoid obstacles. To represent their priorities in the system, these two behaviours have the same weight value. In other words, the weight values for these two behaviours will not change; they both are assigned with the value of 0.5. That is,

$$W_{IR} = E_{sonar} = 0.5 \tag{11}$$

where W_{IR} and W_{sonar} are the weight for obstacle avoidance with infrared and the weight for obstacle avoidance with sonar respectively.

In goal-seeking behaviour team, the two behaviours have different priorities in different situations. In example, when the mobile robot in a free state, the goal-seeking behaviour with planning will have lower priority than goal-seeking behaviour with odometer. On the other hand, when the mobile robot is trapped in a local minima, the goal-seeking behaviour will have higher priority such that its vote could bring the mobile robot out of the visited area. Therefore, the weight values are the function of the change in environment situation, while the environment situation is the degree of belief in a trapped area. It takes an assumption where the belief in trapped area is proportional to the sum of mark point value. If the mark point value is high, the belief in a trap area is high. Thus the weight value for goal-seeking behaviour with planning will increase, as shown below,

$$W_{plan} = \frac{\text{mark\_sum - mark\_min}}{\text{mark\_max - mark\_min}} \tag{12}$$

$$W_{odo} = 1.0 - W_{plan} \tag{13}$$

W_{plan} Weight value for goal-seeking behavior with planning

W_{odo} Weight value for goal-seeking behavior with odometer

mark_sum total sum for mark point

mark_min minimum sum value for mark point to trigger planning

In the centre arbiter, there are two weight values, which are weights for obstacle avoidance behaviour team and goal-seeking behaviour team respectively. The obstacle avoidance behaviour's weight must be larger to reflect that avoiding obstacles is more important than approaching the goal (Rosenblatt, 1997). Therefore, the weight value for obstacle team is set in an interval of [0.6, 0.9] while the goal team is [0.4, 0.1]. The weight values change according to the density of obstacles. An assumption is taken where the obstacle density is proportional to the number of infrared sensors being triggered. If the number of the infrared sensors being triggered is high, the obstacle density is high. It means that the mobile robot is in an obstructed situation with cluttered of obstacles. Thus the obstacle avoidance behaviour team should be given higher weight value relative to the condition when obstacle density is low. This is shown in the equation,

$$W_{obs} = 0.6 + \left(\left(\frac{N}{N_{total}} \right) * 0.3 \right) \tag{14}$$

$$W_{goal} = 1.0 - W_{obs} \tag{15}$$

W_{obs} Weight value for obstacle voidance behavior team

W_{goal} Weight value for goal-seeking behavior team

N Number of infrared sensors being triggered

N_{total} Total number of infrared sensors

5. Experimental Results and Discussion

Experiment is the way to prove the theoretical concept in real world. In the previous chapter, the implementation of dynamic weighted voting technique is discussed. The proposed technique is targeting on solving the problem in action selection for behaviour-based mobile robot. It was implemented on AIBOT for indoor goal-directed navigation. Experiments were carried out to validate the design of the proposed technique. The experimental results, comparisons and discussions are covered in this chapter.

5.1 Experimental Procedure

The proposed idea is evaluated and tested across a variety of different types of environment and behaviour combinations. The mobile robot went through the tests in 14 experimental fields. The objective of the experiment is to test the proposed dynamic weighted voting technique for behaviour-based mobile robot goal-directed navigation in an unknown dynamic environment. AIBOT was designed with four behaviours. The experiments were designed to show the results of the navigation with different behaviour combinations. Therefore, four experiments were carried out with the behaviour combination implemented shown as described below. In each experiment, AIBOT is tested in all of the 14 experimental fields. For each field, AIBOT was run for 50 times to get an average result. Thus in each experiments, 700 experimental runs were conducted. While running the experiment, the battery level for the motor is very important as the difference in battery level may affect the speed of AIBOT and thus affect the performance. To avoid this problem, the battery is fully charged before starting a new experiment set. Data is collected and the performance indices are reliability index and time index.

5.2 Navigation Results

The results of these navigation experiments are presented and discussed in this section.

Experiment 1 - Obstacle avoidance behaviour with infrared and goal-seeking behaviour with odometer

This was an experiment with only two behaviours, which are obstacle avoidance behaviour with infrared and goal-seeking behaviour with odometer. The objective was to test the concept of voting in solving action selection problem. The two behaviours with different objectives may generate conflicting action. Thus the dynamic weighted voting technique should solve the problem and choose the most appropriate action. The results are shown in Table 1.

	Reliability Index, I_R	Time Index, I_T
Field 1	1	10.31
Field 2	1	12.15
Filed 3	0.94	17.89
Field 4	0.88	14.75
Field 5	0.92	15.36
Field 6	1	14.32
Field 7	0.86	19.03
Field 8	0.64	26.25
Field 9	0.70	24.17
Field 10	0.66	27.12
Field 11	0.54	29.87
Field 12	0.30	34.71
Field 13	0.14	80.43
Field 14	0.16	84.59

Table 1. Results of Experiment 1

In field 1, the mobile robot showed reliable navigation to achieve the goal point with an average of 10.31s. Experiments in field 2 to field 5 indicated that the dynamic weighted voting technique was able to handle the conflict between the obstacle avoidance behaviour and goal-seeking behaviour. While avoiding obstacles in the test field, the mobile robot was able to maintain heading to the goal target. An example is illustrated in Fig. 4 for navigation in field 3. In the beginning, the obstacle avoidance behaviour will vote equally for each path because it detects no obstacle. Meanwhile, the goal-seeking behaviour will vote for the direction of the goal point. So, AIBOT will move to the direction of the goal point. When the obstacle is detected, the obstacle avoidance behaviour will vote for a free path. Although the goal-seeking behaviour will vote for the forward move, AIBOT still take a turn because of the greater weight of the obstacle avoidance behaviour. By the time AIBOT pass over the obstacle, all the paths are free and will get equal vote from obstacle avoidance behaviour. Therefore, it will take a right turn to go to the goal point as voted by the goal-seeking behaviour. However, the results show that the mobile robot may sometimes fail to achieve the goal point. It may collide with the obstacles during navigation. This is due to the obstacle angle that is not detected by the infrared, especially at the edges of obstacle.

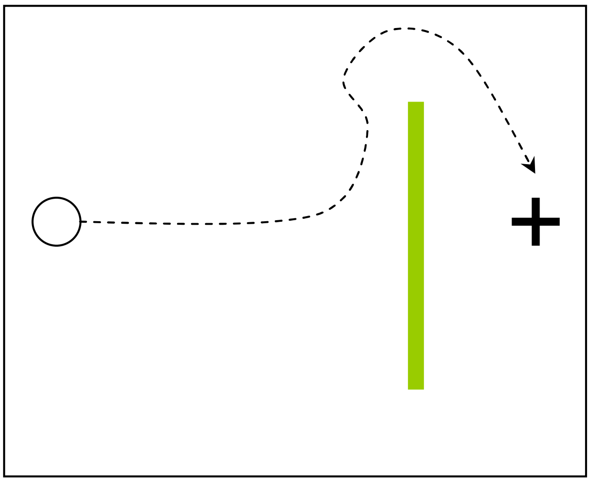


Figure 4. An Example for Navigation in Field 3

For navigation in corridor-like environment, the mobile robot shows high reliability in field 6. For corridor in field 7, there is some failure in the infrared sensors due to the obstacle angle too. In these two experiments, the time index showed an interesting issue where the mobile robot sometimes perform non-optimal path. Fig. 5 shows an example of this condition. Since the navigation is reactive, the mobile robot may in a position and sense that there are too many obstacles around. Thus the obstacle density increases. The obstacle avoidance behaviour then votes for “escape” action. As a result, the mobile robot turns around in the corridor or wobbles while the path is free.

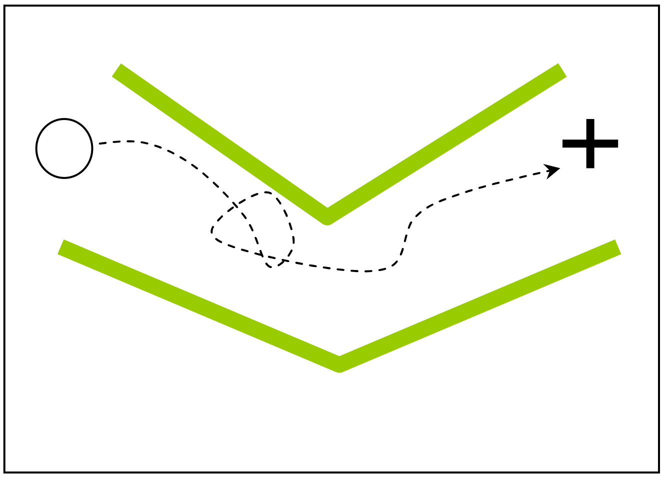


Figure 5. A condition of generating non-optimal path for corridor navigation in Field 7

The test in field 8 to field 11, the mobile robot produced low reliability index. Since the infrared sensors could only sense the presence of obstacles, it is not able to locate the distance of the obstacles. This has reduced the reliability for the mobile robot to find a narrow passage. Even if it can find the passage, it may waste some time to navigate and turn around. This is shown in Fig. 6 for field 9.

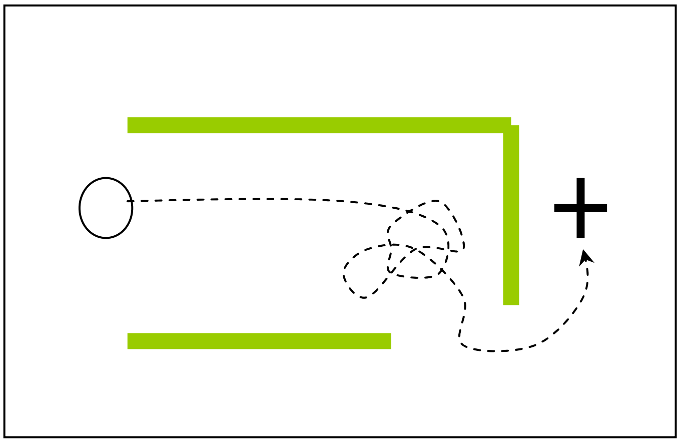


Figure 6. Mobile robot navigate turn around near the narrow passage

For the local minima problem, the mobile robot with only the reactive behaviours will navigate repeatedly in the same area. It can only come out from the trap by chance based on the interaction with the environment. Therefore, the reliability index is very low.

Experiment 2 - Obstacle avoidance behaviour with sonar and goal-seeking behaviour with odometer

This was an experiment of obstacle avoidance behaviour with sonar and goal-seeking behaviour with odometer. It was similar to experiment 1 in that to test the concept of voting in solving action selection problem. However, in this case the sonar sensor was used for obstacle avoidance behaviour. This experiment also provided a comparison regarding the difference in using infrared and sonar for obstacle avoidance. The results were shown in Table 2.

Navigation in field 1 indicated a high reliability to achieve the goal point with an average of 10.45s. The results for experiments in field 2 to field 5 showed a high reliability with only a few failures as experiment 1. With the dynamic weighted voting technique as the backbone, the mobile robot was able to achieve the goal point, to handle the conflict between the obstacle avoidance behaviour and goal-seeking behaviour. The failure, again, was due to the obstacle angle that was not detected by the sonar, especially at the edge of obstacles.

The result of navigation in corridor-like environment was similar to experiment 1. The mobile robot was confronted with the same problem of sensor failure.

For the test of narrow passage, as in field 8 to field 11, the mobile robot showed higher reliability relative to experiment 1. The mobile robot was more reliable in finding the narrow passage. It could find the way faster than it was in experiment 1 as indicated by the time index. This was because of the use of sonar sensors, which enabled the mobile robot to calculate the distance of obstacles and thus generate the vote based on the distances. Fig. 7 illustrates an example of smooth path through the narrow passage. However, there was sometimes some collision with obstacle due to the obstacle angle.

	Reliability Index, I_R	Time Index, I_T
Field 1	1	10.45
Field 2	1	11.97
Filed 3	0.86	15.53
Field 4	0.92	14.12
Field 5	0.92	16.26
Field 6	1	13.25
Field 7	0.78	16.46
Field 8	0.68	19.31
Field 9	0.74	19.67
Field 10	0.74	18.89
Field 11	0.66	20.12
Field 12	0.26	42.28
Field 13	0.20	76.31
Field 14	0.08	86.49

Table 6.2 Results of Experiment 2

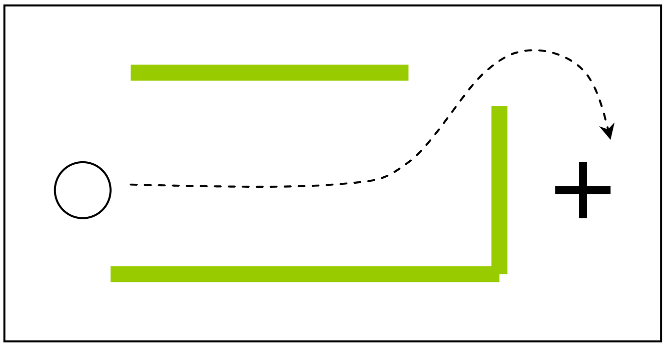


Figure 7. Smooth Navigation through the Narrow Passage

For the local minima problem, the results were similar as in experiment 1 because the mobile robot had only the reactive behaviours. Therefore, the mobile robot will navigate repeatedly in the same area. It can only come out of the trap by chance based on the interaction with the environment.

Experiment 3- Obstacle avoidance behaviour team (infrared and sonar) and goal-seeking behaviour with odometer

This was an experiment with three behaviours, namely obstacle avoidance behaviour with infrared, obstacle avoidance behaviour with sonar and goal-seeking behaviour with odometer. These experiments were designed to test the use of dynamic weighted voting technique for command fusion in homogeneous behaviour team. Besides, the performance of the mobile robot with the use of homogeneous behaviour team was studied. The results are shown in Table 3.

For navigation in field 1, the results were similar as in the previous experiments. AIBOT navigated with high reliability to achieve the goal point in average of 10.41s. In experiment for field 2 to field 5, the mobile robot achieved higher reliability compared to experiments 1 and 2. With the homogeneous behaviour team, both the obstacle avoidance behaviours with infrared and sonar generate voted to achieve the objective of avoiding obstacles.

	Reliability Index, I_R	Time Index, I_T
Field 1	1	10.45
Field 2	1	11.23
Filed 3	0.94	16.85
Field 4	1	13.94
Field 5	1	13. 76
Field 6	1	13.75
Field 7	0.94	14.52
Field 8	0.80	16.19
Field 9	0.78	17.16
Field 10	0.82	16.87
Field 11	0.72	18.43
Field 12	0.34	40.54
Field 13	0.24	81.22
Field 14	0.10	93.50

Table 3. Results of Experiment 3

The dynamic weighted voting technique was able to combine the votes from the behaviours.

In corridor navigation, the results in reliability index and time index showed that the performance was relatively better. The time for achieving goal point was reduced, due to the reduction in non-optimal paths. The navigation was smoother as shown in Fig. 8. This was due to the use of two types of sensors that helped to increase the reliability, as stated in the belief of "uncertainty handling with homogeneous behaviour".

For the test in field 8 to field 11, the mobile robot achieved a higher reliability compared to experiments 1 and 2. It appeared that performance was better with the use of homogeneous behaviour team. This proved that the command fusion in dynamic weighted voting technique was able to be an alternative to traditional sensor fusion. For the experiments in field 12 to field 14, the mobile robot achieved low reliability results. It was trapped in the local minima due to inability to do planning.

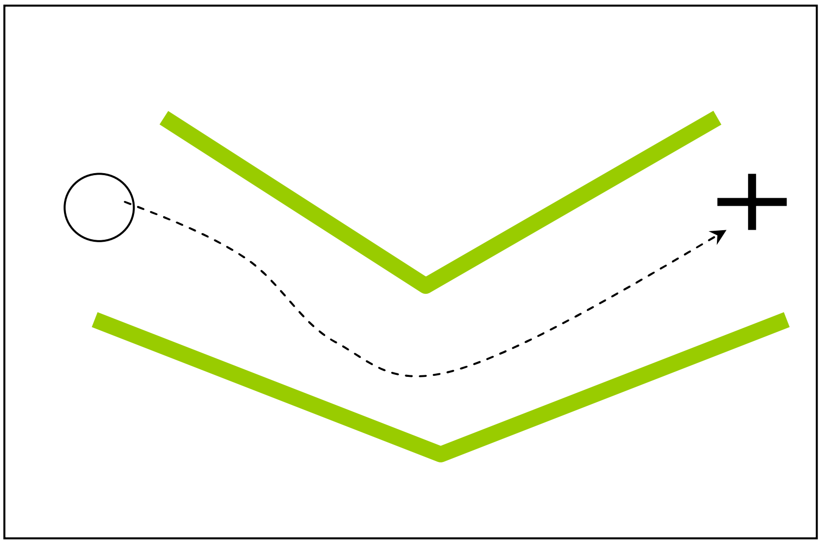


Figure 8. A Smoother Path in Corridor Navigation

Experiment 4 - Obstacle avoidance behaviour team (sonar and infrared) and goal-seeking behaviour team (odometer and planning)

This was an experiment with four behaviours. Two behaviour teams were built, which were obstacle avoidance behaviour team and goal-seeking behaviour team. The new added behaviour was goal-seeking behaviour with planning. The objective was to test the mobile robot with planning capabilities, to test the ability of the dynamic weighted voting technique to handle the use of planning behaviour, and to solve the problem in local minima. The results are shown in Table 4.

	Reliability Index, I_R	Time Index, I_T
Field 1	1	10.57
Field 2	1	12.03
Filed 3	0.90	17.48
Field 4	0.96	15.65
Field 5	1	14.11
Field 6	1	14.02
Field 7	0.94	15.03
Field 8	0.84	16.53
Field 9	0.82	16.77
Field 10	0.82	17.06
Field 11	0.76	19.14
Field 12	0.62	26.18
Field 13	0.54	65.32
Field 14	0.48	67.98

Table 6.4. Results of Experiment 4

For navigation in field 1 to field 11, the mobile robot was able to reach the goal point with the results similar to the previous experiments. This proved an easy way to add in new behaviours. With the design of reusable behaviour modules, a new behaviour could be easily added in without modifications to previous behaviours.

The main focus of this experiment was to test the quality of planning behaviour in field 12 to field 14. An example of navigation in field 13 is shown in Fig. 9. The mobile robot moved into the trap in the beginning because it had no knowledge about the environment. After some navigation in the same area, the mobile robot then marked the grid point and kept it as a local spatial memory. While the sum of mark point increased, the goal-seeking behaviour with planning will be given a higher weight than odometer. Meanwhile, this behaviour found a new goal to move the freest space at the bottom. It thus caused the mobile robot to go out of the local minima trap.

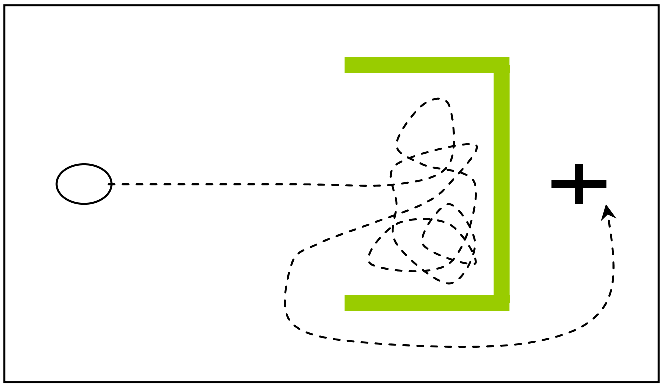


Figure 9. An Example of Navigation to Recover from Local Minima Problem

5.3 Discussions

The four experiments above showed the navigation of AIBOT using dynamic weighted voting technique. The robot navigated in various kinds of environments with various behaviour combinations. The experiments indicated that the proposed technique was able to solve the action selection problem and provide a successful behaviour design.

The results from the four experiments above were rearranged in Fig. 10 and Fig. 11 from the view of reliability index and time index respectively. The experimental data showed that the performance of AIBOT was relatively better in experiment 4, which was the experiment with four behaviours. This behaviour combination with an additional planning behaviour provided a solution for the problem of local minima. Meanwhile, it maintained the performance quality in other simple experimental fields. However, the performance in experiment 1 and 2 was sufficient for simple experimental field with only a few obstacles, such as field 1 to field 6. From the view of cost, these two behaviour combinations were easier to design. For behaviour combination in experiment 3, it indicated better performance compared to experiment 1 and 2 in the experiment for narrow passage, from field 7 to field 11. This is the advantage of using homogeneous behaviour team of obstacle avoidance.

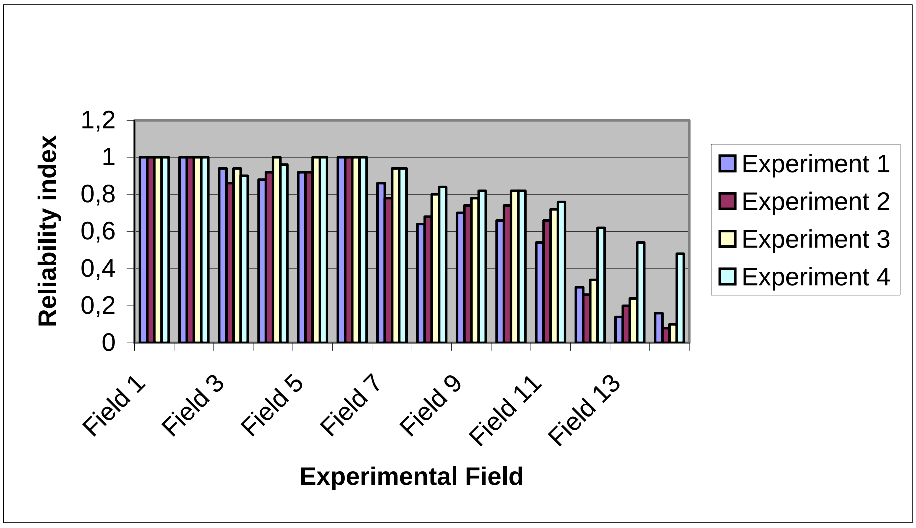


Figure 10. Reliability Indices for All Experiments

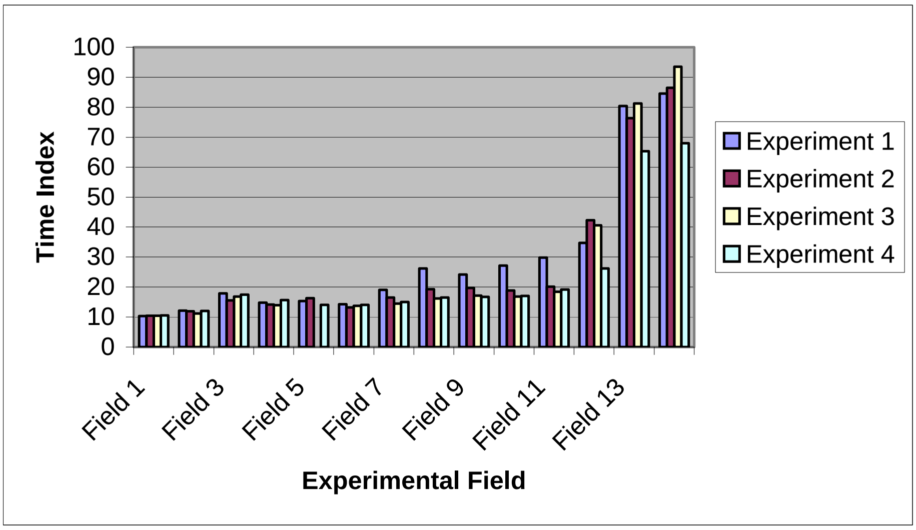


Figure 11. Time Indices for All Experiments

An obvious problem of the experiment above was the use of odometer as the only sensor for localisation. While odometer was easy to implement, it suffered from the problem of common dead reckoning error due to wheel slippage, unequal wheel diameter, and non-point wheel contact with the floor. This may have caused the error in calculation of the robot position. Therefore in some experiments, the mobile robot could not stop exactly on the position of the goal point.

5.4 Comparison with Other Techniques

A challenge in robotics is how to compare different architecture and architectural styles. Research papers on architectures are typically descriptive, describing what the system did and how it was organized but rarely provide metrics that would enable readers to determine what effect, if any, the architecture itself had on system performance. Thus comparison is done based on the theoretical analysis, rather than the runtime performance metrics due to the difference in mobile robot hardware, test field and others.

The dynamic weighted voting technique takes the advantages of previous action selection mechanisms while avoiding the shortcomings. The use of voting technique for command fusion allows the mobile robot to deal with multiple problems. Each behaviour module concurrently shares control of the mobile robot by generating votes for every possible motor command. This has overcome the shortcoming of behaviour arbitration technique that only deals with single problem at each point in time. Rather than choosing among behaviours, the dynamic voting technique allows the mobile robot to take different actions from all behaviours into consideration.

Meanwhile, the votes are generated between 0 and 1, with vote zero being the least desired action and vote one is the most desired action. This multi-value approach takes the idea of fuzzy logic with the belief that "the world is not black and white but only shades of gray". It enables the robot to deal with uncertainty in perception and incomplete information about the environment. The interval vote value shows the possibility for each action to achieve behaviour's goal rather than generate a single action such as behaviour arbitration and superposition command fusion. With the weight generation module, the behaviours' weights are generated based on the readings from various sensors. It modifies the relative importance of each behaviour by varying their weight value. In different environment situation, the behaviours will have different weights. Therefore, the priority of each behaviour changes dynamically according to the situation. This is the feature of winner-take-all behaviour arbitration, where the behaviours' priorities are changing dynamically. It will enable the mobile robot to deal with the complexity of the environment and avoid the discrimination against behaviours.

5.5 Advantages and Disadvantages Dynamic Voting Technique

The main advantage of the dynamic weighted voting technique is that it provides a reactive solution to deal with challenges in real-time responsiveness. The behaviour module directly maps the sensors reading to the vote value as an objective function. The computational process in sensor fusion is avoided. Meanwhile, planning modules is allowed to provide a better solution. With the dynamic weighted voting technique as the backbone, the performance of other behaviour modules will not be affected with the addition of planning modules. It employs a fully distributed architecture where all the behaviour modules run completely independently. The technique also enables the mobile robot to handle the challenges of uncertainties. The votes generated in an interval of [0, 1] in order to take uncertainties into consideration. Furthermore, the dynamically changing weights allow the mobile robot to deal with dynamic changes of the environment. In addition, the proposed design guidelines provide an engineering way to design the mobile robot from initial stage to performance analysis.

However, it too encounters with several problems. From the experiments, it is clear that the dynamic weighted voting technique is not allowed for task sequencing. This means that the mobile robot is not able to handle multiple tasks together and plan the sequence of these tasks in advance. In the other hand, although the design guideline provides a solution in behaviour design, the determination of the vote value is still primitive. A lot of experiments need to be carried out to find a better result.

5.6 Future Development

Current work on the dynamic weighted voting technique has some limitations because it is still on the development stage. There is still a room for improvement. Future development is suggested to focus on several directions, as discussed below.

1.) Learning capability
Currently, the objective function for the generation of vote value is determined empirically through experiments. The approach often taken is an iterative one of trial and error. The design of some behaviour may need several days of experimental debugging. Therefore, the process is time-consuming. Furthermore, once the vote value is fixed, it will not change. Further work may focus on studying the possible role of learning techniques to generate behaviours. Learning capability produces changes within a mobile robot to enable it to perform more effectively in its environment. It serves as an aid to fine-tune the vote value. With the learning capability, the mobile robot could learn during the navigation and tune the objective function to an optimum value.

2.) Additional level for task sequencing
The dynamic weighted voting technique does not support for task sequencing. An additional layer could be added into the architecture to enable task sequencing. This layer does not take control within the voting scheme. It plays the role of planning the sequences of tasks, such as going to the table, then getting the can and finally throwing it into rubbish bin. With the support of task sequencing, the mobile robot may perform more complicated tasks and serve as human assistance in our everyday environment.

3.) Scaling in the number of behaviour
The current dynamic weighted voting technique is implemented on mobile robot indoor navigation with four behaviours. However, the limit of the performance is still an unknown. An interesting extension will have to be in the direction of more complex behaviour. It is interesting to test the performance in complicated task with large composition of behaviours.

4.) Extension into multi agent mobile robots
Since the dynamic weighted voting technique shows a successful result in mobile robot navigation, the work is suggested to be extended into multi agent mobile robot. The work in multi agent mobile robot includes the decision making of

multiple mobile robots. This robot team must be able to coordinate their job together to perform a useful task. The problem is similar to the decision making of multiple conflicting behaviour modules in the work presented here. Thus extending the dynamic weighted voting technique into multi agent mobile robot is a possible work.

6. Conclusion

Experiments were carried out to prove the effectiveness of the proposed dynamic weighted voting technique. The experiments were carried on fourteen experimental fields with four different behaviour combinations. The results and comparison of the different experiments wee discussed. These results appear to show that the dynamic weighted voting technique was able to handle the problem in action selection. Meanwhile, the design also provided a way to design the mobile robot with dynamic weighted voting technique in an organised manner. Comparison with other behaviour-based approaches was briefly discussed, so as well the advantages and the disadvantages of the dynamic weighted voting technique.

7. References

Arkin, R.C. (1989). Towards the Unification of Navigational Planning and Reactive Control. AAAI Spring Symposium on Robot Navigation. 1-5.

Arkin, R.C. (1995). Just what is a Robot Architecture Anyway? Turing Equivalency versus Organizing Principles. Proceedings of the AAAI Spring Symposium: Lessons Learned from Implemented Software Architectures for Physical Agents. Stanford, CA. 21-24.

Arkin, R.C. and D. MacKenjie. 1994. Temporal Coordination of Perceptual Algorithms for Mobile Robot Navigation. IEEE Transaction on Robotics and Automation 10(3): 276-286.

Balch, T. and Arkin, R.C. (1993). Avoiding the Past: A Simple but Effective Strategy for Reactive Navigation. Proceedings of IEEE International Conference on Robotics and Automation. Atlanta, GA. 678-685.

Brooks, R.A. (1986). A Robust Layered Control System for a Mobile Robot. IEEE Journal of Robotics and Automation. 2(1). 14-23.

Fabiani, L., Regnier, S., and Monacelli, E. (1998). Performance Analysis Tool for Reactive Mobile Robots. Proceedings of the Third IFAC Symposium on Intelligent Autonomous Vehicles (IAV '98). Madrid, Paris. 25-30.

Franz, M.O. and Mallot, H.A. (2000). Biomimetic Robot Navigation. Robotics and Autonomous Systems. 30.133-153.

Gallistel, C.R. (1990). The Organization of Learning. MIT Press. Cambridge, MA.

Gat, E. (1995). Towards Principled Experimental Study of Autonomous Mobile Robots. Autonomous Robots. 2(3). 179-789.

Khatib, O. (1986). Real-time Obstacle Avoidance for Manipulators and Mobile Robots. The International Journal of Robotics Research. 5(1). 90-98.

Kosecka, J. & Bajcsy, R. (1993). Discrete Event Systems for Autonomous Mobile Agents. Proceedings of the Intelligent Robotic Systems '93. Zakopane. 21-31.

Levitt, T.S. and Lawton, D.T. (1990). Qualitative Navigation for Mobile Robot. Artificial Intelligence. 44. 305-360.

Maes, P. (1989). How to Do the Right Thing. Connection Science. 1(3). 291-323.

Manikonda, V., Hendler, J., and Krisnaprsad P.S. (1995). Formalizing Behavior-based Planning for Nonholonomic Robots. Proceedings of the International Joint Conference on Artificial Intelligence. Montreal, Canada. 563-570.

Martin, F.G. (1999). The Handy Board Technical Reference. Massachusetts Institute of Technology, United State of America.

Pirjanian, P. (1998). Multiple Objective Action Selection and Behavior Fusion Using Voting. Aalborg University, Denmark: PhD Thesis.

Pirjanian, P. and M. Mataric. 1999. Multiple Objective vs. Fuzzy Behavior Coordination. In Lecture Notes in Computer Science, Fuzzy Logic in Mobile Robot Navigation.

Riekki, J. and Roning, J. (1997). Reactive Task Execution by Combining Action Maps. Proceedings of the International Conference on Integrated Robots and Systems (IROS). Grenoble, France. 224-230.

Rosenblatt, J.K. (1995). DAMN: A Distributed Architecture for Mobile Navigation. AAAI Spring Symposium on Software Architectures for Physical Agents. Stanford CA. 167-178.

Rosenblatt, J.K. (1997). DAMN: A Distributed Architecture for Mobile Navigation. Carnegie Mellon University, United States: PhD Thesis.

Saffiotti , A., Konolige, K., and Ruspini, E.H. (1995). A Multivalued Logic Approach to Integrating Planning and Control. Artificial Intelligence. 76 (1-2). 481-526.

Tunstel, E. (1995). Coordination of Distributed Fuzzy Behaviors in Mobile Robot Control. Proceedings of the IEEE International Conference on Systems, Man and Cybernetics. Vancouver, BC Canada. 4009-4014.

Stochastic State Estimation for Simultaneous Localization and Map Building in Mobile Robotics

Juan Andrade-Cetto, Teresa A. Vidal-Calleja & Alberto Sanfeliu

1. The Problem of Coupled Stochastic

1.1 State Estimation

[1]The study of stochastic models for Simultaneous Localization and Map Building (SLAM) in mobile robotics has been an active research topic for over fifteen years. Within the Kalman filter (KF) approach to SLAM, seminal work (Smith and Cheeseman, 1986) suggested that as successive landmark observations take place, the correlation between the estimates of the location of such landmarks in a map grows continuously. This observation was later ratified (Dissanayake et al., 2001) with a proof showing that the estimated map converges monotonically to a relative map with zero uncertainty. They also showed how the absolute accuracy of the map reaches a lower bound defined only by the initial vehicle uncertainty, and proved it for a one-landmark vehicle with no process noise.

From an estimation theoretic point of view, we address these results as a consequence of partial observability. We show that error free reconstruction of the map state vector is not possible with typical measurement models, regardless of the vehicle model chosen, and show experimentally that the expected error in state estimation is proportional to the number of landmarks used. Error free reconstruction is only possible once full observability is guaranteed.

Explicit solutions to the continuous time SLAM problem for a one-dimensional vehicle called the *monobot* appear in (Gibbens et al., 2000, Kim, 2004). Both give closed form asymptotic values of the state error covariance **P**. Kim observed that, for the case when not all landmarks are observed at all times, the asymptotic value on the determinant of **P** reaches a constant value greater than zero. Gibbens *et al.* on the other hand, observed that the rate of convergence of **P** is proportional to the number of landmarks used, and that its asymptotic value is independent of the plant variance. In their solution to the 1-d Brownian motion case, the state error covariance is linked to the total number of landmarks in the form of the total Fisher information $\mathrm{I}_\mathrm{T} = \sum_1^n \left(1/\sigma_\mathrm{w}^2\right)$. The expression indicates the "*informational equivalence of the measurements and the innovations*" (Bar-Shalom et al., 2001), and was derived from a simple likelihood function, one that does not contain the fully correlated characteristics of the measurement model. This Chapter contains a

[1] This work was supported by the Spanish Ministry of Education and Science under projects DPI 2001-2223, TIC 2003-09291 and DPI 2004-5414.

more general expression for the total Fisher information in SLAM that shows explicitly the unobservable directions of the state space.

To speed up the performance of Kalman filterbased SLAM algorithm, some authors have proposed the use of covariance inflation methods for the decorrelation of the state error covariance (Guivant and Nieto, 2003), subject to suboptimality of the filter. Adding pseudo-noise covariance to the landmark states is equivalent to making the system controllable. However, full decorrelation of a partially observable system might lead to filter unstability (Julier, 2003). In this Chapter we also show how to diagonalize only part of the state error covariance to obtain a suboptimal filter that is both linear in time, and stable, at the same time.

In summary; in SLAM, the state space constructed by appending the robot pose and the landmark locations is fully correlated, a situation that hinders full observability. Moreover, the modelling of map states as static landmarks yields a partially controllable state vector. The identification of these problems, and the steps taken to palliate them, are covered in this Chapter.

The Chapter is structured as follows. In Section 2 we present the Simultaneous Localization and Map Building problem as a stochastic state estimation problem, and choose the Extended Kalman Filter algorithm as an alternative for solving it. The steady state of the filter will always depend on the initial noise parameters. The effect of partial observability is known as marginal stability (Todling, 1999), and is in general an undesirable feature in state estimation. In Section 3 we show that marginal filter stability is a consequence of having a coupled state estimation problem, by anylizing the poles of the state error dynamics. In Section 4 we derive an expression for the total Fisher information in SLAM. The analysis yields a closed form solution that shows, explicitly, the unobservable directions of the map state.

Marginal filter stability and the singularity of the Fisher information matrix are equivalently consequences of having partial observability. In Section 5 we develop expressions for the observable and controllable subspaces for a non-holonomic velocity controlled planar mobile platform. The result is that as the number of landmarks increases, the state components get closer to being reconstructible.

In Section 6 we show how partial observability in SLAM can be avoided by adding a fixed external sensor to the state model, or equivalently, by setting a fixed landmark in the environment to serve as global localization reference. Full observability yields the existence of a (not necessarily unique) steady state positive semi-definite solution for the error covariance matrix, guaranteeing a steady flow of the information about each state component, and preventing the uncertainty (error state covariance) from becoming unbounded (Bar-Shalom et al., 2001).

In Section 7 we show how as a consequence of having a partial controllability, filtering of the landmark state estimates is terminated after a small number of iterations, i.e., their corresponding Kalman gain terms tend to zero. In subsection 7.1 we show a situation in which the filter becomes unstable during covariance inflation. In subsection 7.2 we introduce a method for covariance decorrelation that preserves the stability of the filter. Furthermore, we show in subsection 7.3 another solution for a stable covariance inflation algorithm, consisting on first recovering full observability prior to decorrelating the entire state error covariance matrix. Conclusions are presented in Section 8.

2. Kalman Filter Based SLAM

2.1 System Model

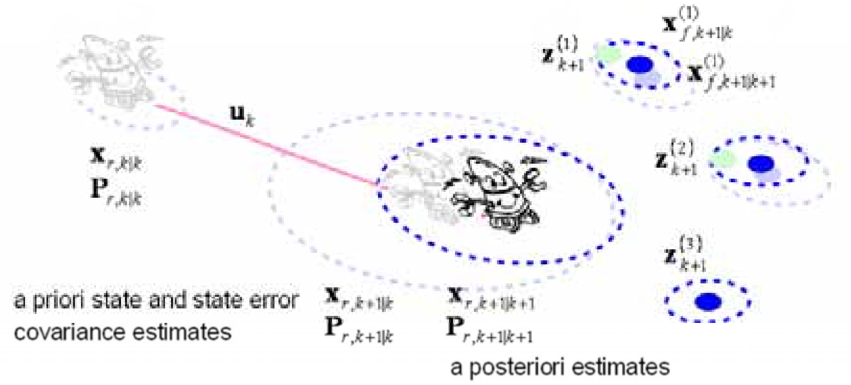


Figure 1. State estimation approach to simultaneous localization and map building

Formally speaking, the motion of the robot and the measurement of the map features are governed by the discrete-time state transition model

$$\begin{aligned} x_{k+1} &= f(x_k, u_k, v_k) \\ z_k &= h(x_k) + w_k \end{aligned} \tag{1}$$

The state vector x_k contains the pose of the robot $x_{r,k}$ at time step *k*, and a vector of stationary map features x_f, i.e.,

$$x_k = \begin{bmatrix} x_{r,k} \\ x_f \end{bmatrix} \tag{3}$$

The input vector $\mathbf{u}_k$ is the vehicle control command, and $\mathbf{v}_k$ and $\mathbf{w}_k$ are Gaussian random vectors with zero mean and psd covariance matrices **Q** and **R**, respectively, representing unmodeled robot dynamics and system noise the former; and measurement noise the latter. See Figure 1.

2.2 Algorithm

Provided the set of observations $Z^k = \{z_1, \ldots, z_k\}$ was available for the computation of the current map estimate $x_{k|k}$, the expression

$$x_{k+1|k} = f(x_{k|k}, u_k, 0) \tag{4}$$

gives an a priori noise-free estimate of the new locations of the robot and map features after the vehicle control command $\mathbf{u}_k$is input to the system. Similarly,

$$z_{k+1|k} = h(x_{k+1|k}) \tag{5}$$

constitutes a noise-free a priori estimate of sensor measurements.
Given that the landmarks are considered stationary, their a priori estimate is simply $x_{f,k+1|k} = x_{f,k|k}$; and the a priori estimate of the map state error covariance showing the increase in robot localization uncertainty is

$$P_{k+1|k} = E\left[\tilde{x}_{k+1|k} \tilde{x}^T_{k+1|k}\right] \tag{6}$$

$$= \mathrm{FP}_{k|k}\mathrm{F}^{\mathrm{T}} + \mathrm{GQG}^{\mathrm{T}} \tag{7}$$

The Jacobian matrices **F** and **G** contain the partial derivatives of **f** with respect to the state **x** and the noise **v**, evaluated at $\left(x_{k|k}, u_k, 0\right)$.

Assuming that a new set of landmark observations $\mathbf{z}_{k+1}$ coming from sensor data has been correctly matched to their map counterparts, one can compute the error between the measurements and the estimates with $\tilde{\mathbf{z}}_{k+1|k} = \mathbf{z}_{k+1} - \mathbf{z}_{k+1|k}$. This error, known as the innovation, aids in revising both the map and robot locations. The a posteriori state estimate is

$$\mathbf{x}_{k+1|k+1} = \mathbf{x}_{k+1|k} + \mathbf{K}\tilde{\mathbf{z}}_{k+1|k} \tag{8}$$

and the Kalman gain is computed with

$$\mathbf{K} = \mathbf{P}_{k+1|k}\mathbf{H}^{\mathrm{T}}\mathbf{S}^{-1} \tag{9}$$

where **S** is the measurement innovation matrix,

$$\mathbf{S} = \mathbf{H}\mathbf{P}_{k+1|k}\mathbf{H}^{\mathrm{T}} + \mathbf{R} \tag{10}$$

and **H** contains the partial derivatives of the measurement model **h** with respect to the state **x** evaluated at $(\mathbf{x}_{k+1|k})$.

Finally, the a posteriori estimate of the map state error covariance must also be revised once a measurement has taken place. It is revised with the Joseph form to guarantee positive semi-definiteness.

$$\mathbf{P}_{k+1|k+1} = (\mathbf{I} - \mathbf{KH})\mathbf{P}_{k+1|k}(\mathbf{I} - \mathbf{KH})^{\mathrm{T}} + \mathbf{KRK}^{\mathrm{T}} \tag{11}$$

3. Convergence

Substituting the linearized version of (4) in (8), we may rewrite the KF in the one-step ahead prediction form

$$\mathbf{x}_{k+1|k} = (\mathbf{F} - \mathbf{KH})\mathbf{x}_{k|k-1} + \mathbf{K}\mathbf{z}_k \tag{12}$$

and with the appropriate substitutions, using (6) and (12), the corresponding prediction error dynamics becomes

$$\tilde{\mathbf{x}}_{k+1|k} = (\mathbf{F} - \mathbf{KH})\tilde{\mathbf{x}}_{k|k-1} + \mathbf{G}\mathbf{v}_k - \mathbf{K}\mathbf{w}_k \tag{13}$$

In general, only for a stable matrix **F-KH**, the estimation error will converge to a zero mean steady state value. However, in SLAM, **F-KH** is marginally stable, thus the steady state error estimate is bounded to a constant value, subject to the filter initial conditions. To show **F-KH** marginally stable, consider the one landmark monobot from Figure 2. **F=I**, $\mathbf{G} = [1,0]^{\mathrm{T}}$, and $\mathbf{H} = [-1,1]$. For any value of

$$\mathbf{P} = \begin{bmatrix} \sigma_r^2 & \rho\sigma_r\sigma_f \\ \rho\sigma_r\sigma_f & \sigma_f^2 \end{bmatrix} \tag{14}$$

the Kalman gain, computed with (9), is

$$\mathrm{K} = \frac{1}{\mathrm{s}}\begin{bmatrix} -\sigma_{\mathrm{v}}^2 \\ \sigma_{\mathrm{w}}^2 \end{bmatrix} \tag{15}$$

where

$$\mathrm{s} = \sigma_{\mathrm{r}}^2 + \sigma_{\mathrm{f}}^2 - 2\rho\sigma_{\mathrm{r}}\sigma_{\mathrm{f}} + \sigma_{\mathrm{w}}^2 \tag{16}$$

is the innovation variance. Consequently,

$$\mathrm{F}-\mathrm{KH}=\frac{1}{\mathrm{s}}\begin{bmatrix}-\sigma_{\mathrm{v}}^{2}+\mathrm{s} & \sigma_{\mathrm{v}}^{2}\\ \sigma_{\mathrm{w}}^{2} & -\sigma_{\mathrm{w}}^{2}+\mathrm{s}\end{bmatrix} \tag{17}$$

with eigenvalues

$$\left\{\begin{matrix}1\\ \frac{1}{\mathrm{s}}\left(\mathrm{s}-\sigma_{\mathrm{v}}^{2}-\sigma_{\mathrm{w}}^{2}\right)\end{matrix}\right\} \quad \text{and} \quad \mathrm{s}\neq 0$$

One of the eigenvalues being on the unitary circle yields marginal stability, i.e., constant bounded non-zero mean error state estimate convergence. Moreover, the marginal stability of **F-KH** guarantees at least one *psd* steady state solution to the Riccati equation for the one-step ahead state error covariance (Vidal-Calleja et al., 2004b, Kailath et al., 2000).

$$\mathbf{P}_{k+1|k}=(\mathbf{F}-\mathbf{KH})\mathbf{P}_{k|k-1}(\mathbf{F}-\mathbf{KH})^{\mathrm{T}}+\mathbf{GQG}^{\mathrm{T}}+\mathbf{KHRH}^{\mathrm{T}}\mathbf{K}^{\mathrm{T}} \tag{18}$$

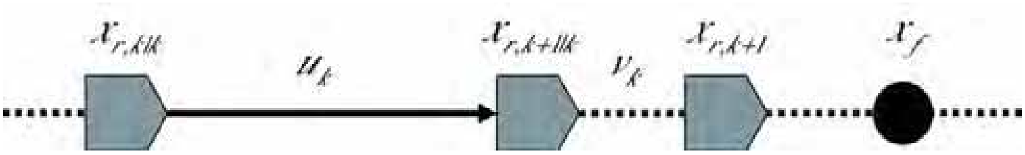


Figure 2. Monobot, a one-dimensional mobile robot

4. Total Fisher Information

Under the Gaussian assumption for the vehicle and sensor noises, and for linear vehicle and measurement models, the Kalman filter is the optimal minimum mean square error estimator. And, as pointed out in (Bar-Shalom et al., 2001), minimizing the least squares criteria $E[\tilde{\mathbf{x}}_{\mathbf{k+1|k+1}}\tilde{\mathbf{x}}_{\mathbf{k+1|k+1}}^{\mathrm{T}}]$ is equivalent to the maximization of a likelihood function $\wedge(\mathrm{x})$ given the set of observations Z^{k}; that is, the maximization of the joint probability density function of the entire history of observations

$$\wedge(\mathrm{x})=\prod_{\mathrm{i}=1}^{\mathrm{k}}\mathrm{p}(\mathrm{z}_{\mathrm{i}}|\mathrm{Z}^{\mathrm{i}-1}) \tag{19}$$

where **x** is the augmented map state (vehicle and landmark estimates), and $\mathbf{z}_i$ the entire observation vector at time *i*.

Given that the above pdfs are Gaussian, and that $E[\mathbf{z}_i]=\mathbf{Hx}_{i|i-1}$, the pdf for each measurement in SLAM is

$$\mathrm{p}(\mathrm{z}_{\mathrm{i}}|\mathrm{Z}^{\mathrm{i}-1})=\mathrm{N}(\tilde{\mathrm{z}}_{\mathrm{i|i-1}};0,\mathrm{S}_{\mathrm{i}}) \tag{20}$$

with $\mathbf{S}_i=E[\tilde{\mathbf{z}}_{\mathbf{i|i-1}}\tilde{\mathbf{z}}_{\mathbf{i|i-1}}^{\mathrm{T}}]$.

In practice however, it is more convenient to consider the log likelihood function $\ln\wedge(\mathrm{x})$. The maximum of $\ln\wedge(\mathrm{x})$ is at the value of the state **x** that most likely gave rise to the observed data Z^{k}, and is obtained by setting its derivative with respect to **x** equal to zero, which gives

$$\nabla_{\mathrm{x}}\ln\wedge(\mathrm{x})=\sum_{\mathrm{i}=1}^{\mathrm{k}}\mathrm{H}^{\mathrm{T}}\mathrm{S}_{\mathrm{i}}^{-1}\tilde{\mathrm{z}}_{\mathrm{i|i-1}} \tag{21}$$

An intuitive interpretation of the maximum of the log-likelihood is that the best estimate for the state **x**, in the least squares sense, is the one that makes the sum of the entire set of Mahalanobis distances $\sum_{i=1}^{k}\tilde{z}_{i|i-1}^{T}S_{i}^{-1}\tilde{z}_{i|i-1}$ as small as possible. A measure that is consistent with the spatial compatibility test described in (Neira and Tardós, 2001). The Fisher information matrix, a quantification of the maximum existing information in the observations about the state **x**, is defined in (Bar-Shalom et al., 2001) as the expectation on the dyad of the gradient of $\ln\wedge(x)$, that is

$$J = E\left[(\nabla_x \ln\Lambda(x))(\nabla_x \ln\Lambda(x))^T\right] \qquad (22)$$

Taking the expectation on the innovation error in the above formula gives the sum

$$J = \sum_{i=1}^{k} H^T(HPH^T + R)^{-1}H \qquad (23)$$

It is easy to verify that in the linear case, this expression for the total Fisher information is only a function of $P_{r,0|0}$, **Q**, and **R**. If, on the other hand, the EKF is used, the Jacobian **H** in (23) should be evaluated at the true value of the states $x_0.,\ldots x_k$ Since these are not available, an approximation is obtained at the estimates $\mathbf{x}_{i|i-1}$. The pre and post multiplying **H** is, in this context, also known as the *sensitivity matrix*.

A necessary condition for the estimator (the Kalman filter) to be consistent in the mean square sense is that there must be an increasing amount of information about the state **x** in the measurements. That is, as $k \to \infty$, the Fisher information must also tend to infinity. Figure 3 shows this for the monobot with constant parameters $P_{r,0|0}$=Q=R=1, and various sizes for the observation vector.

Notice how, as the total number of landmarks grows, the total Fisher information also grows, directly relating the number of landmarks to the amount of information available for state estimation in SLAM. Solving for the *k*-th sum term in **J** for the monobot,

$$J_k = \begin{bmatrix} \sum\sum\varsigma_{ij} & -\varsigma \\ -\varsigma^T & S_k^{-1} \end{bmatrix} \qquad (24)$$

with ς_{ij} the *ij*-th entry in S_k^{-1}, and $\varsigma = \left[\varsigma_{1i},\ldots,\sum\varsigma_{ni}\right]$

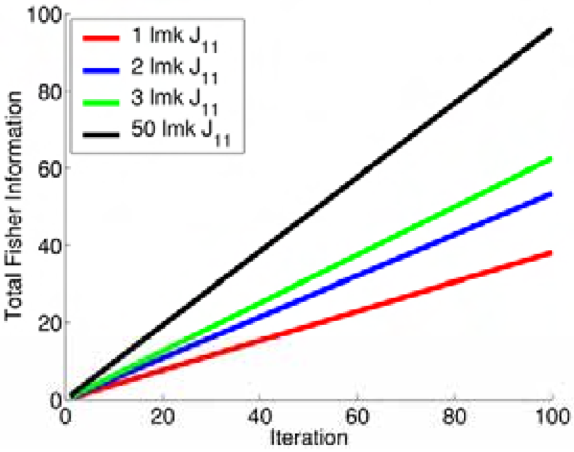


Figure 3. First entry in the total Fisher information matrix $\left(\sum\sum\varsigma_{ij}\right)$ for a monobot with variance parameters $P_{r,0|0}$=Q=R=1, and various sizes for the measurement vector

Citing Bar-Shalom (Bar-Shalom et al., 2001): "*a singular Fisher information matrix means total uncertainty in a subspace of the state space, that is, the information is insufficient for the estimation problem at hand.*" Unfortunately, it can be easily shown, at least for the monobot case, that the first row (or column) of **J** is equivalent to the sum of the rest of the rows (or columns), producing a singular total Fisher information matrix. Thus, SLAM is unobservable.

This is a consequence of the form of the Jacobian **H**, i.e, of the full correlation in SLAM. Zero eigenvalues of $\mathbf{H}^{\mathrm{T}}\mathbf{S}^{-1}\mathbf{H}$ are an indicator of partial observability, and the corresponding vectors give the unobservable directions in state space.

So for example, for a one-landmark monobot, the innovation variance is the scalar (16), and since **H**=[-1,1], the Fisher information matrix in (23) evaluates to

$$\mathrm{J} = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \sum_{i=1}^{k} \frac{1}{s_i} \tag{25}$$

The unobservable direction of the state space is the eigenvector associated to the null eigenvalue of **J**, we denote it $\mathrm{E}_{\mathrm{Ker}\mathcal{O}}$ since it represents a basis for null space of the observability matrix $\mathcal{O}$, and evaluates to

$$\mathrm{E}_{\mathrm{Ker}\mathcal{O}} = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \tag{26}$$

5. Relationship Between Observable and Controllable Subspaces for a Velocity Controlled Non-linear Planar Robot

We have developed closed form expressions for the bases of the observable and controllable subspaces in SLAM for a monobot and for a simple position controlled planar vehicle (Andrade-Cetto and Sanfeliu,).

In this Section we derive the observable and controllable subspace bases for the planar robot shown in Figure 4, a nonlinear nonholonomic velocity controlled wheeled vehicle with three degrees of freedom, and an environment consisting of two-dimensional point landmarks located on the floor.

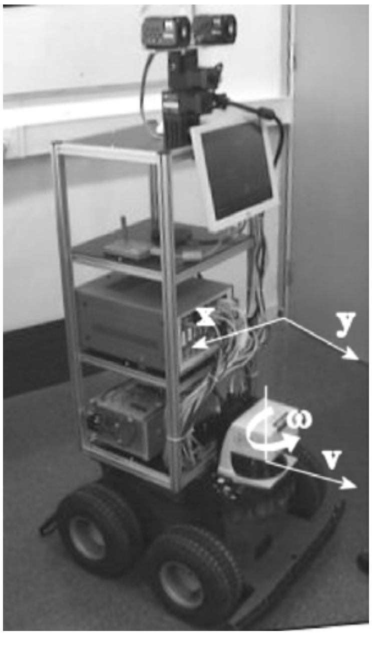


Figure 4. Two-dimensional mobile robot

The vehicle is controlled by a linear velocity v and a steering velocity ω. The process model used to predict the trajectory of the center of projection of the laser range scanner is given by

$$\begin{bmatrix} \mathrm{x}_{\mathrm{k}+1} \\ \mathrm{y}_{\mathrm{k}+1} \\ \theta_{\mathrm{k}+1} \end{bmatrix} = \begin{bmatrix} \mathrm{x}_{\mathrm{k}} + \tau\left(\left(\mathrm{v}_{\mathrm{k}} + \mathrm{v}_{\mathrm{v,k}}\right)\cos\theta_{\mathrm{k}} - \left(\omega_{\mathrm{k}} + \mathrm{v}_{\omega.\mathrm{k}}\right)\mathrm{l}\sin\theta\right) \\ \mathrm{y}_{\mathrm{k}} + \tau\left(\left(\mathrm{v}_{\mathrm{k}} + \mathrm{v}_{\mathrm{v,k}}\right)\sin\theta_{\mathrm{k}} + \left(\omega_{\mathrm{k}} + \mathrm{v}_{\omega.\mathrm{k}}\right)\mathrm{l}\cos\theta\right) \\ \theta_{\mathrm{k}} + \tau(\omega + \mathrm{v}_{\omega,\mathrm{k}}) \end{bmatrix} \tag{27}$$

where l is the distance from the center of the wheel axle to the center of projection of the laser range scanner, τ is the time constant, and $\mathrm{v}_{\mathrm{v}}, \mathrm{v}_{\omega}$ are zero mean Gaussian model noises.

The observation model is

$$\begin{bmatrix} z^i_{r,k} \\ z^i_{\beta,k} \end{bmatrix} = \begin{bmatrix} \sqrt{\left(x^i_f - x_k\right)^2 + \left(y^i_f - y_k\right)^2} + w_{r,k} \\ \tan^{-1}\left(\dfrac{\left(y^i_f - y_k\right)}{\left(x^i_f - x_k\right)}\right) - \theta_k + \dfrac{\pi}{2} + w_{\beta,k} \end{bmatrix} \tag{28}$$

with z^i_r and σ the distance and bearing of an observed point landmark with respect to the laser center of projection. x^i_f and y^i_f are the absolute coordinates of such landmark, and i is used for the labelling of landmarks. i=0 indicates an anchor feature not under estimation in order to guarantee full observability. w_r and w_β are zero mean Gaussian measurement noises.

The Jacobian matrices **F**, **G**, and **H** are obtained by differentiating Equations (27) and (28) with respect to states and noises. That is,

$$\mathrm{F} = \begin{bmatrix} 1 & 0 & -\tau((\mathrm{v}_{\mathrm{k}} + \mathrm{v}_{\mathrm{v,k}})\sin\theta_{\mathrm{k}} - \mathrm{l}\cos\theta(\omega_{\mathrm{k}} + \mathrm{v}_{\omega,\mathrm{k}})) & & 0 & 0 & \\ 0 & 1 & \tau((\mathrm{v}_{\mathrm{k}} + \mathrm{v}_{\mathrm{v,k}})\cos\theta_{\mathrm{k}} - \mathrm{l}\sin\theta(\omega_{\mathrm{k}} + \mathrm{v}_{\omega,\mathrm{k}})) & & 0 & 0 & \\ 0 & 0 & 1 & \ddots & 0 & 0 & \\ 0 & 0 & 0 & & 1 & 0 & \\ 0 & 0 & 0 & & 0 & 1 & \ddots \end{bmatrix} \tag{29}$$

$$\mathrm{G} = \begin{bmatrix} \tau\cos\theta_{\mathrm{k}} & -\tau\mathrm{l}\sin\theta_{\mathrm{k}} \\ \tau\sin\theta_{\mathrm{k}} & \tau\mathrm{l}\cos\theta_{\mathrm{k}} \\ 0 & 1 \\ \vdots & \\ 0 & 0 \\ 0 & 0 \\ \vdots & \end{bmatrix} \tag{30}$$

$$\mathrm{H}_{\mathrm{i}} = \begin{bmatrix} -\dfrac{\mathrm{x}^{(\mathrm{i})}_{\mathrm{f}} - \mathrm{x}_{\mathrm{k}}}{\mathrm{d}} & -\dfrac{\mathrm{y}^{(\mathrm{i})}_{\mathrm{f}} - \mathrm{y}_{\mathrm{k}}}{\mathrm{d}} & 0 & \cdots & \dfrac{\mathrm{x}^{(\mathrm{i})}_{\mathrm{f}} - \mathrm{x}_{\mathrm{k}}}{\mathrm{d}} & \dfrac{\mathrm{y}^{(\mathrm{i})}_{\mathrm{f}} - \mathrm{y}_{\mathrm{k}}}{\mathrm{d}} & \cdots \\ \dfrac{\mathrm{y}^{(\mathrm{i})}_{\mathrm{f}} - \mathrm{y}_{\mathrm{k}}}{\mathrm{d}^2} & -\dfrac{\mathrm{x}^{(\mathrm{i})}_{\mathrm{f}} - \mathrm{x}_{\mathrm{k}}}{\mathrm{d}^2} & -1 & & \dfrac{\mathrm{y}^{(\mathrm{i})}_{\mathrm{f}} - \mathrm{y}_{\mathrm{k}}}{\mathrm{d}^2} & \dfrac{\mathrm{x}^{(\mathrm{i})}_{\mathrm{f}} - \mathrm{x}_{\mathrm{k}}}{\mathrm{d}^2} & \end{bmatrix} \tag{31}$$

with $d = \sqrt{\left(x^{(i)}_f - x_k\right)^2 + \left(y^{(i)}_f - y_k\right)^2}$ $d = \sqrt{\left(x^{(i)}_f - x_k\right)^2 + \left(y^{(i)}_f - y_k\right)^2}$.

The dimensionality of the controllable subspace is dim x_r=3, and for the specific case in which only one landmark is available, a basis for the controllable subspace is simply

$$\mathbf{E}_{\mathrm{Im}\mathcal{C}} = \begin{pmatrix} \mathbf{I} \\ \mathbf{0}_{2\times3} \end{pmatrix}$$

The dimensionality of the observable subspace is, for this particular configuration, rank $\mathcal{O}$=3. This last result is easily verified with simple symbolic manipulation of the expression resulting from substituting expressions (29) and (31) in the observability matrix

$$\mathcal{O} = \begin{bmatrix} \mathbf{H} \\ \mathbf{HF} \\ \vdots \\ \mathbf{HF}^{dim\mathbf{x}-1} \end{bmatrix} \tag{32}$$

Possible bases for $\mathrm{Im}\mathcal{O}^\top$, and for the null space of $\mathcal{O}$ (the unobservable subspace) are

$$\mathbf{E}_{\mathrm{Im}\mathcal{O}^\top} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \end{pmatrix} \quad \mathbf{E}_{\mathrm{Ker}\mathcal{O}} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}$$

The only independently observable state is the one associated to the robot orientation ϕ. The other four states, the Cartesian coordinates of the robot and landmark locations span a space of dimension 2. Even when $\mathrm{Im}\mathcal{C}$ and $\mathrm{Im}\mathcal{O}^\top$ both span $\mathbb{R}^3$, we see that the inequality $\mathrm{Im}\mathcal{C} \neq \mathrm{Im}\mathcal{O}^\top$ still holds, as in the case of the monobot. That is, the observable and controllable subspaces for this one-landmark 3dof-robot SLAM problem correspond to different three-dimensional subspaces in $\mathbb{R}^5$; and, their intersection represents the only fully controllable and observable state, i.e., the robot orientation.

6. Complete Observability

In Section 4 we characterized the unobservable subspace in SLAM as the subspace spanned by the null eigenvectors of the total Fisher information matrix. Furthermore, we showed in Section 5 how the unobservable part of the state space is precisely a linear combination of the landmark and robot pose estimates.

In order to gain full observability we propose to extend the measurement model doing away with the constraint imposed by full correlation by letting one landmark serve as a fixed global reference, with its localization uncertainty independent of the vehicle pose. The principle behind is that full observability requires an uncorrelated measurement Jacobian, or equivalently, a full rank Fisher information matrix. The use of a fixed global reference as anchor guarantees that.

The measurement model of a global reference fixed at the origin, for the nonlinear vehicle from Figure 4. is

$$h^{(0)} = \begin{bmatrix} \sqrt{x_k^2 + y_k^2} + w_{r,k} \\ \tan^{-1}\left(\dfrac{y_k}{x_k}\right) - \theta_k + \dfrac{\pi}{2} + w_{\beta,k} \end{bmatrix} \tag{33}$$

and its corresponding Jacobian is

$$H_0 = \begin{bmatrix} \frac{x_k}{\sqrt{x_k^2 + y_k^2}} & \frac{y_k}{\sqrt{x_k^2 + y_k^2}} & 0 & 0 & 0 & \cdots \\ -\frac{y_k}{x_k^2 + y_k^2} & \frac{x_k}{x_k^2 + y_k^2} & -1 & 0 & 0 & \end{bmatrix} \tag{34}$$

The symbolic manipulation of

$$H = \begin{bmatrix} H_0 \\ H_i \end{bmatrix} \tag{35}$$

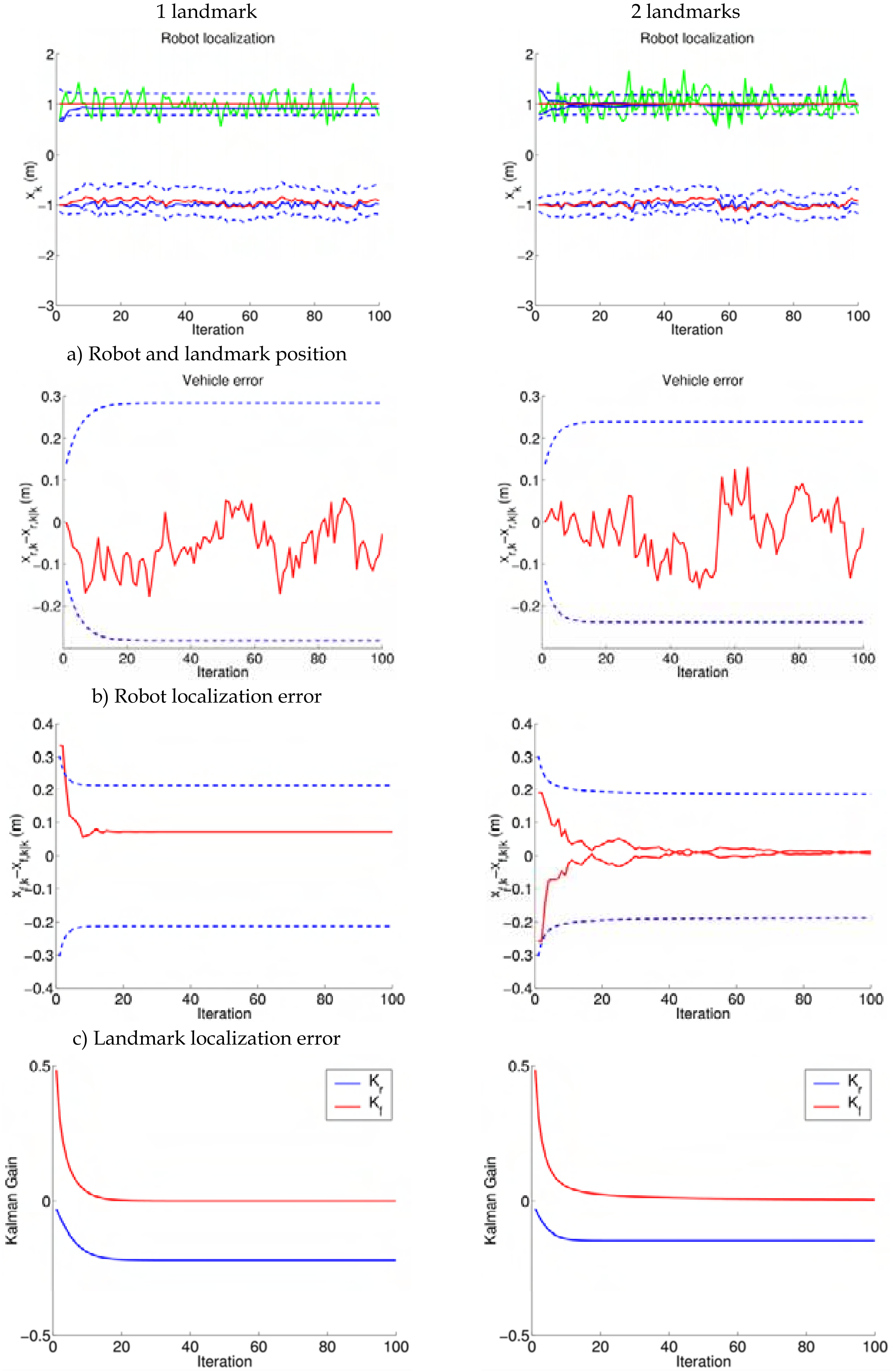
with a commercial algebra package, produces a full rank observability matrix. That is, for the linearized nonholonomic velocity controlled planar mobile robot platform used, the simultaneous measurement of one anchor as global reference, and any other landmark, is sufficient to attain full observability in SLAM.

7. Suboptimal Filter Stability

When a stochastic system is partially controllable, such as in the case of SLAM, the Gaussian noise sources v_k do not affect all of the elements of the state space. The diagonal elements of **P** corresponding to these incorruptible states will be driven to zero by the Kalman filter, and once this happens, these estimates will remain fixed and no further observations will alter their values. The dynamics of the model assume the landmarks are fixed elements, for which no process noise is considered. Therefore, their associated noise covariance (its determinant) will asymptotically tend to zero (Dissanayake et al., 2001). The filter gain for the landmark states will also tend to zero. Figure 5 shows two new simulations for a linear SLAM case, a monobot under Brownian motion with one and two landmarks. The simulations show the evolution of the localization errors for both the monobot and the landmarks, and the reduction of the landmark part of the Kalman gain, due to the uncontrollability of the system. The only way to remedy this situation is to add a positive definite pseudo-noise covariance to those incorruptible states (Bar-Shalom et al., 2001).

In Figure 5, the vehicle location is indicated by the darkest curve at the -1m level in the first row of plots. In the same set of plots, and close to it, is a lighter curve indicating the vehicle location estimate as computed by the filter, along with 2σ bounds on such estimate shown as dotted lines. The dark straight lines at the 1m level indicate the landmark location estimates; and the lighter curves are noise corrupted signals of sensor measurements. Also shown, are a pair of dotted lines for 2σ bounds on the landmark location estimates. The second row of plots shows the vehicle location error only, and its corresponding variance, also on the form of 2σ dotted bounds.

See how the localization error has non-zero mean due to partial observability, an undesirable feature in Kalman filtering. The third row shows non-zero mean landmark state estimate errors. And, the last row shows the Kalman filter gains both for the vehicle and landmark revision terms. The Kalman gains for the revision of the landmark estimates rapidly tend to zero, the reason being that these states are uncontrollable.



a) Robot and landmark position

b) Robot localization error

c) Landmark localization error

d) Kalman gains

Figure 5. Partially observable SLAM for a monobot during Brownian motion with 100 iterations (see text)

7.1 *O*(*N*) but Unstable Partially Observable SLAM

One way to add pseudo-noise to the model is by diagonalizing the state error covariance matrix (Guivant and Nieto, 2001, 2003, Julier, 2003). This technique is often used to reduce the time complexity of the algorithm from O(N^2) to *O*(*N*). The result is a suboptimal filter that will compute inflated estimates for the vehicle and landmark covariances, that has the computational advantage of being uncorrelated. The addition of a covariance term $\Delta \mathbf{P}$ to the a priori state covariance estimate

$$P_{k+1|k} = FP_{k|k}F^{\mathrm{T}} + GQG^{\mathrm{T}} + \Delta P \tag{36}$$

is equivalent to providing a new form to the plant noise Jacobian $G' = \begin{bmatrix} G & I \end{bmatrix}$

$$P_{k+1|k} = FP_{k|k}F^{\mathrm{T}} + G' \begin{bmatrix} Q & \\ & \Delta P \end{bmatrix} G'^{\mathrm{T}} \tag{37}$$

ΔP may be chosen, for example, such as to minimize the trace of a resulting block diagonal P in (36) (Julier, 2003).

Choosing a full rank ΔP is equivalent to having noise input to more states than those that can be observed with the filter. In this case, because of partial observability, both vehicle and landmark variance estimates become unbounded. Figure 6 shows this for the same monobot experiment as in the previous simulation. This phenomena was first observed in (Julier, 2003) using relative maps. Not only both the vehicle and landmark state estimation variances become unbounded. Also, thanks to the full controllability of the system, the Kalman gain for the revision of the landmark states is greater than zero; but still, does not converge to a steady state value. We believe that the addition of pseudo-noise should be performed only at most, in the amount of states equal to the dimension of the observable subspace.

7.2 O(N) and Stable Partially Observable SLAM

One solution to the problem of instability during covariance inflation, is to decorrelate only the landmark state estimates, and to preserve all vehicle to landmark correlations (Vidal-Calleja et al., 2004a).

$$\Delta P = \begin{bmatrix} 0 & \\ & Q_f \end{bmatrix} \tag{38}$$

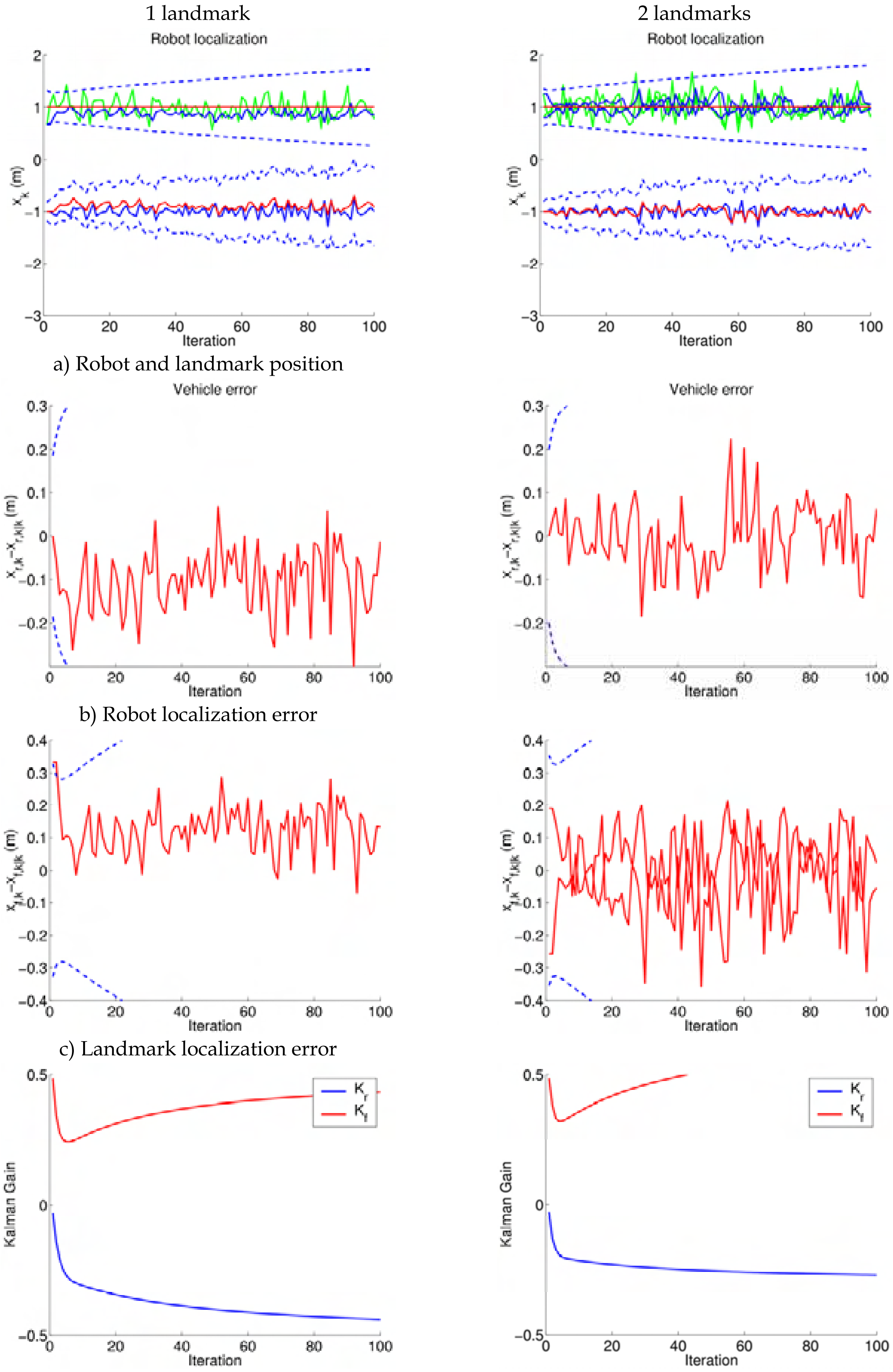
such that P_f +Q_f , the map part of the state error covariance, is block diagonal.

Figure 7 shows a partially observable monobot under Brownian motion for which only the landmark part of the state error covariance matrix has been decorrelated. The algorithm does converge to a steady state solution under this circumstances, and still can be implemented in real time. The one landmark case is identical than the original case, since a linear one landmark map is already diagonal (scalar actually).

For the two-landmark case, the landmark variance estimate is greater than the optimal solution shown in the third column in Figure 5 since the covariance has been inflated during decorrelation. Furthermore, now that the system is controllable, the Kalman gains for the landmark state estimates do not become zero, and they converge to a steady state value.

Moreover, we $\mathcal{O}$, that the covariance inflation suboptimal partially observable SLAM converges only when

$$\text{rank } \Delta \text{P} \le \text{rank } \mathcal{O} \tag{39}$$



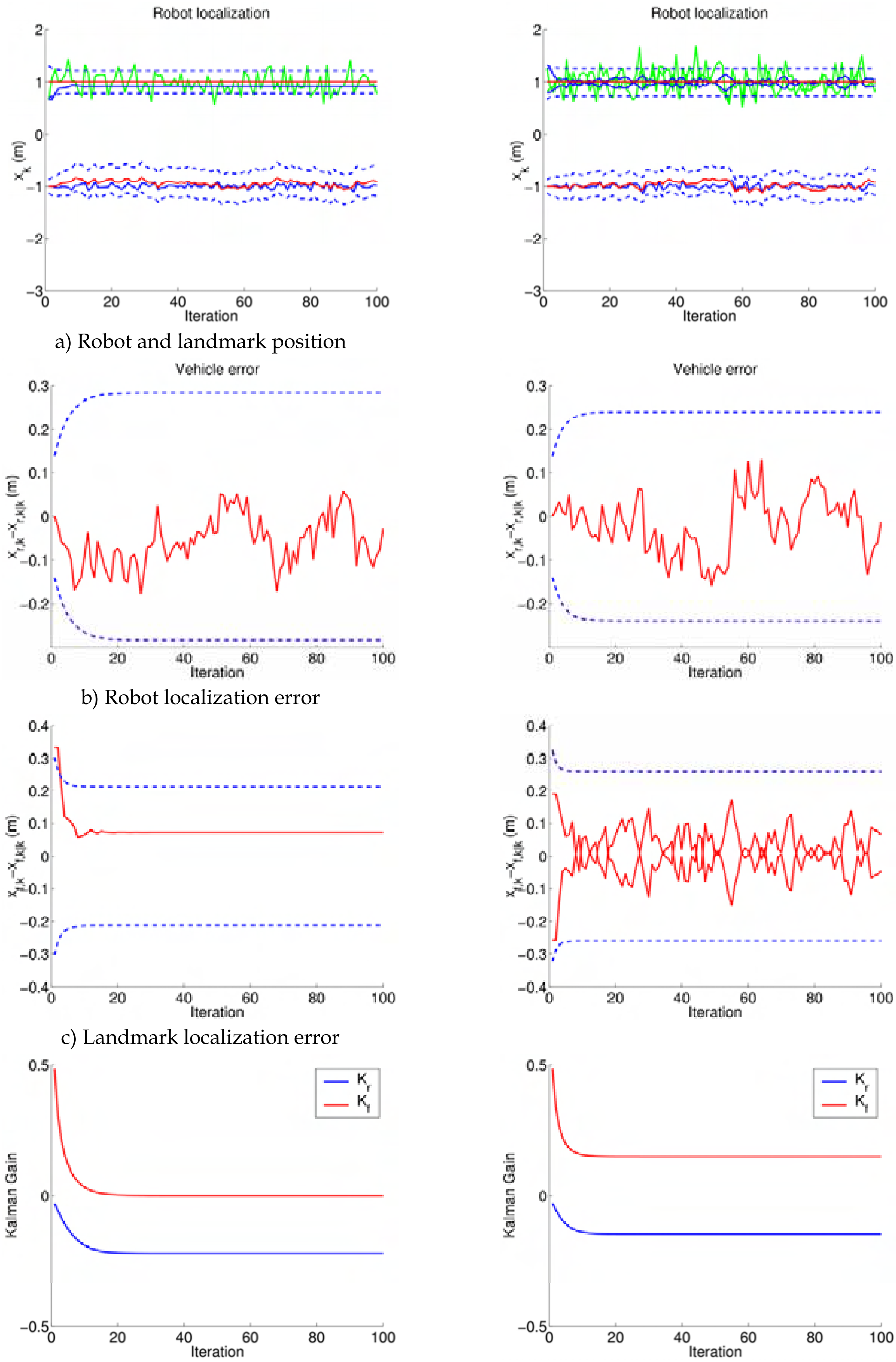
a) Robot and landmark position

b) Robot localization error

c) Landmark localization error

d) Kalman gains

Figure 6. Partially observable SLAM for a Brownian motion monobot with 100 iterations. The entire state error covariance is decorrelated with the minimal trace solution (Julier, 2003). By decorrelating the entire state error covariance matrix, the covariance estimates become unbounded



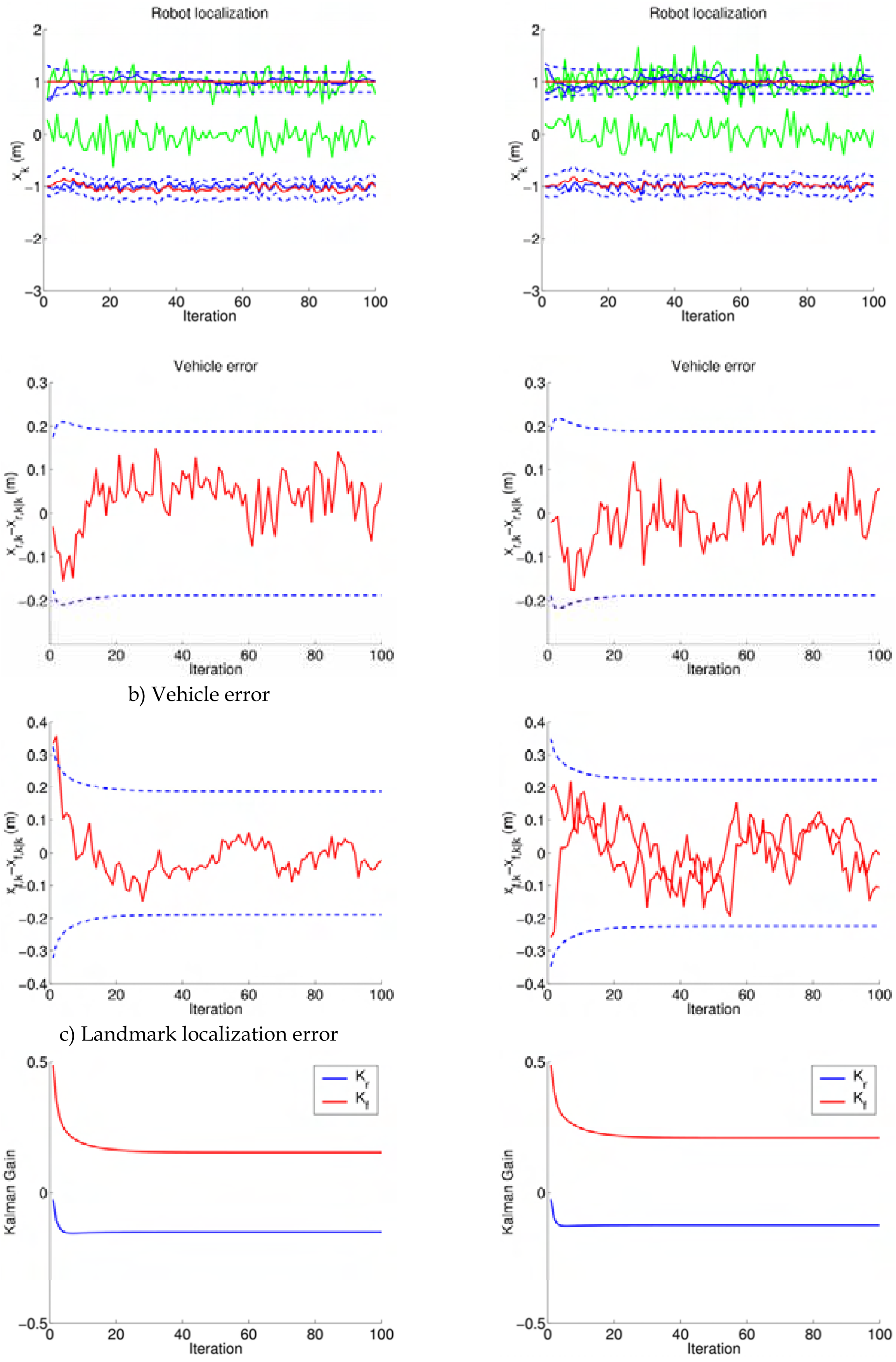
a) Robot and landmark position

b) Robot localization error

c) Landmark localization error

d) Kalman gains

Figure 7. Partially observable SLAM for a Brownian motion monobot with 100 iterations. The state error covarance is decorrelated only for the landmark part of the state vector, with the minimal trace solution. By decorrelating only the map part of the state error covariance matrix, we preserve filter stability

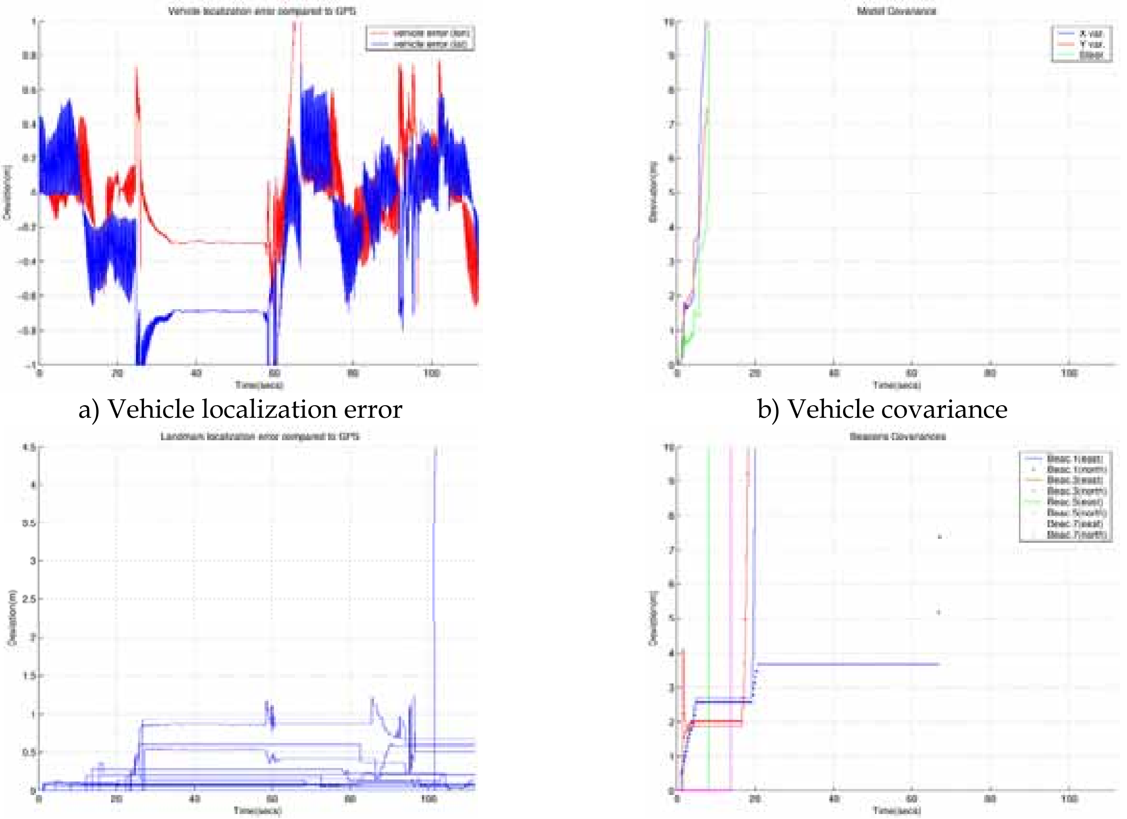


b) Vehicle error

c) Landmark localization error

d) Kalman gains

Figure 8. Fully observable SLAM for a Brownian motion monobot with 100 iterations. The entire state error covarance is decorrelated with the minimal trace solution. In the linear case, it is possible to decorrelate the entire state error covariance matrix, and still preserve filter stability



a) Vehicle localization error

b) Vehicle covariance

c) Landmark localization errors

d) Landmark covariance

Figure 9. Partially observable SLAM for a car-like vehicle at the University of Sydney Car Park. The entire state error covariance matrix is decorrelated with the minimal trace solution (Julier, 2003)

7.3 O(N) and Stable Fully Observable SLAM

Consider now the fully observable case from the previous Section. If we add pseudo-noise to the vehicle as well as to the landmark states, the covariance will reach a steady-state value, and the Kalman gain will not be zero, at least, in the linear case. Figure 8 shows this results diagonalizing the whole state error covariance (not only the landmark part of **P**).

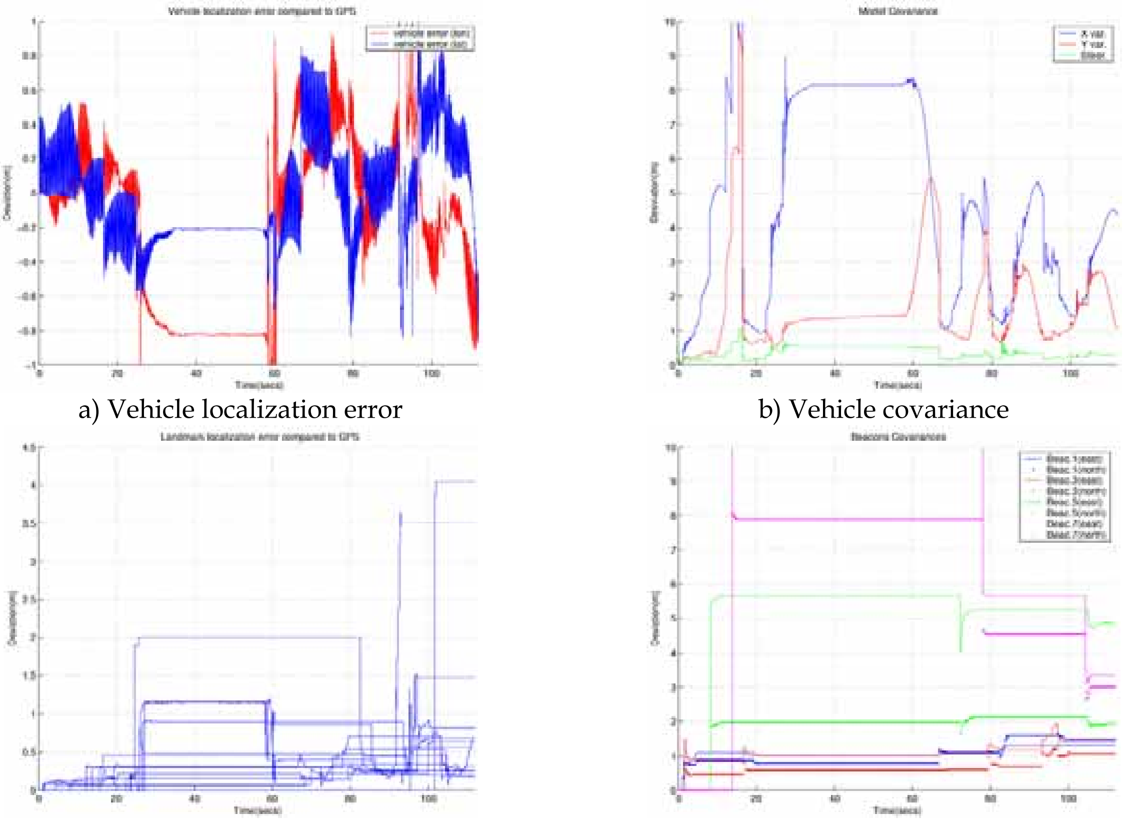
In this latter experiment, the state error variances reach lower values than those in the partially observable case. The solution of the Riccati equation is now independent of the initial covariance estimate $\mathbf{P}_{0|0}$. We have observed experimentally however, that with a nonlinear vehicle model, it is best to also decorrelate only the map part of the state error covariance, even in the fully observable case.

7.4 Experimental Results

We show now results on a series of experiments for a nonlinear vehicle with an also nonlinear measurement model, using the ACFR - University of Sydney database (Nebot et al., 2002). The test run used corresponds to a car-like vehicle at the University Car Park. The landmarks used are tree trunks, as measured with a laser range finder. The reconstructed maps are compared to GPS ground truth for accuracy. The first experiment corresponds to a typical partially observable SLAM run, in which the entire state error covariance is being decorrelated as discussed in Section 7.1. Figure 9 plots results on this run, showing in rows b) and d) unbounded covariances both for the vehicle and landmark state estimates, due to the naïve covariance inflation method used.

The second experiment corresponds to the same partially observable SLAM conditions, but decorrelating only the map part of the state error covariance. Adding pseudo-noise to the landmark states during the inflation procedure amounts to making the system controllable; and doing so for as many states as those observable, produces both vehicle and landmark bounded state covariance estimates. This is shown in Figure 10, frames b) and d). Figure 12 frame a) shows the actual vehicle path and landmark location estimates recovered by the algorithm, compared to GPS ground truth for the beacons. Note that even when the "relative" map is consistent (Dissanayake et al., 2001), it is slightly rotated and shifted from the actual beacon locations. The amount of this shift depends on the initial vehicle uncertainty, i.e., the initial filter conditions, and can be seen in Figure 10, frame c).

The last experiment shown corresponds to a fully observable SLAM run (using the first observed beacon as an anchor), and also decorrelating only the map part of the state error covariance. In this case, the vehicle and landmark covariance estimates do not depend on the initial filter conditions, and thus are significantly reduced. This is shown in frames b) and d) in Figure 11. The absolute landmark estimate error is also significantly reduced, as shown in Figure 11, frame c). Figure 12 frame b) shows the actual vehicle path and landmark estimates as recovered by the filter. The beacon shown in the center of the plot is used as an anchor to the map, and no state estimate is computed for it. This last map was obtained with a suboptimal linear-time SLAM algorithm that has both bounded covariance estimates, and independence on the filter initial conditions; thus producing a fast and accurate absolute map.



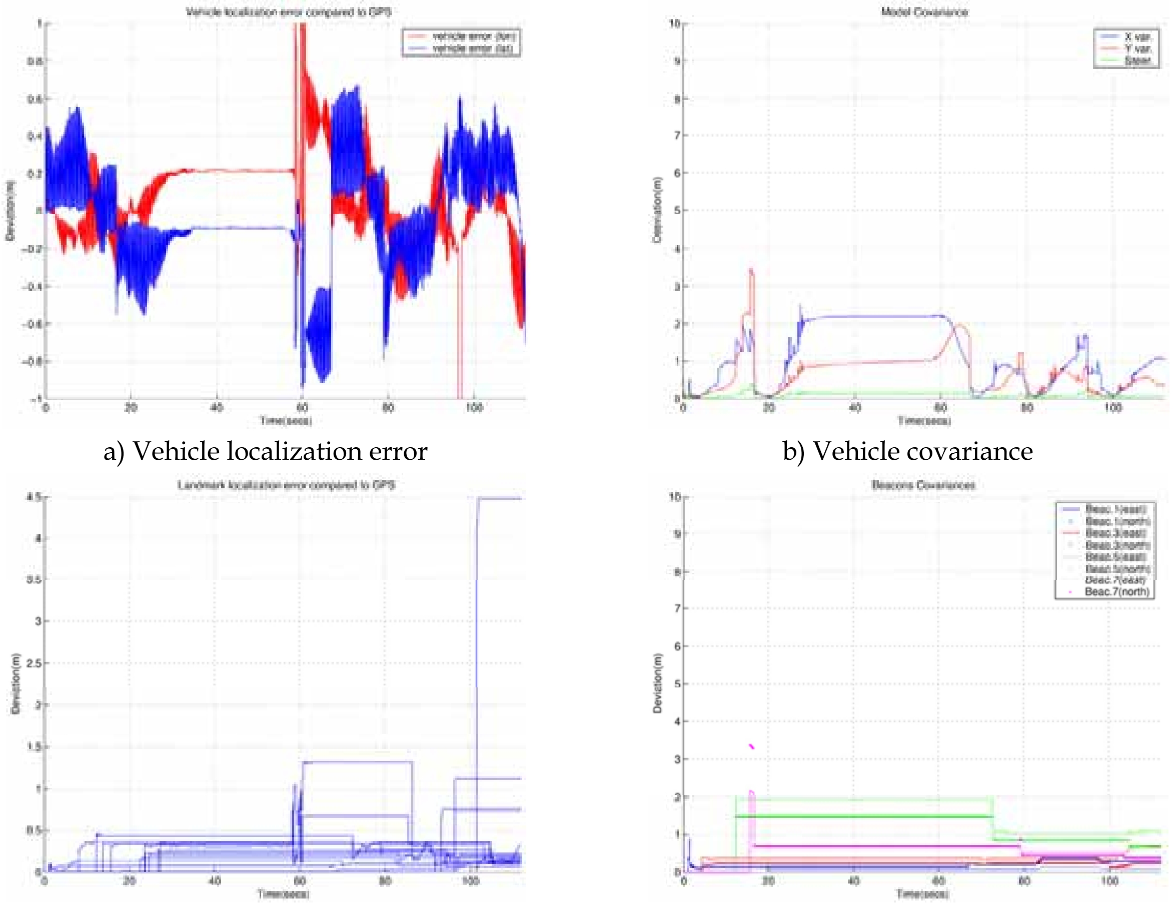
a) Vehicle localization error

b) Vehicle covariance

c) Landmark localization errors

d) Landmark covariance

Figure 10. Partially observable SLAM for a car-like vehicle at the University of Sydney Car Park. Only the map part of the state error covariance matrix is decorrelated with the minimal trace solution. By adding controllability to as many states as those that are observable, the filter remains stable, and the estimated covariances remain bounded



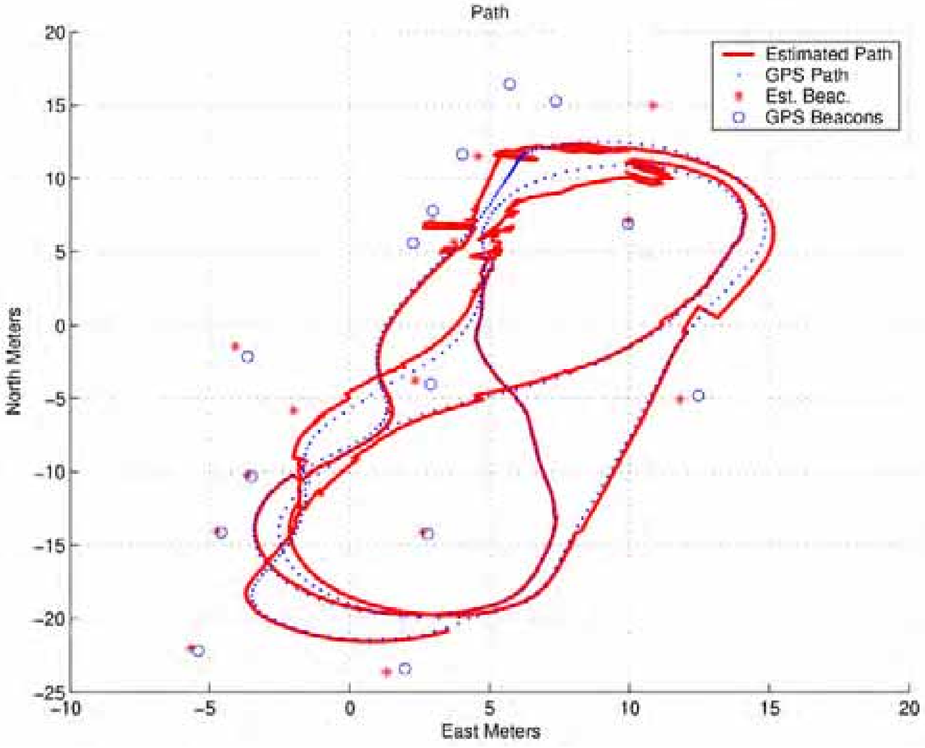
a) Vehicle localization error

b) Vehicle covariance

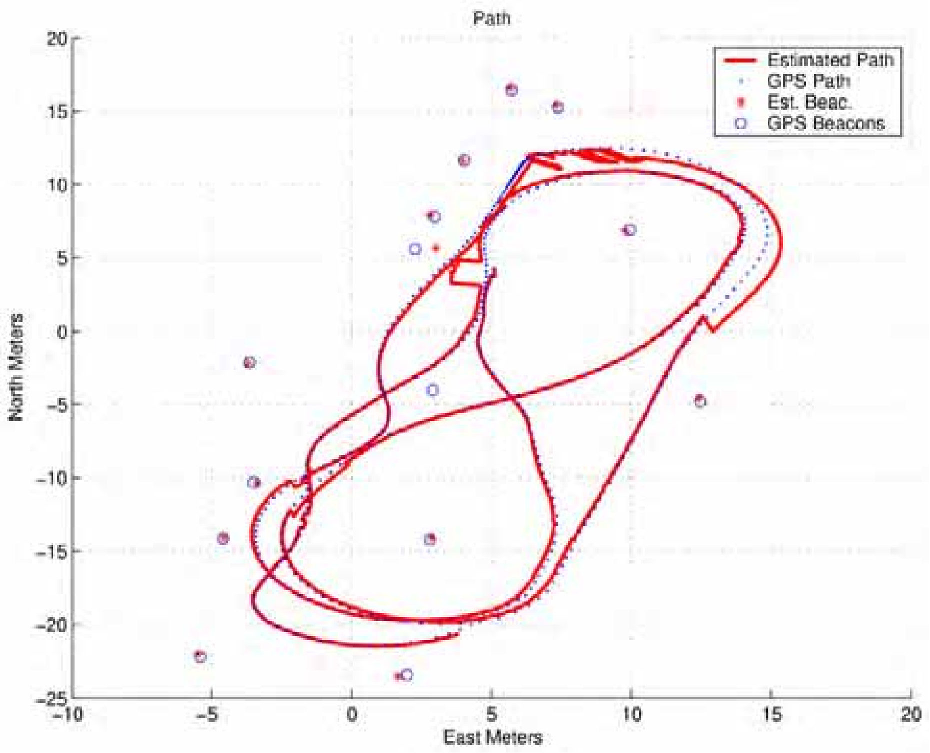
c) Landmark localization errors

d) Landmark covariance

Figure 11. Fully observable SLAM for a car-like vehicle at the University of Sydney Car Park. Only the map part of the state error covariance is decorrelated with the minimal trace solution. Full observability guarantees independence of the filter initial conditions, and an accurate absolute map is obtained, with smaller covariance estimates than its relative counterpart



a)



b)

Figure 12. Vehicle path and landmark location estimates, compared to GPS ground truth for an a) partially observable suboptimal SLAM run, and a b) fully observable suboptimal SLAM run; both with decorrelation of only the map part of the state error covariance matrix

8. Conclusions

We have shown that full correlation of the map model in the Kalman Filter based approach to Simultaneous Localization and Map Building hinders full observability of the state estimate. A unit norm eigenvalue for the matrix **F-KH** makes the state error estimate converge to a non zero mean constant bounded value in the linear case SLAM. Marginal stability of such partially observable system produces also at least one *psd* solution to the steady state Riccati equation for the covariance error, provided the initial conditions of **P** are also *psd*. Partial observability makes the final map dependant on the initial observations. This situation can easily be remedied either by anchoring the map to the first landmark observed, or by having an external sensor that sees the vehicle at all times.

Suboptimal techniques to improve the speed of the algorithm include covariance inflation methods to diagonalize the state error covariance matrix. These techniques may lead to instability if pseudo-noise is added in a higher state dimensionality than what can be observed. We propose in this Chapter to diagonalize only the map part of the state error covariance, thus guaranteeing convergence of **P**, and at the same time obtaining an $O(N)$ algorithm.

9. References

J. Andrade-Cetto and A. Sanfeliu. The effects of partial observability when building fully correlated maps. To appear in *IEEE Transactions on Robotics*.

Y. Bar-Shalom, X. Rong Li, and T. Kirubarajan. *Estimation with Applications to Tracking and Navigation*. John Wiley & Sons, New York, 2001.

M. W. M. G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, and M. Csorba. A solution to the simultaneous localization and map building (SLAM) problem. *IEEE Transactions on Robotics and Automation*, 17 (3):229–241, June 2001.

P. W. Gibbens, G. M. W. M. Dissanayake, and H. F. Durrant-Whyte. A closed form solution to the single degree of freedom simultaneous localisation and map building (SLAM) problem. In *Proceedings of the IEEE International Conference on Decision and Control*, pages 408–415, Sydney, December 2000.

J. E. Guivant and E. M. Nieto. Optimization of simultaneous localization and map-builidng algorithm for real-time implementation. *IEEE Transactions on Robotics and Automation*, 17 (3):242–257, June 2001.

J. E. Guivant and E. M. Nieto. Solving computational and memory requirements of feature-based simultaneous localization and mapping algorithms. *IEEE Transactions on Robotics and Automation*, 19 (4):749–755, August 2003.

S. J. Julier. The stability of covariance inflation methods for SLAM. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2749–2754, Las Vegas, October 2003.

T. Kailath, A. H. Sayed, and B. Hassibi. *Linear Estimation*. Information and System Sciences Series. Prentice Hall, Upper Saddle River, 2000.

S. J. Kim. *Efficient Simultaneous Localization and Mapping Algorithms using Submap Networks*. PhD thesis, Massachusetts Institute of Technology, Cambridge, June 2004.

E. Nebot, J. Guivant, and J. Nieto. ACFR, experimental outdoor dataset, 2002.

J. Neira and J. D. Tardós. Data association in stochastic mapping using the joint compatibility test. *IEEE Transactions on Robotics and Automation*, 17 (6):890–897, December 2001.

R. C. Smith and P. Cheeseman. On the representation and estimation of spatial uncertainty. *International Journal of Robotics Research*, 5 (4):56–68, 1986.

R. Todling. Estimation theory and foundations on atmosferic data assimilation. Technical Report DAO Note 1999-01, Data Assimilation Office. Goddard Space Flight Center, 1999.

T. Vidal-Calleja, J. Andrade-Cetto, and A. Sanfeliu. Conditions for suboptimal filter stability in SLAM. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 1, pages 27–32, Sendei, September 2004a.

T. Vidal-Calleja, J. Andrade-Cetto, and A. Sanfeliu. Estimator stability analysis in SLAM. In *Proceedings of the 5th IFAC/EURON Symposium on Intelligent Autonomous Vehicles*, Lisbon, July 2004b.

Neural Networks in Mobile Robot Motion

Danica Janglova

1. Introduction

Over the last few years, a number of studies were reported concerning a machine learning, and how it has been applied to help mobile robots to improve their operational capabilities. One of the most important issues in the design and development of intelligent mobile system is the navigation problem. This consists of the ability of a mobile robot to plan and execute collision-free motions within its environment. However, this environment may be imprecise, vast, dynamical and either partially or non-structured. Robots must be able to understand the structure of this environment. To reach their targets without collisions, the robots must be endowed with perception, data processing, recognition, learning, reasoning, interpreting, decision-making and action capacities. The ability to acquire these faculties to treat and transmit knowledge constitutes the key of a certain kind of artificial intelligence. Reproduce this kind of intelligence is, up to now, a human ambition in the construction and development of intelligent machines, and particularly autonomous mobile robots. To reach a reasonable degree of autonomy two basic requirements are sensing and reasoning. The former is provided by on-board sensory system that gathers information about robot with respect to the surrounding scene. The later is accomplished by devising algorithms that exploit this information in order to generate appropriate commands for robot. And with this algorithm we will deal in this chapter.

We report on the objective of the motion planning problem well known in robotics. Given an object with an initial location and orientation, a goal location and orientation, and a set of obstacles located in workspace, the problem is to find a continuous path from the initial position to the goal position, which avoids collisions with obstacles along the way. In other words, the motion planning problem is divided into two sub-problems, called 'Findspace' and 'Findpath' problem. For related approaches to the motion planning problem see reference (Latombe, 1991). The findspace problem is construction the configuration space of a given object and some obstacles. The findpath problem is in determining a collision-free path from a given start location to a goal location for a robot.

Various methods for representing the configuration space have been proposed to solve the findpath problem (Brady et al., 1982), (Latombe, 1991), (Vörös, 2002). The major difficulties in the configuration space approach are: expensive computation is required to create the configuration space from the robot shape and the obstacles and the number of searching steps increases exponentially with the number of nodes. Thus, there is a motivation to investigate the use of parallel algorithms for solving these problems, which has the potential for much increased speed of calculations. A neural network is a massive system of parallel-distributed processing elements connected in a graph topology. Several researchers have tried to use neural networks techniques for solving the find path problem (Bekey & Goldberg, 1993).

In this chapter we introduce a neural networks-based approach for planning collision-free paths among known stationary obstacles in structured environment for a robot with translational and rotational motion. Our approach basically consists of two neural networks to solve the findspace and findpath problems respectively. The first neural network is a modified principal component analysis network, which is used to determine the "free space" from ultrasound range finder data. Moving robot is modeled as a two-dimensional object in this workspace. The second one is a multilayer perceptron, which is used to find a safe direction for the next robot step on the collision-free path in the workspace from start configuration to a goal configuration while avoiding the obstacles.

The organization of the chapter is as follows: section 2 brings out the briefly description of neural network applications in robotics. Our approach to solving the robot motion problem is given in section 3. Our method of motion planning strategy, which depends in using two neural networks for solving the findspace problem and the findpath problem respectively will be described in section 4. Simulation results will be included in section 5. Section 6 will summarize our conclusions and gives the notes for our further research in this area.

2. Neural Networks in Robotics

The interest in neural network stems from the wish of understanding principles leading in some manner to the comprehension of the basic human brain functions, and to building the machines that are able to perform complex tasks. Essentially, neural network deal with cognitive tasks such as learning, adaptation, generalization and optimization. Indeed, recognition, learning, decision-making and action constitute the principal navigation problems. To solve these problems fuzzy logic and neural networks are used. They improve the learning and adaptation capabilities related to variations in the environment where information is qualitative, inaccurate, uncertain or incomplete. The processing of imprecise or noisy data by the neural networks is more efficient than classical techniques because neural networks are highly tolerant to noises.

A neural network is a massive system of parallel-distributed processing elements (neurons) connected in a graph topology. Learning in the neural network can be supervised or unsupervised. Supervised learning uses classified pattern information, while unsupervised learning uses only minimum information without preclassification. Unsupervised learning algorithms offer less computational complexity and less accuracy than supervised learning algorithms. Then former learn rapidly, often on a single pass of noisy data. The neural network could express the knowledge implicitly in the weights, after learning. A mathematical expression of a widely accepted approximation of the Hebbian learning rule is

$$w_{ij}(t+1) = w_{ij}(t) + \eta\, x_i(t) y_j(t) \tag{1}$$

where x_i and y_j are the output values of neurons i and j, respectively, which are connected by the synapse w_{ij} and η is the learning rate (note that x_i is the input to the synapse).

Survey of types, architectures, basic algorithms and problems that may be solved using some type of neural networks is presented in (Jain et al., 1996). The applications of neural networks for classification and pattern recognition are good known. Some interesting solutions to problems of classification in the robot navigation domain were successfully solved by means of competitive type of neural networks (Bekey & Goldberg, 1993). Using of competitive neural networks in control and trajectory generation for robots we may find in the book as well as using of neural network for sensor data processing in map updating and learning of the robot trajectories. For the obstacle avoidance purposes recurrent type of neural network was used with the gradient back-propagation technique for training the network (Domany et al., 1991). The using of supervised neural network for robot navigation in partially known environment is presented in (Chochra, 1997). An interested solution with using of Jordan architecture of neural network is described in (Tani, 1996). Here the robot learns internal model of the environment by recurrent neural network, it predicts succession of sensors inputs and on the base of the model it generates navigation steps as a motor commands. The solution of the minimum path problem with two recurrent neural networks is given in (Wang, 1998). Solutions that use the learning ability of the neural network with fuzzy logic for representation of the human knowledge applied to robot navigation also exist; see (Kim & Trivedi, 1998). The complex view for solution of the navigation problem of the autonomous vehicles gives (Hebert et al., 1997). Team of researches CMU here presents results from designing of autonomous terrain vehicle. For learning the path from vision system data and for obstacle avoidances algorithms using laser range finder data and different types of neural networks.

Our first work concerned the using neural networks for object classification in the map of the robot environment was using the cluster analysis with range finder data (Uher & Považan, 1998). This acquiring knowledge we extend for using neural network in the algorithm of the robot motion planning.

3. The Proposed Approach

3.1 The Basic Motion Planning Problem

Let A be a rigid object, a robot, moving in a workspace W, represented as a subspace of R^N, with N=2 or 3. Let O_1,..,O_m be fixed rigid objects distributed in W, called obstacles. Assume that both the geometry and the location of A, O_1,...,O_m, in W is known.

The problem is: Given an initial position and orientation of A in W, generate a path specifying a contiguous sequence of positions and orientations of A avoiding collision with O_i's, starting at the initial position and orientation, and terminating at the goal position and orientation. Report the failure if no such path exists.

3.2 Environment Representation

In general, we consider the case when A is a two-dimensional object that translates and rotates in $W=R^2$. A grid map will represent the working environment. The grid map is an M x N matrix with each element representing the status $S_{i,j}$ of an individual grid; $S_{i,j} = 1$, if its interior intersects with the obstacle region and $S_{i,j} = 0$, if its interior does not intersect the obstacle region.

A configuration of an arbitrary object is a specification of the position of every point in this object relative to a fixed reference frame. In addition, let F_A and F_W be Cartesian frame embedded in A and W, respectively. Therefore, a configuration of A is a specification of the position (x,y) and orientation θ of F_A with respect to F_W . Throughout the chapter we make use of the localization system (Považan et al. 1995) providing the robot with its absolute position with respect to a fixed inertial frame. The configuration space of A is the space of all the configurations of A. Let the resolution in x-axis, y-axis and orientation is M, N, and K respectively. A rectangle $r_{i,j,k}$ is model of the object A located by (x_i, y_j, θ_k) and it represents the region $[x_i - w_x/2, x_i + w_x/2] . [y_j - w_y/2, y_j + w_y/2] . [\theta_k - \Delta\theta/2, [\theta_k + \Delta\theta/2]$, where w_x is the width, w_y is the height and $\Delta\theta = \pi/K$.

3.3 Motion Planning Algorithm

Philosophy of our algorithm appear from motion of human in the environment when he is moving between obstacles on the base of his eyes view and he make already the next step to the goal in the free space. Analogically, our robot will move safely in environment on the base of the data "visible" with scanning ultrasound range finder (Uher & Kello, 1999). First must "mapping" the workspace from measured data and find the free space for robot motion and then determines the next robot azimuth for the safe step to the goal. For the solution of this problems we use neural networks technique. We use the measured range finder data in the learning workspace for mapping the front robot workspace by the first neural network finding the free space segment. This segment is used as an input to the second neural network both with the goal location, which is used to determine the direction of the proposed next navigation step for moving the robot.

The algorithm is of an iterative type. In each iteration, the last orientation of the moving robot is stored and the neural network selects the direction of the next navigation step. To determine the direction, the status in the partial configuration space is required; the map from range finder is proposed to give this status. Moreover, a control unit is used to provide information required by neural networks to control the operating sequence and to check the reachability of the goal configuration from the start configuration.

Our motion planning algorithm can be summarized as follows:

- Specify the object, environment information and the start and goal configurations.
- Set the current object orientation equal to the goal orientation.
- Activate range finder via control unit to determine the local part of the map of the workspace.
- Initialize the first neural network, which will use the measured data from range finder. The neural network is iterated until the weights and the outputs converged to the returned one free space segment.
- Activate the second neural network. It returns the direction θ_k of next robot motion step.
- Generate the robot motion path in the direction θ_k and go to the step 3.

4. Principles of Proposed Algorithm

4.1 The FindSpace Problem Using Neural Network

Therefore we use the sensor data from the environment and the classical findspace problem in our strategy was transform to the procedure 'learning your environment'. The robot has in any position in workspace information about its distances to the all objects in this workspace. We use this information in first neural network that learns these situations and in any position gives the free segment of space for safe path as output. The neural network using for the findspace problem is principal component analysis network (PCA). This neural network uses as inputs the data measured by the range finder. The output is free segment of the robot workspace. Principal component analysis networks combine unsupervised and supervised learning in the same topology (see Fig. 1).

This neural network uses as inputs the data measured by the range finder. The output is free segment of the robot workspace.

Principal component analysis is an unsupervised linear procedure that finds a set of uncorrelated features from the input. A feed-forward network is used to perform the nonlinear classification from these components. PCA is a data reduction method, which condenses the input data down to a few principal components. The number of principal components selected will be a compromise between training efficiency and accurate results.

It is not possible to provide a general formula for selecting an appropriate number of principal components for a given application. Both learning rule choices are normalized implementations of the Hebbian learning rule. Straight Hebbian learning must be utilized with care, since it may become unstable if the inputs are not properly normalized. The network has four layers - input, PCA, hidden and output layer. The learning is realized in two phases. In the first place an unsupervised linear procedure gets a set of uncorrelated features from the inputs and selects a few principal components. These components in hidden layer feed-forward supervised part gives the output.

The PCA neural network learns by generalized Hebbian rule. First updated the synapse weights w_{ij} (see Fig. 2) to a small random number and learning parameter η to a positive small number.

Then for n=1 are calculating outputs y_j and the changes of the weights. The outputs y_j from PCA network are given by (2)

$$y_j(n) = \sum_{i=0}^{p-1} w_{ij}(n) x_i(n) \qquad j = 0,1,\ldots,m-1 \tag{2}$$

The changes of the weights during the learning are calculated by modification of Hebbian rule (3)

$$\begin{aligned} &\Delta w_{ji}(n) = \eta \left[y_j(n) x_i(n) - y_j(n) \sum_{k=0}^{j} w_{ki}(n)\, y_k(n) \right] \\ &i = 0,1,\ldots, p-1 \\ &j = 0,1,\ldots,m-1 \end{aligned} \tag{3}$$

The calculations iterate up to the weights w_{ij} are stable.

At second phase the learning of the network are realised by back-propagation algorithm. Here updated the synaptic weights and threshold neuron coefficients.

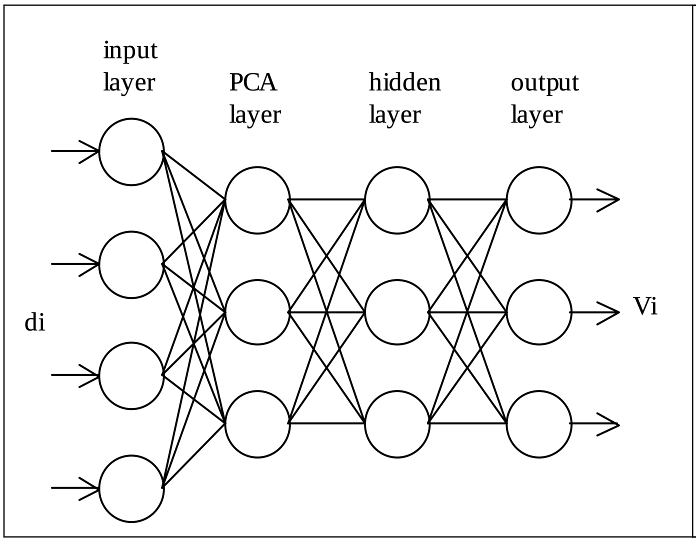


Figure 1. PCA neural network topology

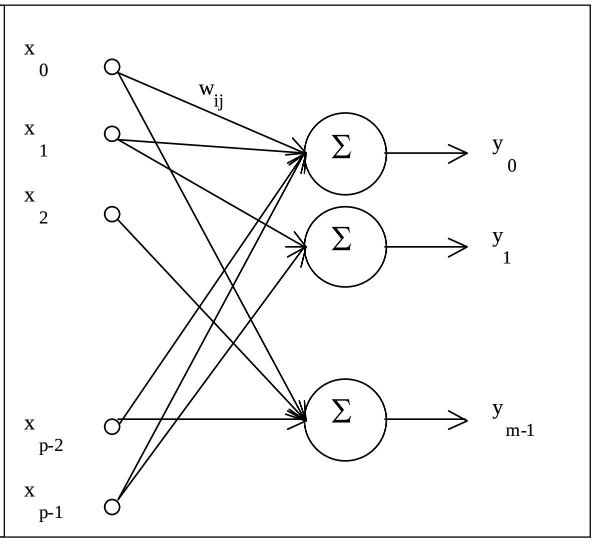


Figure 2. Detail in PCA network

The back-propagation learning algorithm is based on the error-correction principle, i.e. it is necessary to know the network response to the input pattern. The learning process is as follows: On the input are given the input data and then are calculating the response of the network (feed-forward calculation). The error e_i between an actual and a desired output is acquiring by formula (4)

$$e_i(n) = d_i(n) - y_i(n) \tag{4}$$

where d_i is desired output and y_i is the actual output. During the back-propagation are compute the local gradient δ_i for the preceding layers by propagating the errors backwards. Update the weights using formula (5)

$$w_{ij}(n+1) = w_{ij}(n) + \eta\delta_i(n)x_j(n) \tag{5}$$

where w_{ij} is the weight between ith neuron from last layer and jth neuron of the next layer, η is the learning rate. This process is repeated for the next input-output pattern until the error in the output layer is below a prespecified threshold or a maximum number of iterations are reached. We used the minimization of the average squared error cost function E_{avg} given by (6)

$$E_{avg} = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{2}\sum_{j=1}^{V}(d_j(n) - y_j(n))^2 \tag{6}$$

where N is number of the input-output patterns and V is the number of output neurons.
The neural network in that case uses the normalized data from ultrasound range finder as inputs. There are distances d_i, ranging from 20 to 250 cm, to the all objects in the front robot space from 0° to 180°. From input layer of the network we obtained information about free segments Vi. Each of the output neurons "represent" particular segment of the workspace as is depicted on the Fig. 3.

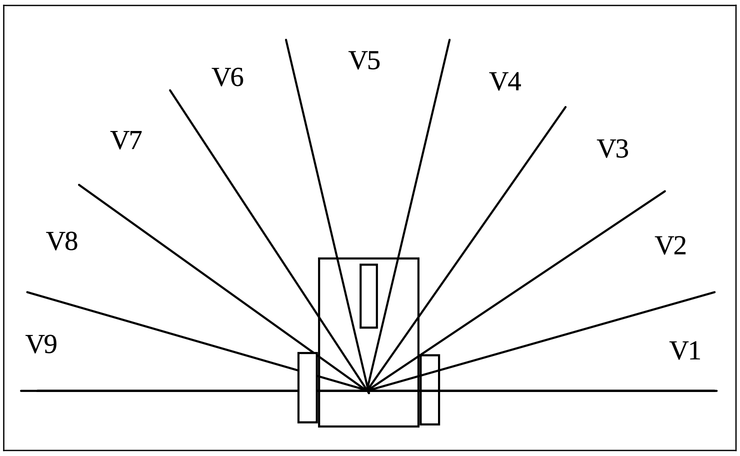


Figure 3. The workspace segments

4.2 Solving the FindPath Problem

For the solving of the findpath problem we use neural network, too. Here we used as a neural network a multilayer perceptron (MLP). Multilayer perceptrons are layered feed-forward networks typically trained with static back-propagation. These networks have found their way into countless applications requiring static pattern classification. Their main advantage is that they are easy to use, and that they can approximate any input/output map. The key disadvantages are that they train slowly, and require lots of training data. Aim of this network is determining of the robot azimuth θ_k for the next robot motion step from the output of first network and from the goal coordinates. The topology of this network is depicted on the Figure 4.

The network contains a three layer – input, hidden and output. This is a layered feed-forward network typically trained with static back-propagation. Here are updating the synapse weights between neurons and threshold. The process is similar to above described for the second phase of the learning process for findspace problem.

On the input of this neural network in our case we give the known free space segment V_i as the output of the first neural network and the goal segments S_i in which the coordinates of the robot goal position should be situated. The choice of the goal segments is the same as is depicted on the Fig. 3. From output layer of this neural network we obtained information O_j about robot motion direction (azimuth) in the next step.

This information is given to the control unit. It manages this information into the robot command for the robot motor control.

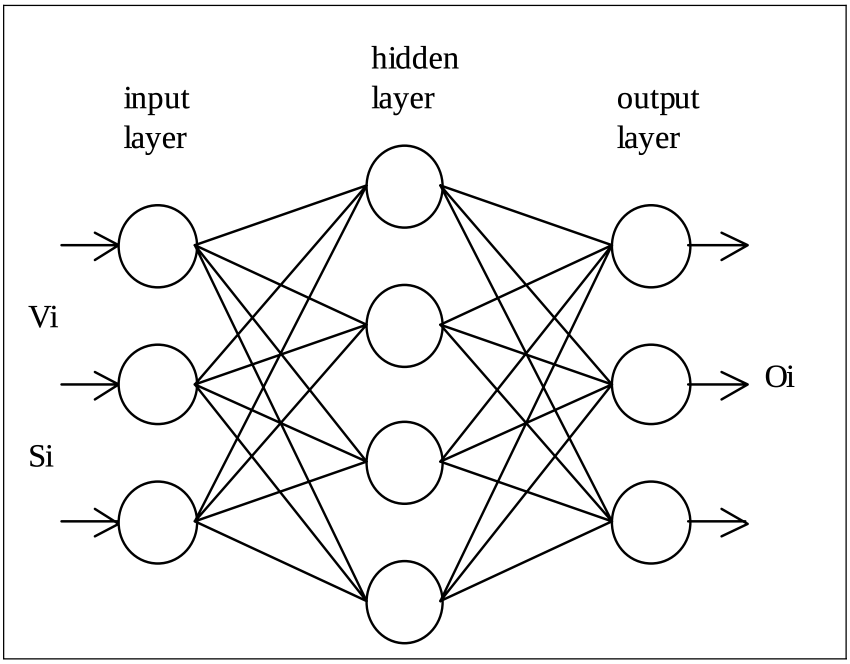


Figure 4. Topology of MLP network

4.3 Realization of the Proposed Algorithm

The proposed algorithm was prepared as a program Neuro in Microsoft Visual C++ with operating system Windows. For the learning and testing of the network we used the programs from NeuroSolutions packages (Neurodimension 2000). The program Neuro will secure the workspace visualization and robot motion simulation.

The learning of all neural networks was realized off-line with scanning data in the work environment. We use a more type of the learning environment. As a basic environment for "learning your environment" was used environment that it is depicted in Fig. 5.

This environment was chosen so that it contains various situations, which can occur during the robot motion. As a testing environment was used the bit map of the laboratory room environment as it is depicted at the Fig. 6. These environments were scanning with ultrasound range finder. The model of the range finder and sensing of its data was implemented as follows: virtual environment with obstacles was represented as a bit map.

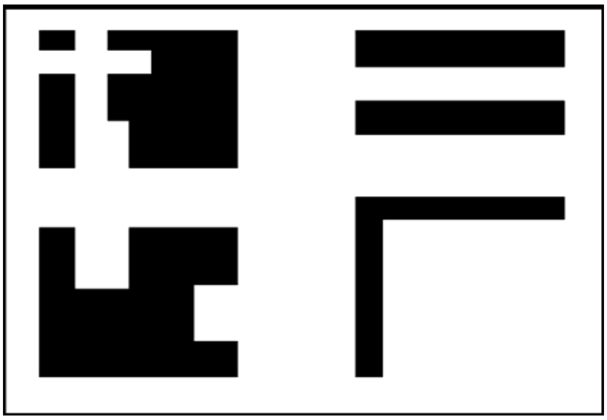


Figure 5. The learning environment

The range finder scanned this environment by emitting beams (see Fig. 6). If the beam in competent direction collided with obstacles (the bit value 0 – black colour) we calculate this distance d_i. The calculated and normalized distances d_i was used as inputs to the neural networks (see Fig.1). Scanning of the environment was in 29 directions. This number was obtaining from many experiments and simulations. At these 29 directions obtained distances d_i create input patterns to the PCA network. The outputs from this network are free environment segments as are depicted on Fig. 3.

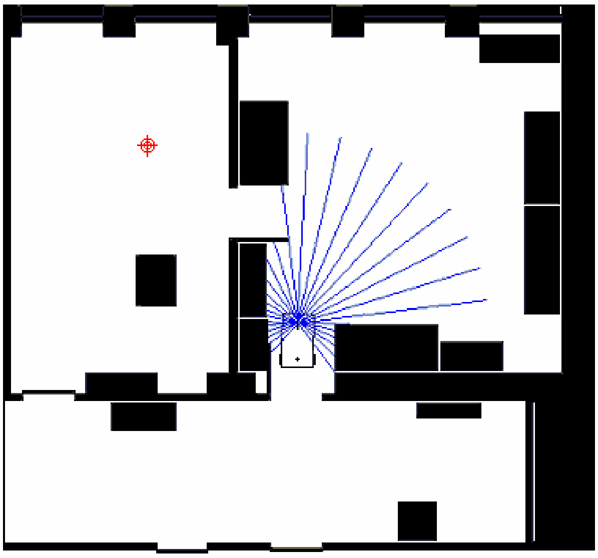


Figure 6. The scanning range finder

For the learning of the free segments was create the safe sensor map. It is the set of minimal distances from obstacles for actual state of the scanning scene using the really dimensions of the robot. At the robot motion (and scanning) in learning environment we compare values d_i with the safe sensor map at the each step and we determine the free segments. The obtained information d_i (29 values) and information about free segments V_i (9 values) was in each step saved to the file. This file forms the training set of pattern for PCA network in second phase.

The second neural network - a feed-forward network - is typically trained with static back-propagation algorithm. For the learning of this supervised MLP network was using the combination status table. This table contains all potential combinations of the space segments V_i and the goal segments S_j. The parameters V_i and S_j have two values in the table. If $V_i = 1$ then space segment V_i not containing the obstacle meaning the motion in this segment is possible. If its value is 0 the segment is occupied and the motion within it is not possible. The value $S_j=1$ says that in this goal segment the goal coordinates take place and the value 0 signalises the absence of the goal coordinates. The supervisor attaches the required output (as the robot's reaction) to any combination of V_i and S_j, and the network learns these situations. The robot reaction (azimuth) the supervisor chooses from free V_i and S_j so as the robot motion was realized in the safe direction to the goal. Our network had 18 input neurons (9 value V_i and 9 value S_j), 20 hidden neurons and 9 output neurons (azimuth O_j). Output O_j of this second network is given to the control unit. It manages this information into the robot command for the robot motor control.

In our case experimental robot distinguish five motion commands: forward, turn left, left, turn right, right. The commands turn left (right) means turning about 45° in left (right) direction. The commands left (right) means turning about 90° in left (right) direction.

When we testing the functionality of proposed algorithm in the laboratory we find out the critical location in which the robot do not know continue in correctly direction. There was a door in the laboratory room or the narrow location in corridor. Therefore we add two neural networks (network H and network D) the multilayer perceptron type. The aim of the network H is recognize when the robot is situated in "hazard" – it stay in narrow location. When this situation is finding the network D execute the safe motion through this narrow location (see Fig. 7).

The network H operates as a switch, which decides about the using the outputs from network MLP (described in section 4. 2.) or from network D for robot motion.

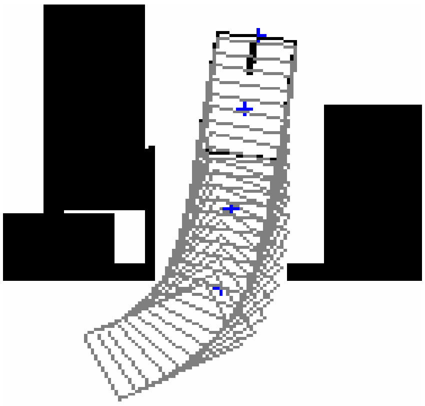


Figure 7. Motion by the door

5. Simulation Results

In our laboratory we have experimental mobile robot AURO, see Fig. 8. It is built up as a prism platform with three-wheeled configuration, which has a length of 850 mm, a width of 500 mm, and a height of 750 mm. It consists of single steerable drive wheel at the front and two passive rear wheels. Two stepper motors are used for driving and steering the front wheel. It has a capability of motion in longitudinal directions and rotation around the robot's reference point and it can reach a maximum speed of 0.1 m/sec.

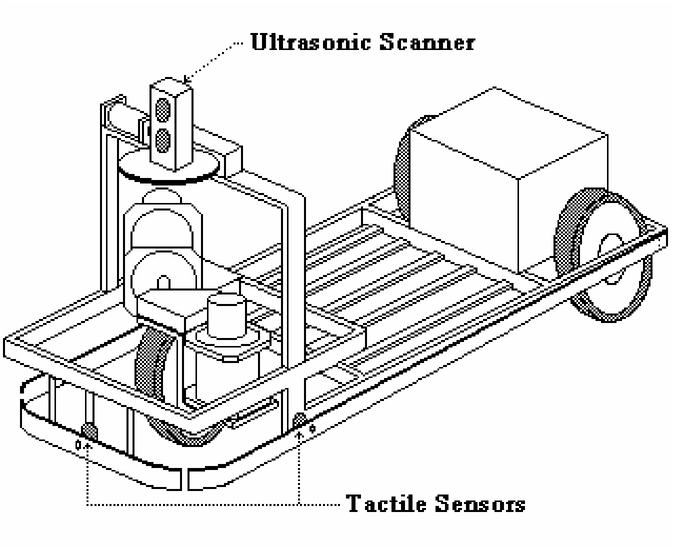


Figure 8. Experimental mobile robot

The drive wheel as well as the passive wheels is equipped with shaft encoders used for odometry measurement. For sensing the environment it has ultrasonic scanning range finder, rotating from 0° to 360° and three tactile sensors on the bumper.

The robot should travel from its initial position to a final desired position across a two-dimensional structured environment. The robot obtains range images by an ultrasound scanning range finder. The ranges for desired angular sector are obtained in N steps, covering up to 180° arcs in front of the robot, are measured by scanning every 200 ms by time-of-flight principle (Uher & Kello, 1999). The ultrasound scanning range finder is disposed on the robot to get the distance measure d_i in the vicinity of the robot where 20 cm < d_i < 250 cm. The main navigation level computation is performed on a host PC via RS 232 communication. The robot manoeuvres are controlled by delivering information of rotation velocity and heading angle of the front wheel.

We have implemented the algorithm described in the above sections in a path planner program Neuro written down in the language C++ on PC Intel Pentium 350 MHz.

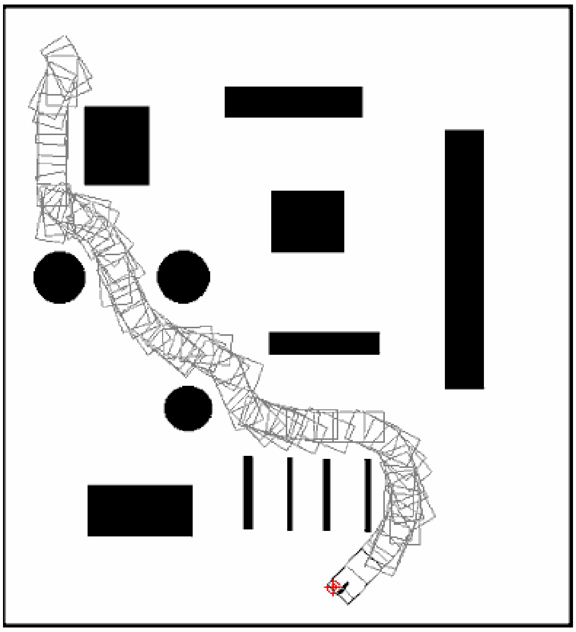


Figure 9. Robot path in unknown environment

For the learning and testing of the network we used the programs from NeuroSolutions packages (Neurodimension 2000). The programs were interconnected with help of the dynamic data change (DDE) in order to enable using the data from program packages NeuroSolutions in the program Neuro.

Several examples were used to test our algorithm at first for a point robot. These first results were presented in (Janglova, 2000).

The functionality of the proposed algorithm was tested in a few type of the workspace. First tests were in the learning environment. Here the robot avoids to all obstacles and it executes each path from the giving start point to the goal point safely. Next testing examples were doing in the environment that was not use for learning of network, i.e. the unknown environment for the robot. At the Fig. 9 is shown this situation – the execute path is collision-free. Then we use this algorithm for simulation of the motion for our experimental mobile platform in the laboratory environment. The obtained results are shown in the Fig. 10 and Fig. 11.

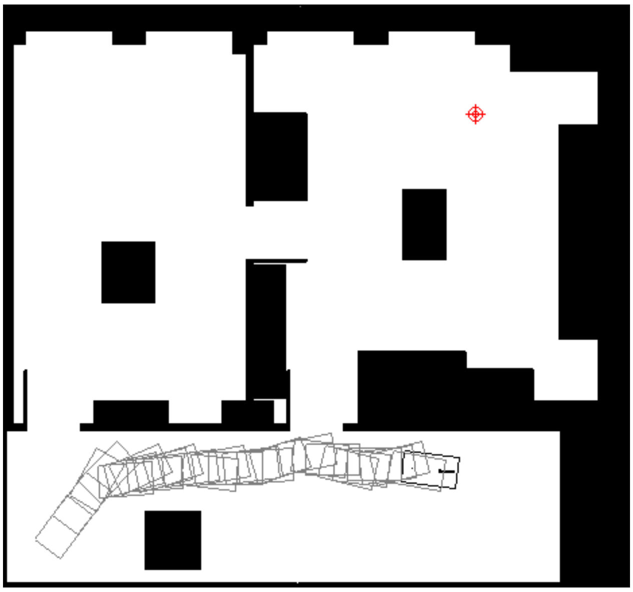


Figure 10. Simulation of robot motion

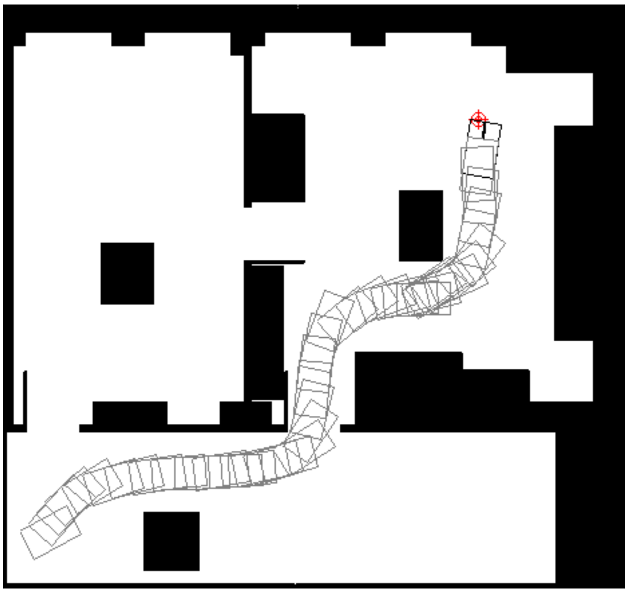


Figure 11. Robot motion from corridor to the room

In these figures are given the bit map of the laboratory environment and the robot paths, which was generated by the above designed algorithm.

The Fig. 10 shows simulation of the robot path when robot task had moving from the left corner of corridor to the goal marked by cross at the laboratory room. The robot does not reach the goal position – it was not pass through the door.This same simulation example with using the adding neural network D and H is shown on the Fig. 11. It is seen that the robot path is collision-free and safe from the start to the goal position.

From the shown examples we conclude that this strategy is usable in general for motion of the robot in arbitrary environment.

6. Conclusion

The chapter presents our results that we obtained making use of the proposed path planning algorithm working with the neural network and sensor data. The simulation examples of the generation of the collision-free path for point robot and for two-dimensional robot show that designed strategy are acceptable for solution of this problem. We played the role of the supervisor to learn the robot to make it's way intelligently toward its target and to avoid obstacles.

In future we will implement this technique for safe motion of our experimental mobile vehicle in indoor conditions. We suppose to use this algorithm not only for the robot motion in known environment but for unknown one, as well. It is necessary to test different parameters in neural network with the aim of reaching the optimal time for finding the (shortest possible) safe path. As the robot collects environment data currently along its path it can avoid not only the static obstacles but also the dynamic ones. We feel that this technique will be suitable also for the motion of mobile devices in complex environment comprising also mobile obstacles.

7. References

Bekey, G. A. & Goldberg, K.Y. (1993). *Neural Networks in Robotics*. Kluwer Academic Publishers, ISBN 0-7923-9268-X, Boston

Brady, M. & et al. (1982). *Robot Motion: Planning and Control.* The MIT Press, ISBN 0-262-02182-X, Cambridge

Domany, E.; Hemmen, J.L. & Schulten, K. (1991). *Models of Neural Networks.* Springer Verlag , ISBN 3-540-51109-1, Berlin

Chohra, A.; Sif, F. & Talaoubrid, S. (1995). Neural Navigation Approach of an Autonomous Mobile Robot in a Partially Structured Environment. *Proceedings of IAV'95,* pp. 238-243, Espoo, June 12-14, Finland

Hebert, M.H.; Thorpe, Ch. & Stentz, A. (1997). *Intelligent Unamnned Grou nd Vehicles.* Kluwer Academic Publishers, ISBN 0-7923-9833-5, Boston

Jain, A.K.; Mao, K. & Mohiuddin, K.M. (1996). Artificial Neural Networks: A Tutorial. *Computer 29,* No. 3, pp. 31-44, ISSN 0018-9162

Janglova, D. (2000). Collision-free Motion Using Neural Networks Approach. *Proceedings of RAAD 2000,* ISBN 86-435-0324-X, pp. 29-34, Maribor, June 1-3, Slovenia

Kim, C.N & Trivedi, M.M. (1998). A Neuro-Fuzzy Controller for Mobile Robot Navigation and Multirobot Convoying. *IEEE Trans.on Systems, Man and Cybernetics-Part B: Cybernetics*, Vol. 28, No. 6, pp. 829-840, ISSN 1083-4419

Latombe, J.C. (1991). *Robot Motion Planning.* Kluwer, ISBN 0-7923-9129-2, Boston

NeuroDimension (2000) Inc. Copyright (c) 1994-1997 in www.nd.com

Tani.J (1996). Model-based Learning for Mobile Robot Navigation from the Dynamical Systems Perspective. *IEEE Trans. on Syst., Man and Cyb.,* Vol. 26, No.3, pp. 421-436, ISSN 1083-4419

Povazan, I.; Janglova, D. & Uher, L. (1995). Odometry in Determination of the Position of an Autonomous Mobile Vehicle. *Proceedings of ISMCR'95,* pp. 425-429, ISBN 80-227-0760-0, Smolenice, June 12-16, Slovak Republic

Uher, L. & Kello, I. (1999). Ultrasound Scanning Range Finder. *Proceedings of the 2nd Int. Conference. Measurement'99,* pp. 232-235, ISBN 80-967402-4-5, Smolenice, April 26-29, Slovak Republic

Voros, J. (2001). Low-cost implementation of distance maps for path planning using matrix guadtrees and octrees. *Robotics and Computer Integrated Manufacturing* Vol. 17, No. 6, pp. 447-459, ISSN 0736-5845

Wang, J. (1998). Primal and Dual Neural Networks for Shortest-Path Planning. *IEEE Trans. on Systems, Man and Cybernetics-Part A: Systems and Humans,* Vol. 28, No. 6, pp. 864-869, ISSN 1083-4419

Generating Timed Trajectories for Autonomous Robotic Platforms: A Non-Linear Dynamical Systems Approach

Cristina Manuela Peixoto dos Santos

1. Introduction

Over the last years, there have been considerable efforts to enable robots to perform autonomous tasks in the unpredictable environments that characterize many potential applications. An autonomous system must exhibit flexible behaviour that includes multiple qualitatively different types of actions and conforms to multiple constraints.

Classically, task planning, path planning and trajectory control are addressed separately in autonomous robots. This separation between planning and control implies that space and time constraints on robot motion must be known before hand with the high degree of precision typically required for non-autonomous robot operation, making it very difficult to work in unknown or natural environments. Moreover, such systems remain inflexible, cannot correct plans online, and thus fail both in non-static environments such as those in which robots interact with humans, and in dynamic tasks or time-varying environments which are not highly controlled and may change overtime, such as those involving interception, impact or compliance. The overall result is a robot system with lack of responsiveness and limited real-time capabilities. A reasonable requirement is that robust behaviour must be generated in face of uncertain sensors, a dynamically changing environment, where there is a continuous online coupling to sensory information. This requirement is especially important with the advent of humanoid robots, which consist of bodies with a high number of degrees-of-freedom (DOFs[1]).

Behavior-based approaches to autonomous robotics were developed to produce timely robotic responses in dynamic and non-engineered worlds in which linkage between perception and action is attempted at low levels of sensory information (Arkin, 1998). In (Khatib, 1986), some of this planning is made “on-line” in face of the varying sensorial information.

Most current demonstrations of behavior-based robotics do not address timing: The time when a particular action is initiated and terminated is not a controlled variable, and is not stabilized against perturbations. When a vehicle, for instance, takes longer to arrive at a goal because it needed to circumnavigate an obstacle, this change of timing is not compensated for by accelerating the vehicle along its path. Timed actions, by contrast, involve stable temporal relationships. Stable timing is important when particular events must be achieved in time-varying environments such as hitting or catching moving

[1] Degree-of-freedom (DOF) is the number of dimensions (variables) required to define the state of the system.

objects, avoiding moving obstacles, or coordinating multiple robots. Moreover, timing is critical in tasks involving sequentially structured actions, in which subsequent actions must be initiated only once previous actions have terminated or reached a particular phase.

This chapter addresses the problem of generating timed trajectories and sequences of movements for robotic manipulators and autonomous vehicles when relatively low-level, noisy sensorial information is used to initiate and steer action. The developed architectures are fully formulated in terms of nonlinear dynamical systems which lead to a flexible timed behaviour stably adapted to changing online sensory information. The generated trajectories have controlled and stable timing (limit cycle type solutions). Incoupling of sensory information enables sensor driven initiation and termination of movement.

Specifically, we address each of the following questions: (a) Is this approach sufficiently versatile such that a whole variety of richer forms of behaviour, including both rhythmic and discrete tasks, can be generated through limit cycle attractors? (b) Can the generated timed trajectories be compatible with the requirement of online coupling to noisy sensorial information? (c) Is it possible to flexibly generate timed trajectories comprising sequence generation and stably and robust implement them both in robot arms and in vehicles with modest computational resources? Flexibility means here that if the sensorial context changes such that the previously generated sequence is no longer appropriated a new sequence of behaviours, adequate to the current situation, emerges. (d) Can the temporal coordination between different end-effectors be applied to the robotics domain such that a tendency to synchronize among two robot arms is achieved? Can the dynamical systems approach provide a theoretically based way of tuning the movement parameters? (e) Can the proposed timing architecture be integrated with other dynamical architectures which do not explicitly parameterize timing requirements?

These questions are answered in positive and shown in a wide variety of experiments. We illustrate two situations in exemplary simulations. In one, a simple robot arm intercepts a moving object and returns to a reference position thereafter. A second simulation illustrates two PUMA arms perform straight line motion in the 3D Cartesian space such that temporal coordination of the two arms is achieved. As an implementation of the approach, the capacity of a low level vehicle to navigate in a non-structured environment while being capable of reaching a target in an approximately constant time is chosen. The evaluation results illustrate the stability and flexibility properties of the timing architecture as well as the robustness of the decision-making mechanism implemented.

This chapter will give a review of the state of the art of modelling control systems with nonlinear dynamic systems, with a focus on arm and mobile robots. Comments on the relationship between this work, similar approaches and more traditional control methods will be presented and the contributions of this chapter are highlighted. It will provide an overview of the theoretical concepts required to extend the use of nonlinear dynamical systems to temporally discrete movements and discuss theoretical as well as practical advantages and limitations.

2. Background and Related Work

The state of the art described in this chapter addresses the work of the most relevant peers developing research in biologically motivated approaches for achieving movement generation and also addresses a reasonable number of demonstrations in the robotic domain which use dynamic systems for movement generation.

In this chapter, we describe a dynamical system architecture to autonomously generate timed trajectories and sequences of movements as attractor solutions of dynamic systems. The proposed approach is inspired by analogies with nervous systems, in particular, by the way rhythmic and discrete movement patterns are generated in vertebrate animals (Beer et al., 1990; Clark et al., 2000). The vertebrate motor system has only little changed during evolution despite the large variety of different morphologies and types of locomotion. These regularities or invariants seem to indicate some fundamental organizational principles in the central nervous systems (Schaal, 2000). For instance, the basic movements of animals, such as walking, swimming, breathing, and feeding consist of reproducible and representative movements of several physical parts of the body which are influenced by the rhythmic pattern produced in the nervous system. Further, in the presence of a variable environment, animals show adaptive behaviours which require coordination of the rhythms of all physical parts involved, which is important to achieve smooth locomotion. Thus, the environmental changes adjust the dynamics of locomotion pattern generation.

The timing of rhythmic activities in nervous systems is typically based on the autonomous generation of rhythms in specialized neural networks located in the spinal cord, called "central pattern generators" (CPGs). Electrical stimulation of the brain stem of decerebrated animals have shown that CPGs require only very simple signals in order to induce locomotion and even changes of gait patterns (Shik and Orlosky, 1966). The dynamic approach offers concepts with which this timing problem can be addressed. It provides the theoretical concepts to integrate in a single model a theory of movement initiation, of trajectory generation over time and also provides for their control. These ideas have been formulated and tested as models of biological motor control in (Schoner, 1994) by mathematically describing CPGs as nonlinear dynamical systems with stable limit cycle (periodic) solutions. Coordination among limbs can be modelled through mutual coupling of such nonlinear oscillators (Schoner & Kelso, 1988). Coupling oscillators to generate multiple phase-locked oscillation patterns has since long time being used to mathematically model animal behaviour in locomotion (Collins, Richmond, 1994), and to formulate mathematical models of adaptation to periodic perturbation in quadruped locomotion (Ito et al, 1998). This framework is also ideal to achieve locomotion by creating systems that autonomously bifurcate to the different types of gait patterns.

The on-line linkage to sensory information can be understood through the coupling of these oscillators to time-varying sensory information (Schoner, 1994). Limited attempts to extend these theoretical ideas to temporally discrete movements (e.g., reaching) have been made (Schoner, 1990). In this chapter, these ideas are further extended to the autonomous generation of discrete movement patterns (Santos, 2003).

This timing problem is also addressable at the robotics domain. While time schedules can be developed within classical approaches (e.g., through configuration-time space representations), timing is more difficult to control when it must be compatible with continuous on-line coupling to low level and often noisy sensory information which is used to initiate and steer action. One type of solution is to generate time structure at the level of control.

In the Dynamical Systems approach to autonomous robotics (Schoner & Dose, 1992; Steinhage & Schoner, 1998, Large et al, 1999, Bicho et al, 2000), plans are generated from stable states of nonlinear dynamical systems, into which sensory information is fed. Intelligent choice of planning variables makes it possible to obtain complex trajectories and action sequences from stationary stable states, which shift and may even go through instabilities as sensory information changes. Herein, an extension of this approach is presented to the timing of motor acts, and an attractor based two-layer dynamics is proposed that autonomously generates timed movement and sequences (Schoner & Santos, 2001). This work is further extended to achieve temporal coordination among two DOFs as described in (Santos, 2003). This coordination is achieved by coupling the dynamics of each DOF.

The idea of using dynamic systems for movement generation is not new and recent work in the dynamic systems approach in psychology has emphasized the usefulness of autonomous nonlinear differential equations to describe movement behaviour. In (Raibert, 1986), for instance, rhythmic action is generated by inserting into dynamic control model terms that stabilized oscillatory solutions. Similarly, (Schaal & Atkeson, 1993) generated rhythmic movements in a robot arm that supported juggling of a ball by inserting into the control system a model of the bouncing ball together with terms that stabilized stable limit cycles. Earlier, (Buhler et al., 1994) obtained juggling in a simple manipulator by inserting into the control laws terms that endowed the complete system with a limit cycle attractor. (Clark et al., 2000) describes a nonlinear oscillator scheme to control autonomous mobile robots which coordinates a sequence of basic behaviours in the robot to produce the higher behaviour of foraging for light. (Williamson, 1998) exploits the properties of a simple oscillator circuit to obtain robust rhythmic robot motion control in a wide variety of tasks. More generally, the nonlinear control approach to locomotion pioneered by (Raibert, 1986) amounts to using limit cycle attractors that emerge from the coupling of a nonlinear dynamical control system with the physical environment of the robot. A limitation of such approaches is that they essentially generate a single motor act in rhythmic fashion, and remain limited with respect to the integration of multiple constraints, and planning was not performed in the fuller sense. The flexible activation of different motor acts in response to user demands or sensed environmental conditions is more difficult to achieve from the control level. However, (Schaal & Sternad, 2000) has been able to generate temporally discrete movement as well.

However, there are very few implementations of oscillators for arm and vehicles control. The work presented in this chapter extends the use of oscillators to tasks both on an arm and on a wheeled vehicle. It also differs from most of the literature in that it is implemented on a real robot.

In the field of robotics, the proposed approach holds the potential to become a much more powerful strategy for generating complex movement behavior for systems with several DOFs than classical approaches. The inherent autonomy of the applied approach helps to synchronize systems and thus reduces the computational requirements for generating coordinated movement. This type of control scheme has a great potential for generating robust locomotion and movement controllers for robots. The work proposed is novel because it significantly facilitates movement generation and sequences of movements.

Finally, the approach shows up several appealing properties, such as perception-action coupling and reusability of the primitives. The technical motivation is that this framework finds a great number of applications in service tasks (e.g. replacement of humans in industrial tasks and unsafe areas, collaborative work with a human/robot operator) and will permit to advance towards better rehabilitation of movement in amputees (e.g. intelligent and more human like prostheses). In summary, it is expected that in the long run one can potentially increase the application of autonomous robots in tasks for helping the common citizen.

3. The Dynamical Systems Trajectory Generator

In this section, we develop an approach to generate rhythmic and discrete movements. This work is innovative in the manner how it formalizes and uses movement primitives, both in the context of biological and robotics research. We apply autonomous differential equations to model the manner how behaviours related to locomotion are programmed in the oscillatory feedback systems of "central pattern generators" in the nervous systems (Schoner, 1994).

The desired movement is to start at time t_{init} in an initial postural state[2], $x_{taskinit}$, and move to a new final postural state, $x_{taskfinal}$, within a desired movement time and keeping that time stable under variable conditions. Such behavior is what we consider timed discrete movement. Figure 1 illustrates such movement along the X-axis: At time t_{init}, the system begins its timed movement between $x_{taskinit}$, and $x_{taskfinal}$. After a certain movement time, denominated *MT*, the movement stops and the system remains at the $x_{\text{taskfinal}}$ position.

In order to understand the concept of discrete movement, consider a two dof robot arm moving in a plane from an initial rest position to a final folded one. The mapping between the robot's movement and the discrete movement is accomplished through simple coordinate transformations in which the value of the timing variable x is updated.

The initial postural state of discrete movement is mapped onto the initial rest position of the arm. The x oscillatory movement during movement time, *MT*, corresponds to the arm moving from the initial rest position to the folded one. The final postural position of the discrete movement corresponds to the arm in its final folded position (see Figure 1).

We use the dynamic systems approach to model the desired behaviour, discrete movement, as a time course of the behavioural variables. These variables are generated by dynamical systems.

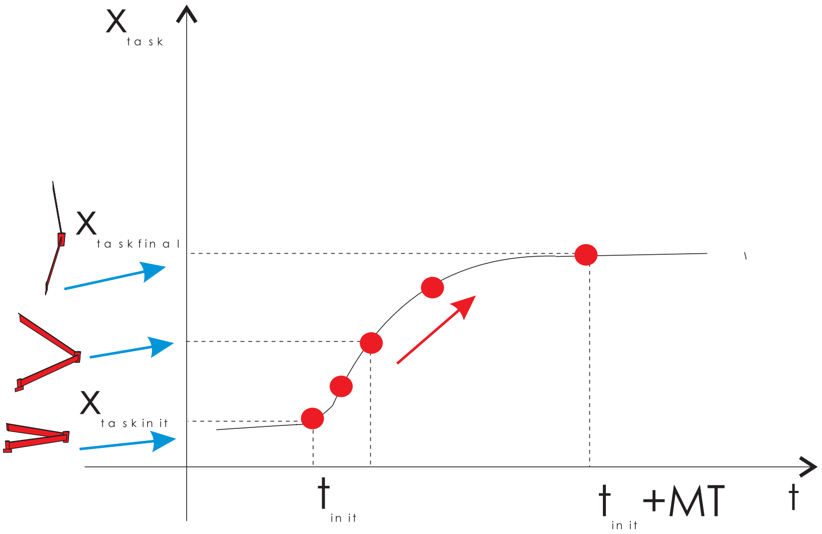


Figure 1. A discrete movement along the ˆX –axis (as indicated by the black arrow). Mapping of the timing variable x onto the movement of a two dof robot arm

The state of the movement is represented by a single variable, x, which is not directly related to the spatial position of the corresponding effector, but rather represents the effector temporal position within the discrete movement cycle. This temporal position x is then converted by simple coordinate transformations into the correspondent spatial position (Figure 1). Generating oscillatory solutions requires at least two dynamical dof.

[2] A postural state is a stationary state in which there is no movement.

Thus, although only the variable *x* will be used to control motion of a relevant robotic task variable, a second auxiliary variable, *y*, is needed to enable the system to undergo periodic motion. We could have select $\dot{x}$ (2*nd* order), but instead we select *y* which provides simplicity to equations that can be easily solved analytically.

The entire discrete trajectory is sub-divided into three different behaviours or task constraints: an initial postural state, a stable periodic movement and a final postural state. The implied instability in the switch between these states does not allow to use local bifurcation theory and is difficult to build a general dynamical model which generates this movement as a stable solution (Schoner,1990). The adopted solution is to expresse each constraint as behavioural information (Schoner & Dose, 1992) captured as individual contributions to the dynamical system. Therefore, the trajectories are generated as stable solutions of the following dynamical system, which consists on the combined addition of the contributions of the individual task constraints,

$$\begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = |u_{\text{init}}| f_{\text{init}} + |u_{\text{hopf}}| f_{\text{hopf}} + |u_{\text{final}}| f_{\text{final}} + \text{gwn} \tag{1}$$

that can operate in three dynamic regimes controlled by the three "neurons" u_i (i = init; hopf; final). These "neurons" can go "on" (= 1) or "off" (=0), and are also governed by dynamical systems, described later on. The f_{init} and f_{final} contributions are described by dynamical systems whose solutions are stable fixed point attractors (postural states), and the f_{hopf} contribution generates a limit cycle attractor solution. The timing dynamics are augmented by a Gaussian white noise term, *gwn*, that guarantees escape from unstable states and assures robustness to the system.

Postural contributions are modelled as attractors of the behavioral dynamics

$$\begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = -\alpha_{\text{post}} \begin{pmatrix} x_i - x_{\text{post}} \\ y_i \end{pmatrix} \tag{2}$$

where x_i is the current *x* position of the *i* movement, x_{post} is the *x* postural position and y_i is the *y* postural position of the *i* movement (set to zero since it is an auxiliary variable required to enable the system to undergo a periodic motion). These states are characterized by a time scale of $\tau_{\text{post}} = \frac{1}{\alpha_{\text{post}}} = 0.2.$

The Hopf contribution generates the limit cycle solution: the periodic stable movement. We use a well-known mathematical expression, a normal form of the Hopf bifurcation (Perko, 1991), to model the oscillatory movement between two *x* values:

$$\begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = \begin{pmatrix} \alpha_h & -\omega \\ \omega & \alpha_h \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} - \gamma(x^2 + y^2) \begin{pmatrix} x \\ y \end{pmatrix}$$

This simple polynomial equation contains a bifurcation from a fixed point to a limit cycle. We use it because it can be completely solved analytically, providing complete control over its stable states. The limit cycle solution is a periodic oscillation with cycle time $T = \frac{2\pi}{\omega}$ and finite amplitude, $A = 2\sqrt{\frac{\alpha h}{\lambda}}$. Relaxation to this stable solution occurs at a time scale of: $\tau_{osc} = \frac{1}{2\alpha_h}$. Further details regarding the Hopf normal form can be found in (Perko, 1991).

An advantage of our specific formulation is the fact that our system is analytically treatable to a large extent, which facilitates the specification of parameters such as movement time, movement extent, or maximal velocity. This analytical specification is also an innovative aspect of our work.

1.1 Neural Dynamics

The "neuronal" dynamics u_i (*i = init; final; hopf*) switches the timing dynamics from the fixed point regimes into the oscillatory regime and back. Thus, a single discrete movement act is generated by starting out with neuron $|u_{\text{init}}| = 1$ activated, the other neurons deactivated ($|u_{\text{hopf}}| = |u_{\text{final}}| = 0$), so that the system is in a postural state. The oscillatory solution is then stabilized ($|u_{\text{init}}| = 0$; $|u_{\text{hopf}}| = 1$). This oscillatory solution is deactivated again when the effector reaches its target state, after approximately a half-cycle of the oscillation, turning on the final postural state instead ($|u_{\text{hopf}}| = 0$; $|u_{\text{final}}| = 1$). These various switches are generated by the following competitive dynamics:

$$\alpha \dot{u}_{\text{init}} = \mu_{\text{init}} u_{\text{init}} - |\mu_{\text{init}}| u_{\text{init}}^3 - v\left(u_{\text{final}}^2 + u_{\text{hopf}}^2\right) u_{\text{init}} + \text{gwn} \tag{3}$$

$$\alpha \dot{u}_{\text{hopf}} = \mu_{\text{hopf}} u_{\text{hopf}} - |\mu_{\text{hopf}}| u_{\text{hopf}}^3 - v\left(u_{\text{init}}^2 + u_{\text{init}}^2\right) u_{\text{hopf}} + \text{gwn} \tag{4}$$

$$\alpha \dot{u}_{\text{final}} = \mu_{\text{final}} u_{\text{final}} - |\mu_{\text{final}}| u_{\text{final}}^3 - v\left(u_{\text{init}}^2 + u_{\text{hopf}}^2\right) u_{\text{final}} + \text{gwn} \tag{5}$$

The first two terms of each equation represent the normal form of a *degenerate pitchfork bifurcation*. A single attractor at $u_{\text{fp}} = 0$ for negative μ_i becomes unstable for positive μ_i, and two new attractors appear at $u_{\text{fp}} = 1$ and $u_{\text{fp}} = -1$. We use the absolute value of u_{i} as a weight factor in the timing dynamics, so that +1 and -1 are equivalent "on" states of a neuron, while $u = 0$ is the "off" state.

The third term in each equation is a competitive term, which destabilizes any attractors in which more than one neuron is "on". The "neurons", u_i, are coupled through the parameter υ, named *competitive interaction*. For positive μ_i, all attractors of this competitive dynamics have one neuron in an "on" state, and the other two neurons in the "off" state.

In the competitive case, the parameter μ_i determines the *competitive advantage* of the correspondent behavioral variable. That is, among the u_{i} variables, the one with the largest μ_i wins ($u_{\text{i}} = 1$) and is turned on, while the others competing variables are turned off ($u_{\text{i}} = 0$). However, for sufficiently small differences between the different μ_i values multiple outcomes are possible (the system is multistable)(Large et al., 1999).

1.2 Sequential Activation Criteria

We design functional forms for parameters μ_i and υ such that the competitive dynamics appropriately bifurcates to the different types of behaviour in any given situation. These bifurcations happen for certain values of parameters μ_i and υ, which are in turn dependent on the environmental situation itself. As the environmental situation changes, the neuronal parameters reflect by design these changes causing bifurcations in the competitive level. To control switching, the parameters, μ_i (*competitive advantages*) are therefore defined as functions of user commands, sensory events, or internal states (Steinhage & Schoner, 1998). Here, we make sure that one neuron is always "on" by varying the μ_i -parameters between the values 1.5 and 3.5, $\mu_i = 1.5 + 2b_i$,where b_i are "quasi-boolean" factors taking on values between 0 and 1 (with a tendency to have values either close to 0 or close to 1). These "quasi-booleans" express logical or sensory conditions controlling the sequential activation of the different neurons (see (Steinhage & Schoner, 1998; Santos, 2003), for a general framework for sequence generation based on these ideas):

1. b_{init} may be controlled by user input: the command "move" sets b_{init} from the default value 1 to 0 to destabilize the initial posture. In a first instance b_{init} is controlled by an initial time set by the user. Thus, its value changes from 1 to 0 when time exceeds t_{init}:

$$b_{init}(t) = \sigma(t_{init} - t) \tag{6}$$

Herein, σ(.) is a sigmoid function that ranges from 0 for negative argument to 1 for positive argument, selected as

$$\sigma(x) = [\tanh(10x) + 1]/2 \tag{7}$$

although any other functional form will work as well.

b_{init} may also be controlled by sensory input, such that, for instance, b_{init} changes from 1 to 0 when a particular sensory event is detected. Below we demonstrate how the time-to-contact of an approaching object computed from sensory information can be used to initiate movement in this manner.

2. b_{hopf} is set from 0 to 1 under the same conditions. This term is multiplied, however, with a second factor $b_{\text{has not reached target}}(x) = \sigma(x_{\text{crit}} - x)$ that resets b_{hopf} to zero when the effector has reached its final state. The factor, $b_{\text{has not reached target}}(x)$ has values close to one while the timing variable x is below $x_{\text{crit}} = 0.7$ and switches to values close to zero when x comes within 0.3 of the target state ($x = 1$). Multiplying two quasi-booleans means connecting the corresponding logical conditions with an "and" operation. Thus, as soon as the timing variable has come within the vicinity of the final state, it autonomously turns the oscillatory state off. In actual implementation, this switch can be driven from the sensed actual position of an effector rather than from the timing dynamics. The final expression for b_{hopf} is:

$$b_{\text{hopf}} = \sigma(t - t_{\text{init}})b_{\text{has not reached target}}(x) \tag{8}$$

3. b_{final} is, conversely, set from 0 to 1 when the timing variable comes into the vicinity of the target: $b_{final} = 1 - b_{\text{has not reached target}}$.

The time scale of the neuronal dynamics is given by $\tau_u = \dfrac{-1}{\mu - \upsilon}$ and is set to a relaxation time of τ_u=0.02, ten times faster than the relaxation time of the timing variables. This difference in time scale guarantee that the analysis of the attractor structure of the neural dynamics is unaffected by the dependence of its parameters, μ_i on the timing variable, x, which is a dynamical variable as well. Strictly speaking, the neural and timing dynamics are thus mutually coupled. The difference in time scale makes it possible to treat x as a parameter in the neural dynamics (adiabatic variables). Conversely, the neural weights can be assumed to have relaxed to their corresponding fixed points when analyzing the timing dynamics (adiabatic elimination). The adiabatic elimination of fast behavioral variables reduces the complexity of a complicated behavioural system built up by coupling many dynamical systems (Steinhage & Schöner, 1998; Santos, 2003). By using different time scales one can design the several dynamical systems separately.

1.3 An Example: A Timed Temporally Discrete Movement Act

Periodic movement can be trivially generated from the timing and neural dynamics by selecting u_{hopf} "on" through the corresponding quasi-booleans. A timed, but temporally discrete movement act, is autonomously generated by these two coupled levels of nonlinear dynamics through a sequence of neural switches, such that an oscillatory state exists during an appropriate time interval of about a half-cycle. This is illustrated in Figure

2. The timing variable, x, which is used to generate effector movement, is initially in a postural state at -1, the corresponding neuron u_{init} being "on". When the user initiates movement, the quasi-booleans, b_{init} and b_{hopf} exchange values, which leads, after a short delay, to the activation of the "hopf" neuron. This switch initiates movement, with x evolving along a harmonic trajectory, until it approaches the final state at +1. At that point, the quasi-boolean b_{final} goes to one, while b_{hopf} changes to zero. The neurons switch accordingly, activating the final postural state, so that x relaxes to its terminal level x = 1. The movement time is approximately a half cycle time, here MT = 2.

2. Simulation of a Two Dof Arm Intercepting a Ball

As a toy example of how the dynamical systems approach to timing can be put to use to solve robotic problems, consider a two dof robot arm moving in a plane (Figure 3).
The task is to generate a timed movement from an initial posture to intercept an approaching ball. Movement with a fixed movement time (reflecting manipulator constraints) must be initiated in time to reach the ball before it arrives in the plane in which the arm moves. Factors such as reachability and approach path of the ball are continuously monitored, leading to a return of the arm to the resting position when interception becomes impossible (e.g., because the ball hits outside the workspace of the arm, the ball is no longer visible, or ball contact is no longer expected within a criterion time-to-contact). After the ball interception, the arm moves back to its resting position, ready to initiate a new movement whenever appropriate sensory information arrives.

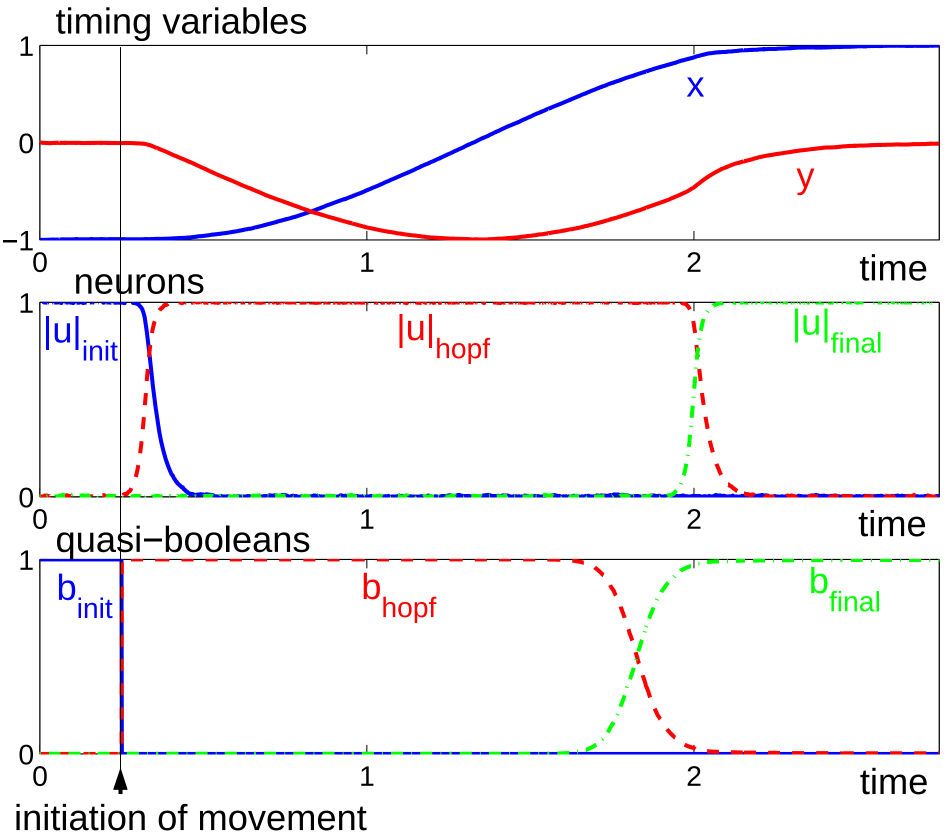


Figure 2. Simulation of a user initiated temporally discrete movement represented by the timing variable, x, which is plotted together with the auxiliary variable, y, in the top panel. The time courses of the three neural activation variables, u_{init}, u_{hopf}, and u_{final}, which control the timing dynamics, are shown in the middle panel. The quasi-boolean parameters, b_{init}, b_{hopf}, and b_{final}, plotted on bottom, determine the competitive advantage of each neuron

In order to formulate this task using the nonlinear dynamical systems approach, three relevant coordinate systems are defined: a) The timing variable coordinate system, *{P}*[3], describes the timing variable position along a straight path from the initial to the final postural position. This is a conceptual frame, in which temporal movement is planned.

b) The task reference coordinate system (universal coordinate system) describes the end-effector position, *(x, y, z)*, of the arm along a straight path from the initial position (initial posture) to the target position (computed coordinates of point of interceptance) and the ball position. c) The base coordinate system *{R}* is attached to the robot's base and the arm kinematics is described by two joint angles in this frame.

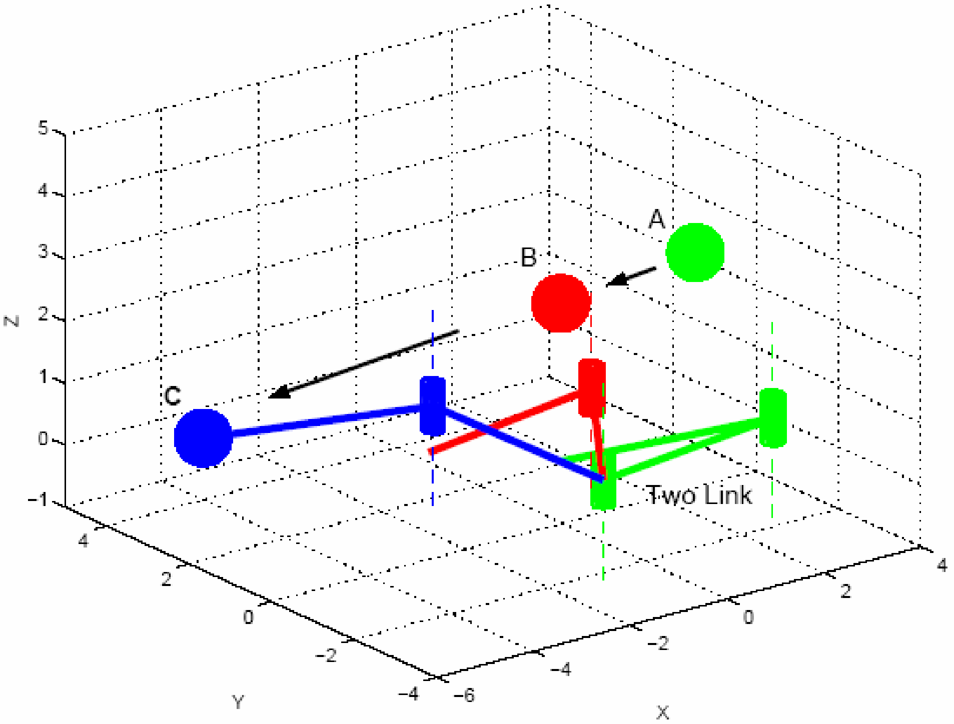


Figure 3. A two dof arm intercepts an approaching ball. Corresponding ball and arm positions are illustrated by using the same grey-scale. The first position (light grey) is close to the critical time-to-contact, where arm motion starts. The last position (dark grey) is close to actual contact. The black arrows indicate the ball's movement

The timing variable coordinate system *{P}* is positioned such that ^{P}x coordinates vary between -1 and +1, with ^{P}y and ^{P}z coordinates equal to zero. ^{P}x is scaled to the desired amplitude, *A*, dependent on the predicted point of interceptance.

Frame (a) and (b) are linked through straightforward formulae, which depend on the predicted point of interceptance, ($x(\tau_{t2c})$; $y(\tau_{t2c})$;) in the task reference frame. Frame (b) and (c) are linked through the kinematic model of the robot arm and its inverse (which is exact).

During movement execution, the timing variables are continuously transformed into task frame (b), from which joint angles are computed through the inverse kinematic transformation. The solutions of the applied autonomous differential equations are converted by simple coordinate transformations, using model-based control theory, into motor commands.

[3] However, in the text, the timing variables ^{P}x and ^{P}y are referred to as timing variable *x*, *y* without superscript.

2.1 Coupling to Sensorial Information

In order to intercept an approaching ball it is necessary to be at the right location at the right time. We use the visual stimulus as the perception channel to our system. In these simulations we have extracted from a simulated ball trajectory two measures: the time-to-contact, τ_{t2c}, and the point-of-contact. The robot arm intersects the ball on the plane of the camera (mounted on its base) such that the ball's movement crosses the observer (or image) plane ($z_B = 0$). The time it takes to the ball to intersect the arm at this point in space, that is, the time-to-contact, is extracted from segmented visual information without having estimated the full cartesian trajectory of the ball (Lee, 1976). We consider the ball has a linear trajectory in the 3D cartesian space with a constant approach constant velocity. The point of contact can be computed along similar lines if the ball size is assumed to be known and can be measured in the image.

To simulate sensor noise (which can be substantial if such optical measures are extracted from image sequences), we added either white or coloured noise to the estimated time-to-contact. Here we show simulations that used coloured noise, ξ, generated from

$$\zeta = -\frac{1}{\tau_{\mathrm{corr}}}\zeta + \sqrt{\mathrm{Q}}\,\mathrm{gwn} \tag{9}$$

where *gwn* is gaussian white noise with zero mean and unit variance, so that $Q = 5$ is the effective variance.

The correlation time, τ_{corr}, was chosen as 0.2 sec. The simulated time-to-contact was thus

$$\tau_{\mathrm{t2c}} = \mathrm{true\ \ time\!-\!to\!-\!contact} + \zeta(\mathrm{t}) \tag{10}$$

2.2 Behavior Specifications

These two measures, time-to-contact and point-of-contact, fully control the neural dynamics through the quasi-boolean parameters. A sequence of neural switches is generated by translating sensory conditions and logical constraints into values for these parameters. For instance, the parameter, b_{init}, controlling the competitive advantage of the initial postural state must be "on" (= 1) when the timing variable x is close to the initial state -1, **and** either of the following is true: a) Ball not approaching or not visible ($\tau_{t2c} \leq 0$). b) Ball contact not yet within a criterion time-to-contact ($\tau_{t2c} > \tau_{crit}$). c) Ball is approaching within criterion time-to-contact but is not reachable ($0 < \tau_{t2c} < \tau_{crit}$; $b_{reachable} = 0$).

These logical conditions can be expressed through this mathematical function:

$$\mathrm{b}_{\mathrm{init}} = \sigma(-\mathrm{x}_{\mathrm{crit}} - \mathrm{x})\left[\sigma(\tau_{\mathrm{t2c}} - \tau_{\mathrm{crit}}) + \sigma(\tau_{\mathrm{t2c}})\sigma(\tau_{\mathrm{crit}} - \tau_{\mathrm{t2c}})\sigma(1 - \mathrm{b}_{\mathrm{reacchable}}) + \sigma(-\tau_{\mathrm{t2c}})\right] \tag{11}$$

where $\sigma(.)$ is the threshold-function used earlier (Equation 7).

The "or" is realized by summing terms which are never simultaneously different from zero. In other cases, the "or" is expressed with the help of the "not" (subtracting from 1) and the "and". This is used in the following expressions for b_{hopf} and b_{final} which can be derived from a similar analysis:

$$\begin{aligned}\mathrm{b}_{\mathrm{hopf}} &= 1 - (1 - \left[\sigma(\mathrm{x}_{\mathrm{crit}} - \mathrm{x})\sigma(\tau_{\mathrm{t2c}})\sigma(\tau_{\mathrm{crit}} - \tau_{\mathrm{t2c}})\sigma(\mathrm{b}_{\mathrm{reacchable}})\right]) \\ &(1 - \left[\sigma(\mathrm{x} + \mathrm{x}_{\mathrm{crit}})\{\sigma(1 - \mathrm{b}_{\mathrm{reacchable}}) + \sigma(-\tau_{\mathrm{t2c}}) + \sigma(\tau_{\mathrm{t2c}} - \tau_{\mathrm{crit}}) + \sigma(\mathrm{x}_{\mathrm{crit}} - \mathrm{x})\ \}\right])\end{aligned} \tag{12}$$

$$\mathrm{b}_{\mathrm{final}} = \sigma(\tau_{\mathrm{t2c}})\sigma(\tau_{\mathrm{crit}} - \tau_{\mathrm{t2c}})\sigma(\mathrm{b}_{\mathrm{reacchable}}) + \sigma(\mathrm{x} - \mathrm{x}_{\mathrm{crit}}) \tag{13}$$

2.3 Properties of the Generated Timed Trajectory

Figure 3 shows how this two dof arm intercepts an approaching ball. The detailed time courses of the relevant variables and parameters are shown in Figure 4. As the ball approaches, the current time-to-contact becomes smaller than a critical value (here 3), at which time the quasi-boolean for motion, b_{hopf} becomes one, triggering activation of the corresponding neuron, u_{hopf}, and movement initiation. Movement is completed (x reaches the final state of +1) well before actual ball contact is made. The arm waits in the target posture. In this simulation the ball is reflected upon contact. The negative time-to-contact observed then leads to autonomous initiation of the backward movement to the arm resting position.

The fact that timed movement is generated from attractor solutions of a nonlinear dynamical system leads to a number of properties of this system, that are potentially useful to real-world implementations of this form of autonomy. The simulation shown in Figure 4 illustrates how the generation of the timing sequence resists against sensor noise: the noisy time-to-contact data led to strongly fluctuating quasi-booleans (noise being amplified by the threshold functions). The neural and timing dynamics, by contrast, are not strongly affected by sensor noise so that the timing sequence is performed as required. When simulations with this level of sensor noise are repeated, failure is never observed, although there are instances of missing the ball at even larger noise levels. By simulating strong sensor noise we demonstrate that the approach is robust. Note how the autonomous sensor-driven initiation of movement is stabilized by the hysteresis properties of the competitive neural dynamics, so that small fluctuations of the input signal back above threshold do not stop the movement once it has been initiated (Schoner & Dose, 1992; Schoner & Santos, 2001).

The design of the quasi-boolean parameters of the competitive dynamics guarantees that flexibility is fulfilled: if the sensorial context changes such that the previously generated sequence is no longer adequate, the plan is changed and a new sequence of events emerges.

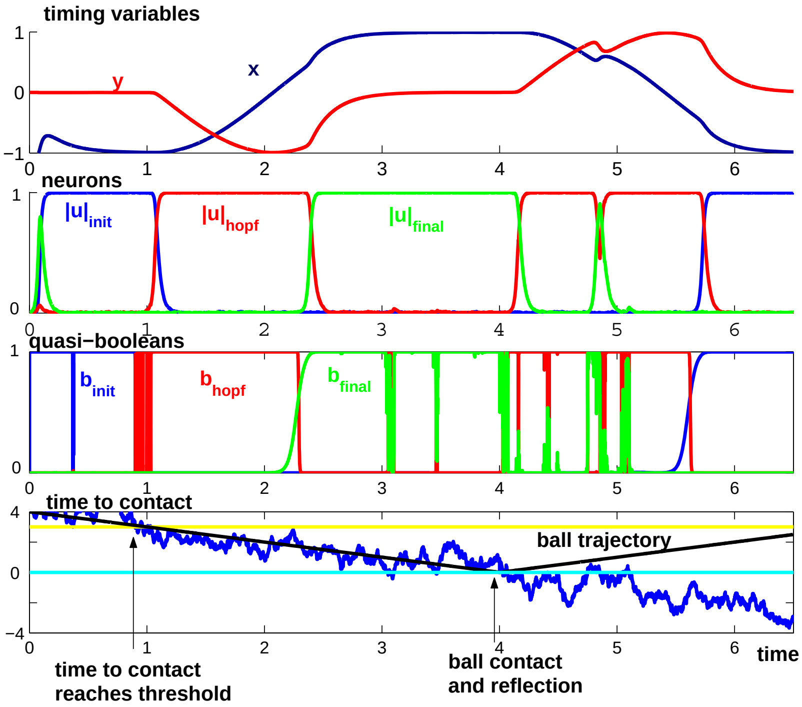


Figure 4. Trajectories of variables and parameters in autonomous ball interception and return to resting position. The top three panels represent timing variables, neural variables and quasi-booleans. The bottom panel shows the time-to-contact, which crosses a threshold at about 0:5 time units. When contact is made, the ball is assumed to be reflected, leading to negative time-to-contact

When sensory conditions change an appropriate new sequence of events emerges. When one of the sensory conditions for ball interception is invalid (e.g., ball becomes invisible, unreachable, or no longer approaches with appropriate time-to-contact), then one of the following happens depending on the point within the sequence of events at which the change occurs: 1) If the change occurs during the initial postural stage, the system stays in that postural state. 2) If the change occurs during the movement, then the system continues on its trajectory, now going around a full cycle to return to the reference posture. 3) When the change occurs during posture in the target position, a discrete movement is initiated that takes the arm back to its resting position.

The decision is dependent on local information available at the system's current position: on the current location of the timing variable, *x*, on the time-to-contact and point-of-contact information currently available. The non-local sequence of events is generated through local information without needing symbolic representations of the behaviours. This is achieved by obeying the principles of the Dynamic Approach and illustrates the power of our approach: the behaviour of the system itself leads to the changing sensor information which controls the change and persistence of a rich set of behaviors.

Consider the ball is suddenly shifted away from the arm at about 1.9 time units, leading to much larger time-to-contact, well beyond threshold for movement initiation. In Figure 5, time-to-contact becomes suddenly larger than the critical value when the arm is in its motion stage: u_{hopf} neuron is activated and the other neurons are deactivated. The u_{hopf} neuron rests activate while the arm continues its movement a full cycle. At the time the x timing variable is captured by the initial postural state (x = -1), the quasi-boolean b_{init} becomes one, triggering the activation of the neuron, u_{init}, and b_{hopf} becomes zero, deactivating the corresponding neuron u_{hopf}. The arm rests in the reference position. This behaviour emerges from the sensory conditions controlling the neuronal dynamics.

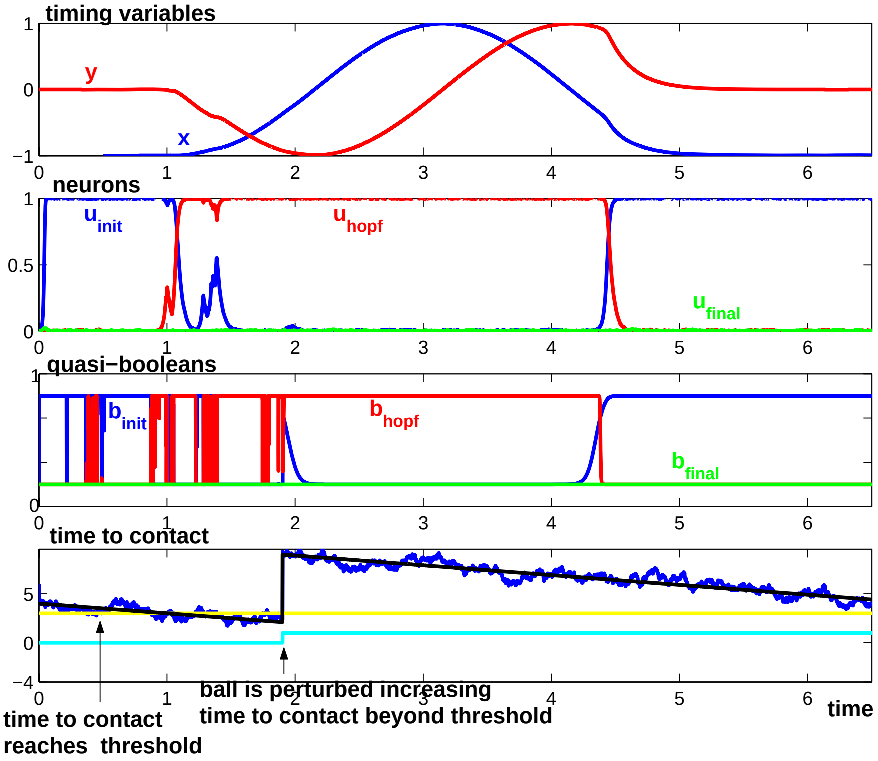


Figure 5. Similar to Figure 4, but the ball is suddenly shifted at about 1.9 time units leading to a time to contact larger than the threshold value (3) required for movement initiation

3. Coupling among Two Timing Systems

Another task illustrating uses of the dynamical systems approach to timing is the temporal coordination of different dofs. In robotics, the control of two dof is generally achieved by considering the dofs are completely independent. Therefore, this problem reduces to the one of controlling two robot arms instead of one. However, in motor control of biological systems where there are numerous dofs, this independence is not verified. Movement coordination requires some form of planning: every dof needs to be supplied with appropriate motor commands at every moment in time. However, there exist actually an infinite number of possible movement plans for any given task. A rich area of research has been evolving to study the computational principles and implicit constraints in the coordination of multiple dofs, specifically the question whether or not there are specific principles in the organization of central nervous systems, that coordinate the movements of individual dofs. This research has been mainly directed towards coordination of rhythmic movement. In rhythmic movements, behavioural characteristics show off in that the oscillations verified in these dofs remain coupled in-phase, and also because the dofs show a tendency to become coupled in a determined way (Schaal, 2000). In reaching movements, behavioural characteristics reveal in the synchronization and/or sequencing of movements with different on-set times.

The coordination of rhythmic movements has been addressed within the dynamic theoretical approach (Schoner,1990). These dynamic concepts can be generalized to understand the coordination of discrete movement. Temporal coordination of discrete movements is enabled through the coupling among the dynamics of several such systems such that in the periodic regime the two forms of movement, stable in-phase and anti-phase, are recovered. This is achieved by introducing a discrete movement for each dof (end-effector), and two dofs are considered. The idea is to couple the two discrete movements with the same frequency in a specific way, determined by well established parameters, such that within a certain time the two movements become locked at a given relative phase. The coupling term is multiplied with the neuronal activation of the other system's "Hopf" state such that coupling is effective only when both components are in movement state. This is achieved by modifying the "Hopf" contribution to the timing dynamics as follows

$$\begin{pmatrix} \dot{x}_1 \\ \dot{y}_1 \end{pmatrix} = \ldots + |u_{hopf,1}| \left[f_{hopf,1}(x_1, y_1) + |u_{hopf,2}| cR(\theta) \begin{pmatrix} x_2 - x_1 \\ y_2 - y_1 \end{pmatrix} \right] \ldots \tag{14}$$

$$\begin{pmatrix} \dot{x}_2 \\ \dot{y}_2 \end{pmatrix} = \ldots + |u_{hopf,2}| \left[f_{hopf,2}(x_2, y_2) + |u_{hopf,1}| cR(-\theta) \begin{pmatrix} x_1 - x_2 \\ y_1 - y_2 \end{pmatrix} \right] \ldots \tag{15}$$

where index i = 1, 2 refers to the dof 1 and 2, respectively, and θ is the desired relative phase. This coordination through coupling approaches the generation of coordinated patterns of activation in locomotory behaviour of nervous biological systems.

3.1 An Example: Two 6 Dof Arms Coupled

Consider two PUMA arms performing a straight line motion in 3D Cartesian space (Schoner & Santos, 2001; Santos, 2003). In the simulations, the inverse kinematics of the PUMA arms were based on the exact solution. Each robot arm initiates its timed movement from an initial posture to a final one. Movement parameters such as initial posture, movement initiation, amplitude and movement time are set for each arm individually.

Each arm is driven by a complete system of timing and neural dynamics. Further, the two timing dynamics are coupled as described by Equations 14 and 15. The specified relative phase is θ = 0º. In discrete motor acts, a coupling of this form tends to synchronize movement in the two components, a tendency captured in terms of relative timing of the movements of both components. The two robot arms are temporally coordinated: if movement parameters such as movement on-sets or movement times are not identical, the control level coordinates the two components such that the two movements terminate approximately simultaneously.

This coupling among two timing systems helps synchronize systems and reduces the computational requirements for determining identical movement parameters across such components. Even if there is a discrepancy in the MT programmed by the parameter ω of the timing dynamics, coupling generates identical effectives MTs. This discrete analogue of frequency locking is illustrated in left panel of Figure 6.

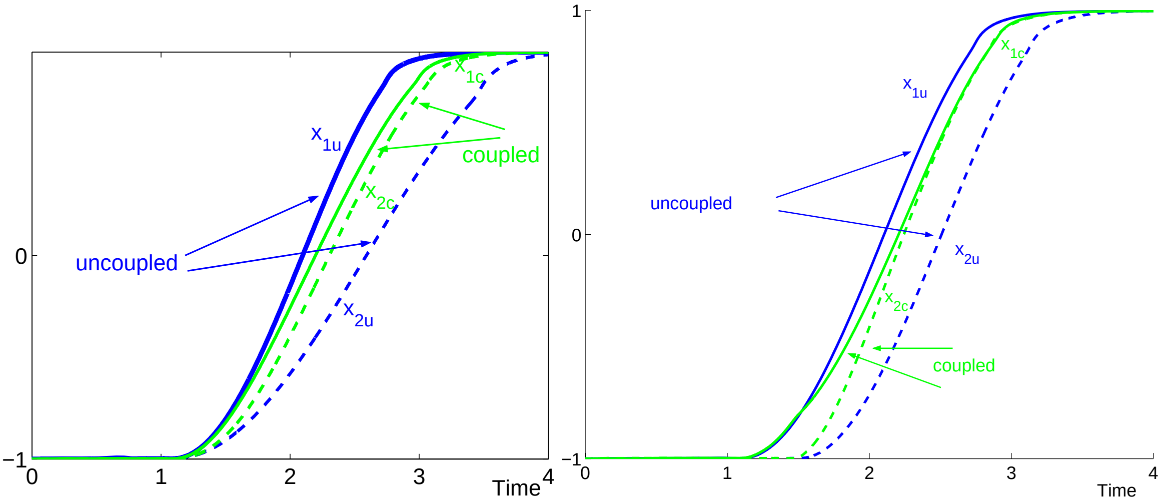


Figure 6. a) Coordination between two timing dynamics through coupling leads to synchronization when movement times differ (2 vs. 3). b) Movement initiation is slightly asynchronous t_{init1} = 1 and t_{init2} = 1.4s (Δt_{init} = 5 % of *MT*), MT_1 = MT_2 = 2s and c = 1)

This tendency to synchronize is also verified when both movements exhibit equal movement times but the on-sets are not perfectly synchronized (right panel of Figure 6). In this case, the effect in the delayed component is to move faster, again in the direction of restoring synchronization.

In case we set a relative phase of θ = 180º we verify anti-phase locking. In the case of discrete movement, anti-phase locking leads to a tendency to perform movements sequentially. Thus, if movement initiation is asynchronous, the movement time of the delayed movement increases such that the movements occur with less temporal overlap.

4. Integration of Different Dynamical Architectures

As an implementation of the approach, the capacity of a low level vehicle to navigate in a non-structured environment while being capable of reaching a target in an approximately constant time is chosen.

4.1 Attractor Dynamics for Heading Direction

The robot action of turning is generated by varying the robot's heading direction, ϕ_h, measured relative to an allocentric coordinate system, as a solution of a dynamical system

(Schöner and Dose, 1992). This behavioural variable is governed by a nonlinear vector field in which task constraints contribute independently by modelling desired behaviours (*target acquisition*) as attractors and undesired behaviours (*obstacle avoidance)* as repellers of the overall behavioural dynamics (Bicho, 2000).

Target location, (x_B, y_B), is continuously extracted from visual segmented information acquired from the camera mounted on the top of the robot and facing in the direction of the driving speed. The angle ϕ_h of the target's direction as "seen" from the robot is:

$$\phi_{tar} = \arctan \frac{y_B - y_R}{x_B - x_R} \Leftrightarrow \phi_{tar} = \arctan \frac{R_{y_B}}{R_{x_B}} \tag{16}$$

where (x_R, y_R) is the current robot position in the allocentric coordinate system as given by the dead-reckoning mechanism.

Integration of the *target acquisition (behaviour f_{tar})* and *obstacle avoidance (behaviour f_{obs})* contributions is achieved by adding each of them to the vector field that governs heading direction dynamics

$$\frac{d(\phi_h)}{dt} = F_{obs}(\phi_h) + f_{tar}(\phi_h) + f_{stoch}(\phi_h) \tag{17}$$

We add a stochastic component force, F_{stoch}, to ensure escape from unstable states within a limited time. The complete behavioural dynamics for heading direction has been implemented and evaluated in detail on a physical mobile robot (Bicho, 2000).

4.2 The Dynamical Systems of Driving Speed

Robot velocity is controlled such that the vehicle has a fixed time to reach the target. Thus, if the vehicle takes longer to arrive at the target because it needed to circumnavigate an obstacle, this change of timing must be compensated for by accelerating the vehicle along its path.

The path velocity, *v*, of the vehicle is controlled through a dynamical system architecture that generates timed trajectories for the vehicle. We set two spatially fixed coordinates frames both centred on the initial posture, which is the origin of the allocentric coordinate system: one for the *x* and the other for the *y* spatial coordinates of robot movement. A complete system of timing and neural dynamics is defined for each of these fixed coordinate frames. Each model consists of a timing layer (Schoner & Santos, 2001), which generate both stable oscillations (contribution f_{hopf}) and two stationary states (contributions "*init*" and "*final*").

$$\begin{pmatrix} \dot{x}_i \\ \dot{a}_i \end{pmatrix} = |u_{init,i}| f_{init,i} + |u_{hopf,i}| f_{hopf,i} + |u_{final,i}| f_{final,i} + gwn \tag{18}$$

where the index i = *x*, *y* refers to timing dynamics of *x* and *y* spatial coordinates of robot movement. A neural dynamics controls the switching between the three regimes through three neurons, $u_{j,i}$ (*j = init, hopf, final*) (Equation 20). The "init" and "final" contributions generate stable stationary solutions at $x_i = 0$ for "init" and A_{ic} for "final" with $a_i = 0$ for both. These states are characterized by a time scale of $\tau = 1/5 = 0.2$.

The "Hopf" contribution to the timing dynamics is defined as follows:

$$f_{hopf,i} = \begin{pmatrix} \alpha_h & -\omega \\ \omega & \alpha_h \end{pmatrix} \begin{pmatrix} x_i - \frac{A_{ic}}{2} \\ a_i \end{pmatrix} - \gamma_i \left(\left(x_i - \frac{A_{ic}}{2} \right)^2 + a_i^2 \right) \begin{pmatrix} x_i - \frac{A_{ic}}{2} \\ a_i \end{pmatrix} \tag{19}$$

where $\lambda_i = \frac{4\alpha_h}{A_{ic}^2}$ defines amplitude of *Hopf i* contribution.

4.3 Behavioural Specifications

The neuronal dynamics of $u_{j,i} \in [-1; 1]$ (j = *init, hopf, final*) switches each timing dynamics from the initial and final postural states into the oscillatory regime and back, and are given by

$$\alpha_u \dot{u}_{j,i} = \mu_{j,i} u_{j,i} - |\mu_{j,i}| u_{j,i}^3 - v \sum_{a \neq j} u_{a,i}^2 u_{j,i} + gwn \tag{20}$$

We assure that one neuron is always "on" by varying the μ_i -parameters between the values 1.5 and 3.5: $\mu_i = 1.5 + 2b_i$, where b_i are the quasi-boolean factors.

The competitive advantage of the initial postural state is controlled by the parameter b_{init}. This parameter must be "on" (= 1) when either of the following is true: (1) time, t, is bellow the initial time, t_{init}, set by the user ($t < t_{init}$); (2) timing variable x_i is close to the initial state 0 ($b_{xi\ close\ xinit}$ (xi)); **and** time exceeds t_{init} ($t > t_{init}$); **and** target has not been reached.

We consider that the target has not been reached when the distance, d_{tar}, from the actual robot position (as internally calculated through dead-reckoning) and the (x_{target}, y_{target}) position is higher than a specified value, d_{margin}. This logical condition is expressed by the quasi-boolean factor, $b_{xi\ has\ not\ reached\ target}(d_{tar}) = \sigma(d_{tar}\text{-}d_{margin})$, where $\sigma(.)$ is the sigmoid function explained before (Equation 7). Note that this switch is driven from the sensed actual position of the robot.

The factor $b_{xi\ close\ xinit}(x_i) = \sigma(x_{crit}\text{-}x_i)$ has values close to one while the timing variable x_i is bellow $0.15A_{ic}$ and switches to values close to zero elsewhere.

These logical conditions are expressed through the mathematical function:

$$b_{init} = 1 - \{ (t \geq t_{init}) [1 - (b_{x_i\ close\ x_{init}}(x_i)(t \geq t_{init}) b_{x_i\ has\ not\ reached\ target}(d))] \} \tag{21}$$

A similar analysis derives the b_{hopf} and b_{final} parameters:

$$b_{hopf} = (t \geq t_{init}) b_{x_i\ not\ close\ x_{final}}(x_i) b_{x_i\ has\ not\ reached\ target}(d_{tar}) \sigma(b_{update\ A_{ic}}) \tag{22}$$

$$b_{final} = (t \geq t_{init}) [b_{x_i\ not\ close\ x_{final}}(x_i) + b_{x_i\ reached\ target}(d_{tar}) + b_{x_i\ not\ close\ x_{final}}(x_i) + (1 - \sigma(b_{update\ A_{ic}}))] \tag{23}$$

We algorithmically turn off the update of the *i* timed target location, ${}^{T}x_{target}$ or ${}^{T}y_{target}$, once this changes sign relatively to the previous update and the corresponding timing level is in the initial postural state.

The factor $b_{xi\ not\ close\ xfinal}(x_i) = \sigma(d_{switch} - d_{crit})$ is specified based on absolute values, where d_{switch} represents the distance between the timing variable x_i and the final postural state, A_{ic} and d_{crit} is tuned empirically.

The competitive dynamics are the faster dynamics of the all system. Its relaxation time, τ_u, is set ten times faster than the relaxation time of the timing variables ($\tau_u = 0.02$).

The system is designed such that the planning variable is in or near a resulting attractor of the dynamical system most of the time. If we control the driving velocity, *v*, of the vehicle, the system is able to track the moving attractor. Robot velocity depends whether or not obstacles are detected for the current heading direction value. This velocity depends on the behaviour exhibited by the robot and is imposed by a dynamics equal to that described by (Bicho et al, 2000)

$$\frac{dv}{dt} = -c_{obs}\left(v - V_{obs}\right)\exp\left(-\frac{(v - V_{obs})^2}{2\sigma_v^2}\right) - c_{ti\,min\,g}\left(v - V_{ti\,min\,g}\right)\exp\left(-\frac{(v - V_{ti\,min\,g})^2}{2\sigma_v^2}\right) \qquad (24)$$

V_{obs} is computed as a function of distance and is activated when an obstacle is detected. V_{timing} is specified by the temporal level as

$$V_{ti\,min\,g} = \sqrt{\dot{x}_x^2 + \dot{x}_y^2} \qquad (25)$$

For further details regarding this dynamics refer to (Bicho, 2000).

The following hierarchy of relaxation rates ensures that the system relaxes to the stable solutions, *obstacle avoidance* has precedence over *target acquisition* and *target achievement* is performed in time

$$\tau_{v,obs} << \tau_{v,obs}, \tau_{v,ti\,min\,g} << \tau_{tar}, \tau_{obs} << \tau_{tar} \qquad (26)$$

Suppose that at t = 0 s the robot is resting at an initial fixed position, the same as the origin of the allocentric coordinate system. The robot rotates in the spot in order to orient towards the target direction. At time t_{init}, the quasi-boolean for motion, b_{hopf}, becomes one, triggering activation of the corresponding neuron, u_{hopf}, and movement initiation. Movement initiation is accomplished by setting the driving speed, *v*, different from zero. During periodic movement, the target location in time is updated each time step based on error, x_R - ${}^{T}x_x$, such that

$${}^{T}x_{t\,arg\,et} = x_{t\,arg\,et} - \left(x_R - {}^{T}x_x\right) \qquad (27)$$

where ${}^{T}x_x$ is the current timing variable x_x, x_r is the x robot position and is the timing variable x_x. The periodic motion's amplitude, A_{xc}, is set as the distance between ${}^{T}x_{target}$ and the origin of the allocentric reference frame (which is coincident with the x robot position previously to movement initiation), such that

$$A_{xc}(t) = {}^{T}x_{t\,arg\,et}(t) \qquad (28)$$

The periodic solution is deactivated again when the vehicle comes into the vicinity of the x timed target, and the final postural state is turned on instead (neurons $|u_{hopf}| = 0$; $|u_{final}| = 1$). At this moment in time, the *x* timed target location is no longer updated in the timing dynamics level.

The same behaviour applies for the timing level defined for the *y* spatial coordinate.

4.4 Experimental Results

The dynamic architecture was implemented and evaluated on an autonomous wheeled vehicle (Santos, 2004). The dynamics of heading direction, timing, competitive neural, path velocity and dead-reckoning equations are numerically integrated using the Euler method with fixed time step.

Image processing has been *simplified* by working in a structured environment, where a red ball lies at coordinates (xB; yB) = (-0.8; 3.2) m (on the allocentric coordinate system) on the top of a table at approximately 0.9m tall. The initial heading direction is 90 degrees. The sensed obstacles do not block vision. An image is acquired only every 10 sensorial cycles such that the cycle time is 70 ms, which yields a movement time (*MT*[4]) of 14s. Forward

[4] Specified time for the robot to meet the ball after the forward movement is initiated.

movement initiation is triggered by an initial time set by the user and not from sensed sensorial information. Forward movement only starts for tinit = 3s.

The rotation speeds of both wheels are computed from the angular velocity, w, of the robot and the path velocity, v. The former is obtained from the dynamics of heading direction. The later, as obtained from the velocity dynamics is specified either by *obstacle avoidance contribution* or the *timing dynamics*. By simple kinematics, these velocities are translated into the rotation speeds of both wheels and sent to the velocity servos of the two motors.

A. Properties of the Generated Timed Trajectory

The sequence of video images shown in Figure 7 illustrates the robot motion in a very simple scenario: during its path towards the target, the robot faces two obstacles separated of 0.7m, which is a distance larger enough for the robot to pass in between. The time courses of the relevant variables and parameters are shown in Figure 8.

At time t = 7.2 s obstructions are detected and the velocity dynamics are dominated by the obstacle constraints (bottom panel of Figure 8). Due to obstructions circumnavigation, the robot position differs from what it should be according to the timing layer (at time t = 11.5s). The x robot position is advanced relatively to the ${}^{T}x_{x}$ timing dynamics specifications and A_{xc} is decreased relatively to x_{target} (Figure 8).

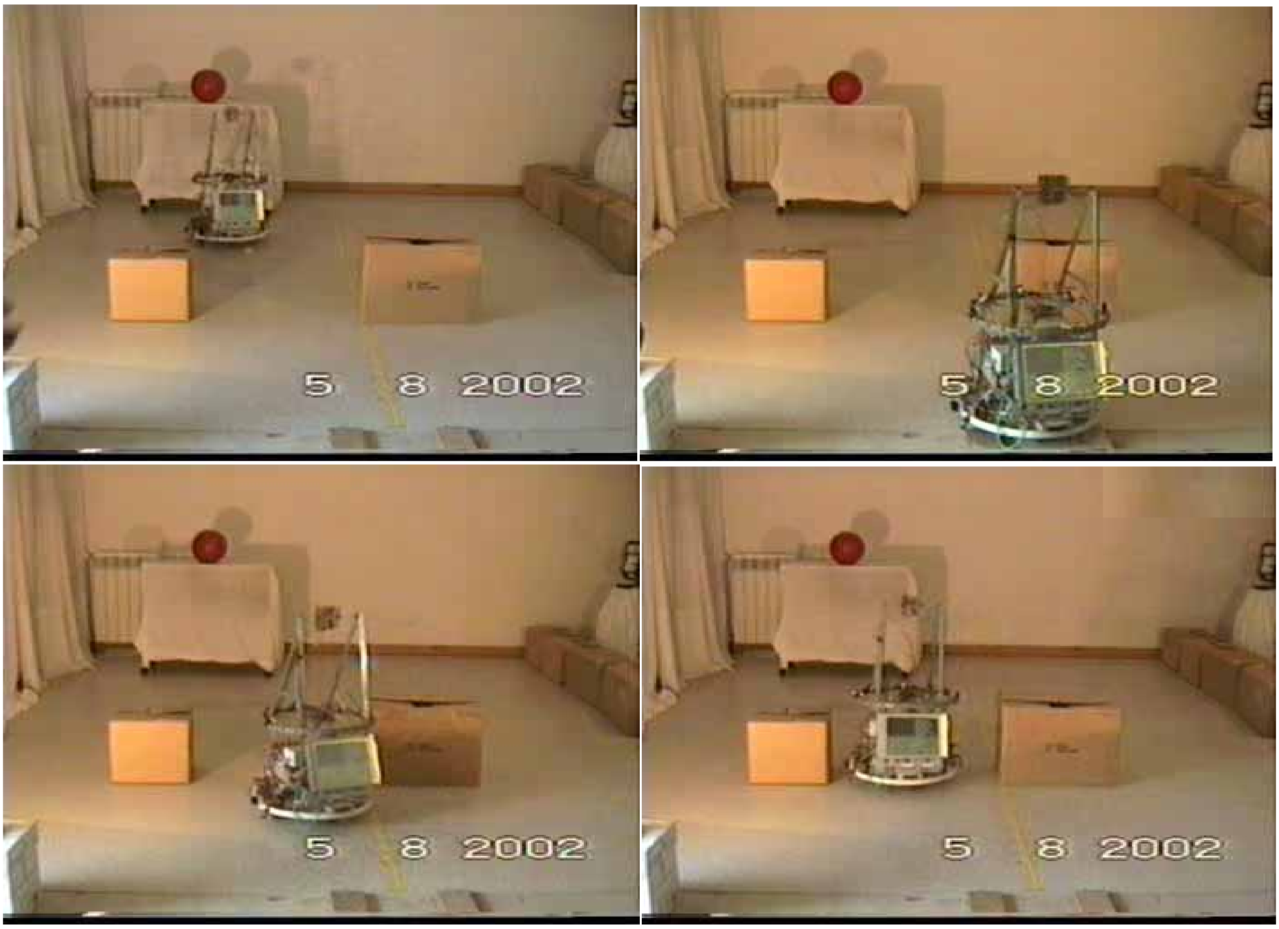


Figure 7. Robot motion when the robot faces two objects separated of 0.7m during its path. The robot successfully passes through the narrow passage towards the target and comes to rest at a distance of 0.9m near the red ball at t = 16.56s. The effective movement time is 13.56 s

Therefore, the robot velocity is de-accelerated in this coordinate. Conversely, the y robot position lags the ${}^{T}x_{y}$ timing variable and robot velocity is accelerated in this coordinate (third panel of Figure 8). Finally, the target is reached and the robot comes to rest at a distance of 0.90m near the red ball. The overall generated timed trajectory takes t = 16.56 - 3 s to reach the target (forward movement started at time t = 3s).

This trajectory displays a number of properties of dynamical decision making. Figure 8 shows how the hysteresis property allows for a special kind of behavioral stability. At t = 15s the quasi-boolean parameter $b_{x,hopf}$ becomes zero but the $u_{x,hopf}$ neuron remains activated until the neuron $u_{x,final}$ is more stable, what happens around t = 16.2s. At this time, the x periodic motion is turned off. Thus, hysteresis leads to a simple kind of memory which determines system performance depending on its past history.

Figure 9 shows the robot trajectory as recorded by the dead-reckoning mechanism when the distance between the two obstacles is smaller than the vehicle's size (0.3m). The path followed by the robot is qualitatively different. In case timing dynamics stabilize the velocity dynamics the robot is strongly accelerated in order to compensate for the object circumnavigation. Light crosses on the robot trajectory indicate robot positions where vision was not acquired because the robot could not see the ball. The ball position as calculated by the visual system slight differs from the real robot position (indicated by a dark circle).

B. Trajectories Generated with and without Timing Control

Table **1** surveys the time the robot takes to reach the target lying at coordinates (-0.8, 3.24) m for several configurations when path velocity, v, is controlled with and without timing control. In the latter, path velocity is specified differently: when no obstructions are detected the robot velocity is stabilized by an attractor, which is set proportional to the distance to the target (Bicho, 2000). Note that forward movement starts immediately. Conversely, forward movement only happens at t = 3 s when there is timing control. The specified movement time is 14s.

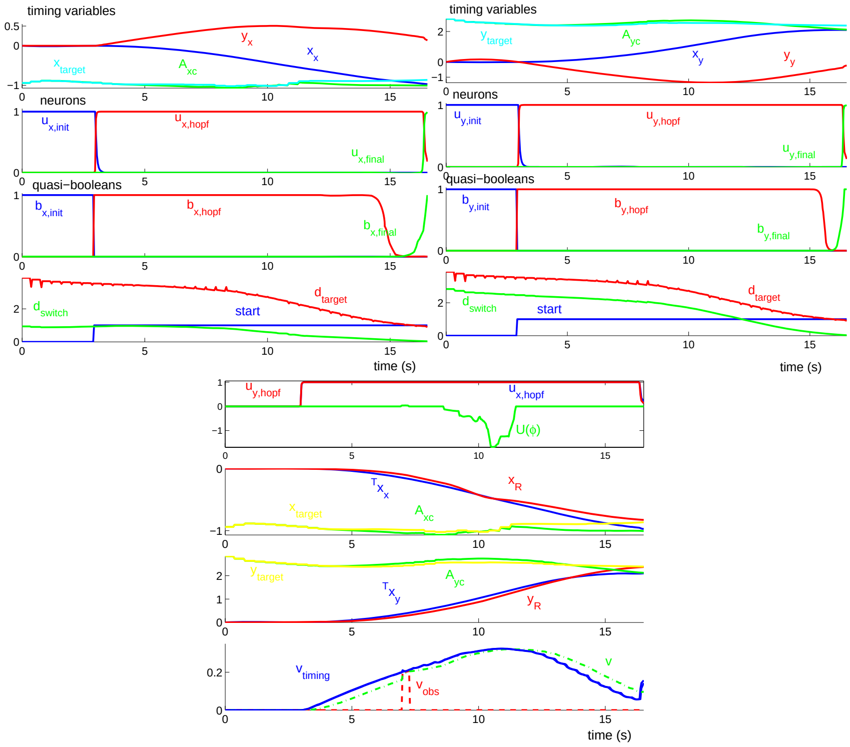


Figure 8. Time courses of variables and parameters for the robot trajectory depicted in Figure 7. Top panels depict timing and neural dynamics (x and y coordinate in left and right panels, respectively). Bottom panel depicts neural and timing variables, robot trajectories, real target locations, periodic motion amplitudes and velocity variables.

We observe that both controllers have stably reached the target but the former is capable of doing it in an approximately constant time independently of the environment configuration.

Experiments	Time to reach target with timing and MT	Time to reach target without timing
No obstacles	16.8 (13.8)	18.6
One obstacle	16.6 (13.6)	19.0
obstacles separated 0.8m	16.5 (13.5)	18.9
obstacles separated 0.7m	16.6 (13.6)	19.0
obstacles separated 0.3m	18.8 (15.8)	23.6
Complex configuration 1	19.3 (16.3)	22.5
Complex configuration 2	17.2 (14.2)	19.2
Complex configuration 3	17.4 (14.4)	19.6
Complex configuration 4	16.8 (13.8)	19.7
Complex configuration 5	17.3 (14.3)	18.3
Complex configuration 6	17.7 (14.7)	19.8
Complex configuration 7	23.3 (20.3)	28.0

Table 1. Time (in seconds) the robot takes to reach a target for several environment configurations, when robot's forward velocity, v, is controlled with and without timing control

We have also compared time the robot takes to reach the target when velocity is controlled with and without timing control for different target locations and same configurations as in Table **1**. The results have shown that the achieved movement time is approximately constant and independent of the distance to the target.

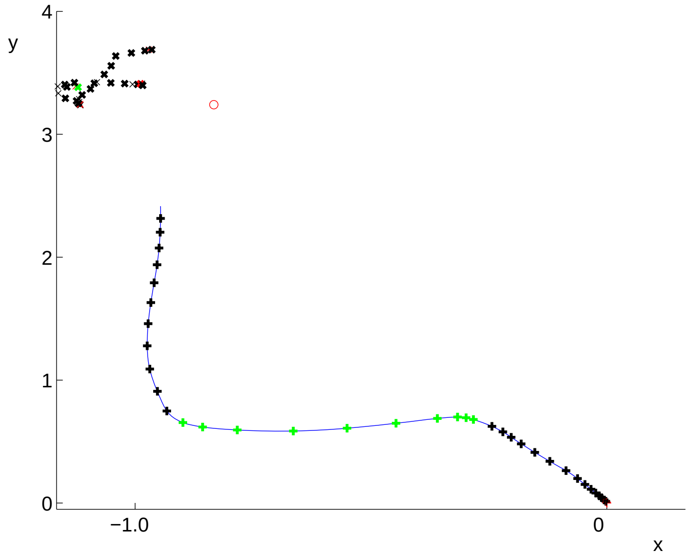


Figure 9. Robot trajectory as recorded by the dead-reckoning mechanism when obstacles are separated of 0.3m

5. Conclusion and Discussion

This paper addressed the problem of generating timed trajectories and sequences of movements for autonomous vehicles when relatively low-level, noisy sensorial information is used to initiate and steer action. The developed architectures are fully formulated in terms of nonlinear dynamical systems. The model consists of a timing layer with either stable fixed points or a stable limit cycle. The qualitative dynamics of this layer is controlled by a neural competitive dynamics. By switching between the limit cycle and the fixed points, discrete movements and sequences of movements are obtained. These switches are controlled by the parameters of the neural dynamics which express sensory information and logical conditions. Coupling to sensorial information enables sensor driven and initiation. To corroborate the proposed solution experiments were performed using robot arms and low-level autonomous vehicles. The implemented decision making mechanism allowed the system to flexibly respond to the demands of the sensed environment at any given situation. The generated sequences were stable and a decision maintained stable by the hysteresis property. The described implementation in hardware probes how the inherent stability properties of neural and timing dynamics play out when the sensory information is noisy and unreliable.

Some aspects are unique to this work and have enabled the introduction of timing constraints. We have shown how the attractor dynamics approach to the generation of behaviour can be extended to the timing of motor acts. Further, we have shown that by manipulating the timing of a limit cycle the system performed well tasks with complex timing constraints.

The dynamical systems approach has various desirable properties. Firstly, its inherent properties, such as temporal, scale and translation invariance relatively to the tuning parameters, provide the ability to modify online the generated attractor landscape to the demands of the current situation, depending on the sensorial context. Because movement plans are generated by the time evolution of autonomous differential equations, they are not explicitly indexed by time, and thus by means of coupling perceptual variables to the dynamic equations, they can accomplish flexible on-line modification of the basic behaviors. This property enables to create several forms of on-line modifications, e.g., based on contact forces in locomotion, perceptual variables in juggling or the tracking error of a robotic system (Ijspeert et al., 2002). A globally optimized behaviour is achieved through local sensor control and global task constraints, expressed through the logics contained in the parameters of the differential equations and not in an explicit program. A smooth stable integration of discrete events and continuous processes is thus achieved. Further, we guarantee the stability and the controllability of the overall system by obeying the time scale separation principle. Further, this approach does not make unreasonable assumptions, or place unreasonable constraints on the environment in which the robot operates and assures a quick reaction to eventual changes in the sensed environment.

The ease with which the system is integrated into larger architectures for behavioural organization that do not necessarily explicitly represent timing requirements is a specific advantage of our formulation. This integration enables to achieve behavioural organization. By obeying the time scale separation principle we design the ordering principle for the coupled behavioural dynamics. This scalability property implies a high modularity. On the opposite, the integration of new behaviours, using symbolic representations, obliges to redesign the whole behavioural system.

Another advantage of our specific formulation is the fact that it is possible to parameterize the system by analytic approximation, which facilitates the specification of parameters such as movement time, movement extent, maximal velocity, etc. Not only we have generated discrete movement as well as we provide a theoretically based way of tuning the dynamical parameters to fix a specific movement time or extent. Comparatively, other non-linear approaches, such as (Buhler et al, 1994; Raibert, 1986), the overall movement parameters emerge from the interaction of the control system with the environment so that achieving specific movement times or amplitudes is only possible by empirical tuning of parameters. While these approaches only achieved rhythmic movement, (Schaal et al., 2000) have, like us, been able to generate temporally discrete movement as well. It does not appear, however, that there is a theoretically based way of tuning the dynamical parameters to fix a specific movement time or extent.

Further, the approach is a powerful method for obtaining temporal coordinated behaviour of two robot arms. The coupled dynamics enable synchronization or sequentialization of the different components providing an independency relatively to the specification of their individual movement parameters. Such coupling tends to synchronize movement in the two components such that the computational requirements for determining identical movement parameters across such components are reduced. From the view point of engineering applications, the inherent advantages are huge, since the control system is released from the task of recalculating the movement parameters of the different components.

6. References

Arkin, R C. (1998) Behavior-Based Robotics. MIT Press, Cambridge.

Bajcsy, R. and Large, E. (1999) When and where will AI meet robotics? *AI Magazine*, 20:57-65.

Beer, R D; Chiel, H J and Sterling, L S. (1990) A biological perspective on autonomous agent design. *Robotics and Autonomous Systems*, 6:169-189.

Bicho, E. (2000) Dynamic Approach to Behavior-based Robotics Design, Specification, Analysis, Simulation and Implementation. Shaker-Verlag, Phd Thesis, Aachen.

Bicho, Estela; Mallet, Pierre and Schoner, G. (2000) Target representation on an autonomous vehicle with low-level sensors. *The International Journal of Robotics Research*, 19, 424-447.

Bühler, M; Koditscheck, D E and Skinner, R D. (1994) Planning and control of a juggling robot. *International Journal of Robotics Research*, 13(2):101-118.

Clark, M R; Anderson, G T and Skinner, R D. (2000) Coupled oscillator control of autonomous mobile robots. *Autonomous Robots*, 9:189-198.

Collins, J; Richmond, S, (1994) Hard-wired central pattern generators for quadruped locomotion, *Biological Cybernetics*, 71, 375-385.

Ijspeert, A.J.; Nakanishi J., Schaal S., (2002): Learning Rhythmic Movements by Demonstration using Nonlinear Oscillators, *Proceedings of the IEEE/RSJ Int. Conference on Intelligent Robots and Systems* (IROS2002), 958-963.

Ito,S; Yuasa,H; Luo,S; Ito,M and Yanagihara,D (1998) A mathematical model of adaptive behaviour in quadruped locomotion, *Biological Cybernetics*, 78:337-347.

Khatib, O. (1986) Real-time obstacle avoidance for manipulators and mobile robots. *International Journal Robotics Research*, 5(1):90-98.

Large, E W; Christensen, H I and Bajcsy, R. (1999) Scaling the dynamic approach to path planning and control: Competition among behavioral constraints. *International Journal of Robotics Research*, 18(1):37-58.

Perko, L. (1991) Differential Equations and Dynamical Systems. Springer-Verlag.

Raibert, M (1986) Legged robots that balance. MIT Press, Cambridge, Massachusetts.

Santos, Cristina (2004) Generating Timed Trajectories for an Autonomous Vehicle: A Non-linear Dynamical Systems Approach. *In IEEE International Conference on Robotics and Automation* April 26-1 May, New Orleans, LA USA, 3741-3746.

Santos, Cristina, (2003) Attractor dynamics based generation of timed robotic trajectories, *PhD Thesis*, November.

Schaal, S and Atkeson, C (1993) Open loop stable control strategies for robot juggling. *In IEEE International Conference on Robotics and Automation*, vol. 3, 913-918.

Schaal, S; Kotosaka, S and Sternad, D. (2000) Nonlinear dynamical systems as movement primitives. *In IEEE International Conference on Humanoid Robotics*. IEEE, Cambridge, MA.

Schöner, G and Kelso, J. (1988) Dynamic pattern generation in behavioural and neural systems. *Science*, 239:1513-1520.

Schöner, Gregor (1994a) Dynamic theory of action-perception patterns: The time-before-contact paradigm. *Human Movement Science*, 3: 415-439.

Schöner, Gregor and Dose, Michael. (1992) A dynamical systems approach to task-level system integration used to plan and control autonomous vehicle motion. *Robotics and Autonomous Systems*, 10:253-267.

Schöner, Gregor and Santos, Cristina. (2001) Control of movement time and sequential action through attractor dynamics: A simulation study demonstrating object interception and coordination. *In 9th Intelligent Symposium on Intelligent Robotic Systems - SIRS'2001*, Toulouse, France, 18-20,July.

Schöner, Gregor. (1990) A dynamic theory of coordination of discrete movement. *Biological Cybernetics*, 63:257-270.

Shik, M; Severin, F and Orlovsky, G. Control of walking by means of electrical stimulation of the midbrain. Biophysics, 11:756-765, 1966

Steinhage, A. and Schöner, G. (1998) Dynamical systems for the behavioral organization of autonomous robot navigation. In *Sensor Fusion and Decentralized Control in Robotic Systems: Proceedings of SPIE*, volume 3523, 169-180.

Williamson, Matthew. (1998) Rhythmic robot arm control using oscillators. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'98)*, Victoria, B.C., Canada, October.

Lee, D. (1976) A theory of visual control of breaking based on information about time-to-collision. *Perception*, 5:437-459.

Coevolution Based Adaptive Monte Carlo Localization

Luo Ronghua, Hong Bingrong & Li Maohai

1. Introduction

Self-localization, a basic problem in mobile robot systems, can be divided into two sub-problems: pose tracking and global localization. In pose tracking, the initial robot pose is known, and localization seeks to identify small, incremental errors in a robot's odometry (Leonard & Durrant-Whyte, 1991). In global localization, however the robot is required to estimate its pose by local and incomplete observed information under the condition of uncertain initial pose. Global localization is a more challenging problem. Only most recently, several approaches based on probabilistic theory are proposed for global localization, including grid-based approaches (Burgard et al., 1996), topological approaches (Kaelbling et al., 1996) (Simmons & Koenig, 1995), Monte Carlo localization (Dellaert et al., 1999) and multi-hypothesis tracking (Jensfelt & Kristensen, 2001) (Roumeliotis & Bekey, 2000). By representing probability densities with sets of samples and using the sequential Monte Carlo importance sampling (Andrieu & Doucet, 2002), Monte Carlo localization (MCL) can represent non-linear and non-Gaussian models well and focus the computational resources on regions with high likelihood. So MCL has attracted wide attention and has been applied in many real robot systems.

But traditional MCL has some shortcomings. Since samples are actually drawn from a proposal density, if the observation density moves into one of the tails of the proposal density, most of the samples' non-normalized importance factors will be small. In this case, a large sample size is needed to represent the true posterior density to ensure stable and precise localization. Another problem is that samples often too quickly converge to a single, high likelihood pose. This might be undesirable in the case of localization in symmetric environments, where multiple distinct hypotheses have to be tracked for extended periods of time. How to get higher localization precision, to improve efficiency and to prevent premature convergence of MCL are the key concerns of the researchers. To make the samples represent the posterior density better, Thrun et al. proposed mixture-MCL (Thrun et al., 2001), but it needs much additional computation in the sampling process. To improve the efficiency of MCL, methods adjusting sample size adaptively over time are proposed (Fox, 2003) (Koller & Fratkina, 1998), but they increase the probability of premature convergence. Although clustered particle filters are applied to solve premature convergence (Milstein et al., 2002), the method loses the advantage of focusing the computational resources on regions with high likelihood because it maintains the same sample size for all clusters. In this paper, a new version of MCL is proposed to overcome those limitations. Samples are clustered into groups which are also called species. A coevolutionary model derived from competition of ecological species is introduced to make the species evolve cooperatively, so the premature convergence in highly symmetric environment can be prevented. The population growth model of species enables the sample size to be adjusted according to the total environment resources which represent uncertainty of the pose of the robot. And genetic operators are used for intra-species evolution to search for optimal samples in each species. So the samples can represent the desired posterior density better, and precise localization can be realized with a small size of sample. Compared with the traditional MCL, the new algorithm has the following advantages: (1) it can adaptively adjust the sample size during localization; (2) it can make stable localization in highly symmetric environment; (3) it can make precise localization with a small sample size.

2. Background

2.1 Robot Localization Problem

Robot localization is to estimate the current state x_t of the robot, given the information about initial state and all the measurements Y^t up to current time:

$$Y^t = \{y_t \mid t = 0,1,\cdots,t\} \tag{1}$$

Typically, the state x_t is a three-dimensional vector including the position and direction of the robot, i.e. the pose of the robot. From a statistical point of view, the estimation of x_t is an instance of Bayes filtering problem, which can be implemented by constructing the posterior density $p(x_t \mid Y^t)$. Assuming the environment is a Markov process, Bayes filters enable $p(x_t \mid Y^t)$ to be computed recursively in two steps.

Prediction step: Predicting the state of the next time-step with previous state x_{t-1} according to the motion model $p(x_t \mid x_{t-1}, u_{t-1})$:

$$p(x_t \mid Y^{t-1})_{t-1} = \int p(x_t \mid x_{t-1}, u_{t-1}) p(x_{t-1} \mid Y^{t-1}) dx \tag{2}$$

Update step: Updating the state with the newly observed information y_t according to the perceptual model $p(y_t \mid x_t)$:

$$p(x_t \mid Y^t) = \frac{p(y_t \mid x_t)\, p(x_t \mid Y^{t-1})}{p(y_t \mid Y^{t-1})} \tag{3}$$

2.2 Monte Carlo localization (MCL)

If the state space is continuous, as is the case in mobile robot localization, implementing equations (2) and (3) is not trivial. The key idea of MCL is to represent the posterior density $p(x_t \mid Y^t)$ by a set of weighted samples S_t:

$$S_t = \{(x_t^{(j)}, w_t^{(j)}) \mid j = 1,\cdots,N\} \qquad (4)$$

Where $x_t^{(j)}$ is a possible state of the robot at current time *t*. The non-negative numerical factor $w_t^{(j)}$ called importance factor represents the probability that the state of robot is $x_t^{(j)}$ at time *t*. MCL includes the following three steps:

(1) Resampling: Resample *N* samples randomly from S_{t-1}, according to the distribution defined by w_{t-1};

(2) Importance sampling: sample state $x_t^{(j)}$ from $p(x_t \mid x_{t-1}^{(j)}, u_{t-1})$ for each of the *N* possible state $x_{t-1}^{(j)}$; and evaluate the importance factor $w_t^{(j)} = p(y_t \mid x_t^{(j)})$.

(3) Summary: normalize the importance factors $w_t^{(j)} = w_t^{(j)} / \sum_{k=1}^{N} w_t^{(k)}$; and calculate the statistic property of sample set S_t to estimate the pose of the robot.

2.3 Coevolutionary Algorithms

Evolutionary algorithms (EAs), especially genetic algorithms, have been successfully used in different mobile robot applications, such as path planning (Chen & Zalzala, 1995) (Potvin et al., 1996), map building (Duckett, 2003) and even pose tracking for robot localization (Moreno et al., 2002). But premature convergence is one of the main limitations when using evolutionary computation algorithms in more complex applications as in the case of global localization.

Coevolutionary algorithms (CEAs) are extensions of EAs. Based on the interaction between species, coevolution mechanism in CEAs can preserve diversity within the population of evolutionary algorithms to prevent premature convergence. According to the characters of interaction between species, CEAs can be divided into cooperative coevolutionary algorithms and competitive coevolutionary algorithms. Cooperative (also called symbiotic) coevolutionary algorithms involve a number of independently evolving species which together form complex structures. The fitness of an individual depends on its ability to collaborate with individuals from other species. Individuals are rewarded when they work well with other individuals and punished when they perform poorly together (Moriarty & Miikkulainen, 1997) (Potter & De Jong, 2000). In competitive coevolutionary algorithms, however the increased fitness of one of the species implies a diminution in the fitness of the other species. This evolutionary pressure tends to produce new strategies in the populations involved so as to maintain their chances of survival. This "arms race" ideally increases the capabilities of the species until they reach an optimum. Several methods have been developed to encourage the arms race (Angeline & Pollack, 1993) (Ficici & Pollack, 2001) (Rosin & Belew, 1997), but these coevolution methods only consider interaction between species and neglect the effects of the change of the environment on the species.

Actually the concept of coevolution is also derived from ecologic science. In ecology, much of the early theoretical work on the interaction between species started with the Lotka-Volterra model of competition (Yuchang & Xiaoming, 1996). The model itself was a modification of the logistic model of the growth of a single population and represented the result of competition between species by the change of the population size of each species. Although the model could not embody all the complex relations between species, it is simple and easy to use. So it has been accepted by most of the ecologists.

3. Coevolution Based Adaptive Monte Carlo Localization

To overcome the limitations of MCL, samples are clustered into different species. Samples in each species have similar characteristics, and each of the species represents a hypothesis of the place where the robot is located. The Lotka-Volterra model is used to model the competition between species. The competition for limited resources will lead to the extinction of some species, and at the same time make some species become more complex so as to coexist with each other. The environment resources which represent uncertainty of the pose of the robot will change over time, so the total sample size will also change over time. Compared with other coevolution models, our model involves the effects of competition between species as well as that of the change of environment on species.

3.1 Initial Species Generation

In the traditional MCL, the initial samples are randomly drawn from a uniform distribution for global localization. If a small sample size is used, few of the initial samples will fall in the regions where the desired posterior density is large, so MCL will fail to localize the robot correctly. In this paper, we propose an efficient initial sample selection method, and at the same time the method will cluster the samples into species.

In order to select the samples that can represent the initial location of the robot well, a large test sample set $S_{test} = \{(\tilde{x}_0^{(1)}, \tilde{w}_0^{(1)}), \cdots, (\tilde{x}_0^{(N_{test})}, \tilde{w}_0^{(N_{test})})\}$ with N_{test} samples is drawn from a uniform distribution over the state space, here $\tilde{w}_0^{(j)} = p(y_0 \mid \tilde{x}_0^{(j)})$. Then the multi-dimensional state space of the robot is partitioned into small hyper-rectangular grids of equal size. And samples in S_{test} are mapped into the grids. The weight of each grid is the average importance factor of the samples that fall in it. A threshold $T = \mu$ is used to classify the grids into two groups, here the coefficient $\mu \in (0,1)$. Grids with weight larger than T are picked out to form a grid set V . The initial sample size N_0 is defined by:

$$N_0 = \eta |V| / \overline{w}_0 \tag{5}$$

Where η is a predefined parameter, $\overline{w}_0$ is the average weight of grids in set V , and $|V|$ is the number of grids in set V . This equation means that if the robot locates in a small area with high likelihood, a small initial sample size is needed.

Using the network defined through neighborhood relations between the grids, the set V is divided into connected regions (i.e. sets of connected grids). Assuming there are Ω connected regions, these connected regions are used as seeds for the clustering procedure. A city-block distance is used in the network of grids. As in image processing field, the use of distance and seeds permits to define influence zones, and the boundary between influence zones is known as SKIZ (skeleton by influence zone) (Serra, 1982). So the robot's state space is partitioned into Ω parts. And $N_0^{(i)}$ samples which have the largest importance factor will be selected from the test samples falling in the *i*th part.

$$N_0^{(i)} = N_0 \cdot \overline{w}_0(i) / \sum_{k=1}^{\Omega} \overline{w}_0(k) \tag{6}$$

Where $\overline{w}_0(k)$ is the average weight of grids in the kth part of the state space. The selected $N_0^{(i)}$ samples from the *i*th part form an initial species of $N_0^{(i)}$ population size.

3.2 Inter-Species Competition

Inspired by ecology, when competing with other species the population growth of a species can be modeled using the Lotka-Volterra competition model. Assuming there are two species, the Lotka-Volterra competition model includes two equations of population growth, one for each of two competing species.

$$\frac{dN_t^{(1)}}{dt} = r^{(1)} N_t^{(1)} \left(1 - \frac{N_t^{(1)} + \alpha_t^{(12)} N_t^{(2)}}{K_t^{(1)}}\right) \tag{7}$$

$$\frac{dN_t^{(2)}}{dt} = r^{(2)} N_t^{(2)} \left(1 - \frac{N_t^{(2)} + \alpha_t^{(21)} N_t^{(1)}}{K_t^{(2)}}\right) \tag{8}$$

Where $r^{(i)}$ is the maximum possible rate of population growth, $N_t^{(i)}$ is the population size and $K_t^{(i)}$ is the upper limit of population size of species *i* that the environment resource can support at time step *t* respectively, and $\alpha_t^{(ij)}$ refers to the impact of an individual of species *j* on population growth of species *i*; here *i*, *j* $\in\{1,2\}$. Actually, The Lotka-Volterra model of inter-specific competition also includes the effects of intra-specific competition on population of the species. When $\alpha_t^{(ij)}$ or $N_t^{(i)}$ equals 0, the population of the species *j* will grow according to the logistic growth model which models the intra competition between individuals in a species.

These equations can be used to predict the outcome of competition over time. To do this, we should determine equilibria, i.e. the condition that population growth of both species will be zero. Let $dN_t^{(1)}/dt=0$ and $dN_t^{(2)}/dt=0$. If $r^{(1)}N_t^{(1)}$ and $r^{(2)}N_t^{(2)}$ do not equal 0, we get two line equations which are called the isoclines of the species. They can be plotted in four cases, as are shown in Fig.1. According to the figure, there are four kinds of competition results determined by the relationship between $K_t^{(1)}$, $K_t^{(2)}$, $\alpha_t^{(12)}$ and $\alpha_t^{(21)}$.

(a) When $K_t^{(2)}/\alpha_t^{(21)}<K_t^{(1)}$, $K_t^{(1)}/\alpha_t^{(12)}>K_t^{(2)}$ for all the time steps, species 1 will always win and the balance point is $N_t^{(1)}=K_t^{(1)}, N_t^{(2)}=0$.

(b) When $K_t^{(2)}/\alpha_t^{(21)}>K_t^{(1)}, K_t^{(1)}/\alpha_t^{(12)}<K_t^{(2)}$ for all the time steps, species 2 will always win and the balance point is $N_t^{(1)}=0, N_t^{(2)}=K_t^{(2)}$.

(c) When $K_t^{(2)}/\alpha_t^{(21)}<K_t^{(1)}, K_t^{(1)}/\alpha_t^{(12)}<K_t^{(2)}$ for all the time steps, they can win each other; the initial population of them determines who will win.

(d) When $K_t^{(2)}/\alpha_t^{(21)}>K_t^{(1)}, K_t^{(1)}/\alpha_t^{(12)}>K_t^{(2)}$ for all the time steps, there is only one balance point and they can coexist with their own population size.

For an environment that includes Ω species, the competition equation can be modified as:

$$\frac{dN_t^{(i)}}{dt}=r^{(i)}N_t^{(i)}(1-\frac{N_t^{(i)}+\sum_{j=1,j\neq i}^{\Omega}\alpha_t^{(ij)}N_t^{(j)}}{K_t^{(i)}}) \tag{9}$$

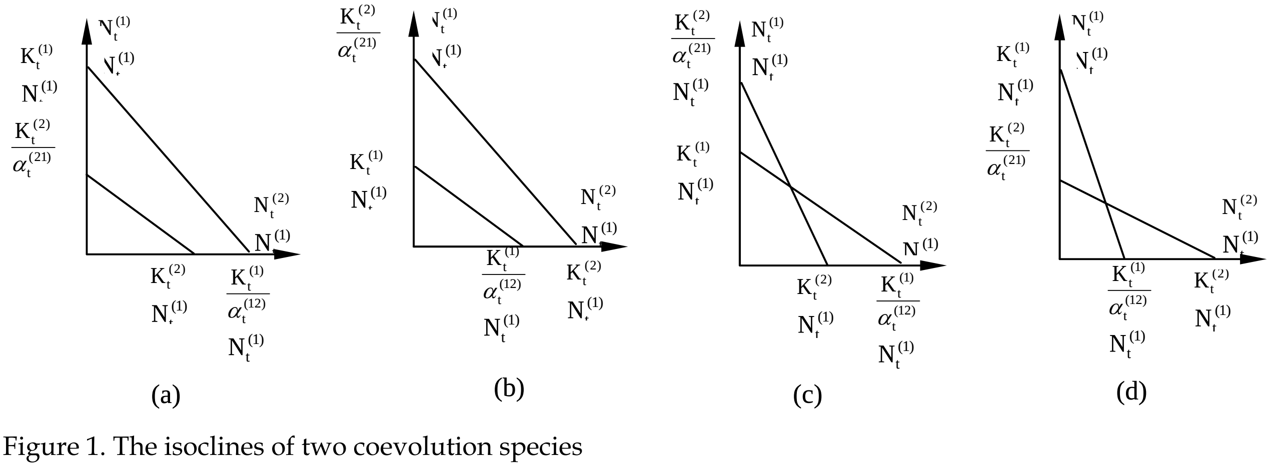


Figure 1. The isoclines of two coevolution species

3.3 Environment Resources

Each species will occupy a part of the state space, which is called living domain of that species. Let matrix $Q_t^{(i)}$ represent the covariance matrix calculated using the individuals in a species *i*. $Q_t^{(i)}$ is a symmetric matrix of $n\times n$, here *n* is the dimension of the state.

Matrix $Q_t^{(i)}$ can be decomposed using singular value decomposition:

$$Q_t^{(i)}=UDU^T \tag{10}$$

$$U=(e_1,\cdots,e_n)\in R^{n\times n} \tag{11}$$

$$D = \mathrm{diag}(d_1, \cdots, d_n) \tag{12}$$

Where d_j is the variance of species *i* in direction e_j, $e_j \in R^n$ is a unit vector, and

$$e_j^T \cdot e_k = \begin{cases} 1 & j = k \\ 0 & j \neq k \end{cases} \tag{13}$$

We define the living domain of the species *i* at time *t* to be an ellipsoid with radius of $2\sqrt{d_j}$ in direction e_j, and it is centered at $\bar{x}_t^{(i)}$ which is the weighted average of the individuals in species *i*. The size of the living domain is decided by:

$$A_t^{(i)} = 2^n C_n \prod_{j=1}^{n} d_j \tag{14}$$

Where C_n is a constant, depending on *n*, for example $C_2 = \pi$, $C_3 = 4\pi/3$. Actually the living domain of a species reflects the uncertainty of the robot's pose estimated according to that species, and it is similar to the uncertainty ellipsoid (Herbert et al., 1997).

Environment resources are proportional to the size of the living domain. The environment resources occupied by a species are defined as:

$$R_t^{(i)} = \begin{cases} \delta \cdot A_t^{(i)} & A_t^{(i)} > \varepsilon \\ \delta \cdot \varepsilon & A_t^{(i)} \leq \varepsilon \end{cases} \tag{15}$$

Where δ is the number of resources in a unit living domain, and ε is the minimum living domain a species should maintain. Assuming a species can plunder the resources of other species through competition, i.e. the environment resources are shared by all species. And the number of individuals that a unit of resource can support is different for species with different fitness. The upper limit of population size that the environment resources can support of a species is determined by:

$$K_t^{(i)} = \exp(1 - \bar{w}_t^{(i)}) \cdot R_t \tag{16}$$

Where parameter $R_t = \sum_{i=1}^{\Omega} R_t^{(i)}$ is the total resources of the system and $\bar{w}_t^{(i)}$ is the average importance factor of species *i*.

It is obvious that the environment resources will change over time. In the beginning, the pose of the robot is very uncertain, so the environment resources are abundant. When the pose of robot becomes certain after running for some time, the living domains will become small and the environment resources will also be reduced. The upper limit of population of species will change according to the environment resources, but the change of the resources will not affect the competition results of the species. Supposing $R_t = \lambda R_{t-1}$ and there are two species, upper limit of the population of species will be $K_t^{(1)} = \lambda K_{t-1}^{(1)}$ and $K_t^{(2)} = \lambda K_{t-1}^{(2)}$. This will not change the relation between $K_t^{(2)} / \alpha_t^{(21)}$ and $K_t^{(1)}$, and that between $K_t^{(1)} / \alpha_t^{(12)}$ and $K_t^{(2)}$ in the Lotka-Volterra competition model.

3.4 Intra-Species Evolution

Since genetic algorithm and sequential Monte Carlo importance sampling have many common aspects, Higuchi (1997) has merged them together. In CEAMCL the genetic operators, crossover and mutation, are applied to search for optimal samples in each species independently. The intra-species evolution will interact with inter-species competition: the evolution of individuals in a species will increase its ability for inter-species competition, so as to survive for a longer time. Because the observation density $p(y_t \mid x_t)$ includes the most recent observed information of the robot, it is defined as the fitness function. The two operators: crossover and mutation, work directly over the floating-points to avoid the trouble brought by binary coding and decoding. The crossover and mutation operator are defined as following:

Crossover: for two parent samples $(x_t^{(p1)}, w_t^{(p1)})$, $(x_t^{(p2)}, w_t^{(p2)})$, the crossover operator mates them by formula (17) to generate two children samples.

$$\begin{cases} x_t^{(c1)} = \xi x_t^{(p1)} + (1-\xi) x_t^{(p2)} \\ x_t^{(c2)} = (1-\xi) x_t^{(p1)} + \xi x_t^{(p2)} \\ w_t^{(c1)} = p(y_t \mid x_t^{(c1)}) \\ w_t^{(c2)} = p(y_t \mid x_t^{(c2)}) \end{cases} \tag{17}$$

Where $\xi \sim U[0,1]$, and $U[0,1]$ represents uniform distribution. And two samples with the largest importance factors are selected from the four samples for the next generation.

Mutation: for a parent sample $(x_t^{(p)}, w_t^{(p)})$, the mutation operator on it is defined by formula (18).

$$\begin{cases} x_t^{(c)} = x_t^{(p)} + \tau \\ w_t^{(c)} = p(y_t \mid x_t^{(p)}) \end{cases} \tag{18}$$

Where $\tau \sim N(0, \Sigma)$ is a three-dimensional vector and $N(0, \Sigma)$ represents normal distribution. The sample with larger importance factor is selected from the two samples for next generation.

In CEAMCL, the crossover operator will perform with probability p_c and mutation operator will perform with probability p_m. Because the genetic operator can search for optimal samples, the sampling process is more efficient and the number of samples required to represent the posterior density can be reduced considerably.

3.5 CEAMCL Algorithm

The coevolution model is merged into the MCL, and the new algorithm is termed coevolution based adaptive Monte Carlo localization (CEAMCL). During localization, if two species cover each other and there is no valley of importance factor between them, they will be merged; and a species will be split if grids occupied by the samples can be split into more than one connected regions as in initial species generation. This is called splitting-merging process. The CEAMCL algorithm is described as following:

(1) Initialization: select initial samples and cluster the samples into Ω species; for each species *i*, let $dN_0^{(i)} / dt$ =0; and let *t* =1.

(2) Sample size determination: if $dN_{t-1}^{(i)} / dt > 0$, draw $dN_{t-1}^{(i)} / dt$ samples randomly from the living domain of the *i*th species and merge the newly drawn samples to $S_{t-1}^{(i)}$; let $N_t^{(i)} = \max(N_{t-1}^{(i)} + dN_{t-1}^{(i)} / dt, 0)$

(3) Resampling: for each species *i*, resample $N_t^{(i)}$ samples from $S_{t-1}^{(i)}$ according to $w_{t-1}^{(i)}$;

(4) Importance Sampling: for each species *i*, sample state $x_t^{(ij)}$ from $p(x_t \mid x_{t-1}^{(ij)}, u_{t-1})$ for each of the $N_t^{(i)}$ possible state $x_{t-1}^{(ij)}$; and evaluate the importance factor $w_t^{(ij)} = p(y_t \mid x_t^{(ij)})$.

(5) Intra-species evolution: for each species *i* randomly draw two samples, and mate them with probability p_c, repeat this for $N_t^{(i)} / 2$ times; then randomly draw one sample from species *i*, and mutate it with probability p_m, repeat this for $N_t^{(i)}$ times.

(6) Splitting-merging process: split and merge the species using the rules defined in the splitting-merging process.

(7) Calculating the sample size increment: for each species i calculate the upper limit population size of species *i*, and calculate $dN_t^{(i)}/dt$ using equation (9).

(8) Summary: The species whose average importance factor is the largest is assumed to be the location of the robot, and the importance factors of each species are normalized independently.

(9) *t*=*t*+1; go to step 2) if not stop.

3.6 Computational Cost

Compared with MCL, CEAMCL requires more computation for each sample. But the sampling process is more efficient in CEAMCL. So it can considerably reduce the number of samples required to represent the posterior density.

The resampling, importance factor normalization and calculating statistic properties have almost the same computational cost per sample for the two algorithms. We denote the total cost of each sample in these calculations as T_r. The importance sampling step involves drawing a *n* dimensional-vector according to the motion model $p(x_t \mid x_{t-1}^{(j)}, u_{t-1})$ and computing the importance factor whose computational costs are denoted as T_s and T_f respectively. The additional computational cost for CEAMCL arises from the intra-species evolution step and the splitting-merging process. The evolution step computes the importance factor with probability p_c in crossover and with probability p_m in mutation. The other computation in evolution step includes drawing a *n* dimensional-vector from a Gaussian distribution, drawing a random number from uniform distribution of *U*[0,1] and several times of simple addition, multiplication and comparison. Since the other computational cost in evolution step is almost the same as T_s which also includes drawing a *n* dimensional-vector from a Gaussian distribution, the total computational cost for each sample in evolution step is approximated by $(p_c + p_m)T_f + T_s$. The splitting or merging probability of species in each time step is small, especially when the species become stable no species need to be split or merged, so the computational cost of the splitting-merging process denoted as T_m is small. And the computational costs of other steps in CEAMCL are not related to the sample size, so they can be neglected. Defining N_M and N_C as the number of samples in MCL and CEAMCL respectively, the total computational costs for one of the iteration in localization T_M and T_C are given by:

$$T_M = N_M (T_f + T_s + T_r) \tag{19}$$

$$T_C \approx N_C ((1+p) \cdot T_f + 2T_s + T_r + T_m) \tag{20}$$

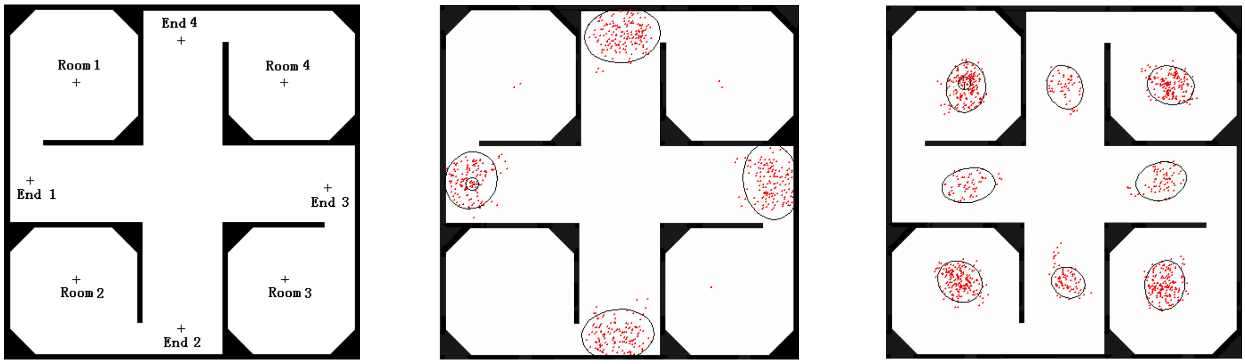
Where $p = p_c + p_m$. The most computationally intensive procedure in localization is the computation of the importance factor which has to deal with the high dimensional sensor data, so T_f is much larger than the other terms. It is safe to draw the following rule:

$$T_C \approx (1+p) \cdot T_M (N_C / N_M) \tag{21}$$

4. Experimental Results

We have evaluated CEAMCL in the context of indoor mobile robot localization using data collected with a Pioneer2 robot. The data consist of a sequence of laser range-finder scans along with odometry measurements annotated with time-stamps to allow systematic real-time evaluations. The experimental field is a large hall in our laboratory building whose size is of 15 × 15 m², and the hall is partitioned into small rooms using boards to form a highly symmetric map shown in Fig 2(a).

In the initial species generation, we only use the x-y position and don't use direction to cluster the samples. The map is divided into 150 × 150 grids. The test sample size N_{test} equals 1000000; the threshold parameter μ equals 0.85; and parameter η is 2. The initial species generation results are shown in Fig 2. It is obvious that even in the same environment the initial sample sizes are different when the robot is placed in different places. The real position of robot is marked using a small circle.



(a) (b) (c)

Figure 2. The initial species for localization. (a) The map (b) 4 species with 537 samples (c) 8 species with 961 samples

In the localization experiments, the robot was placed at the center of one of the four rooms, and it was commanded to go to a corner of another room. 5 times of experiments were conducted for each center. The three algorithms MCL, GMCL and CEAMCL were applied to localize the robot using the data collected in the experiments. Here GMCL is genetic Monte Carlo localization which merges genetic operators into MCL but without coevolution mechanism. The parameter $\alpha_t^{(ij)} = \overline{w}_t^{(j)} / \overline{w}_t^{(i)}$, where $\overline{w}_t^{(i)}$ is the fitness (average importance factor) of species *i*; parameter $r^{(i)}$, the maximum possible rate of population growth of species, equals 0.2; and parameter ε, the minimum living domain of species, equals 0.5m²; parameter δ, the number of resources in a unit living domain which is 1 m² in this paper, equals 80; the crossover probability p_c is 0.85; and the mutation probability p_m is 0.15. The success rate of the three algorithms to track the hypothesis corresponding to the robot's real pose until the robot reaches the goal is shown in table 1. In table 1, the converged time is the time when the samples converged to the four most likely positions; the expired time is the time when one of the hypotheses vanished. The data shows that the CEMCL can converge to the most likely positions as fast as GMCL, but it can maintain the hypotheses for a much longer time than the other two algorithms. This is because the species with much lower fitness will die out because of the competition between species, and the species with similar fitness can coexist with each other for a longer time.

	CEAMCL	GMCL	MCL
Converged time (s)	6. 8	6.4	9.7
Expired time (s)	Never	30.4	36.4
Success rate	100%	82%	86%

Table 1. Success rate of multi-hypotheses tracking

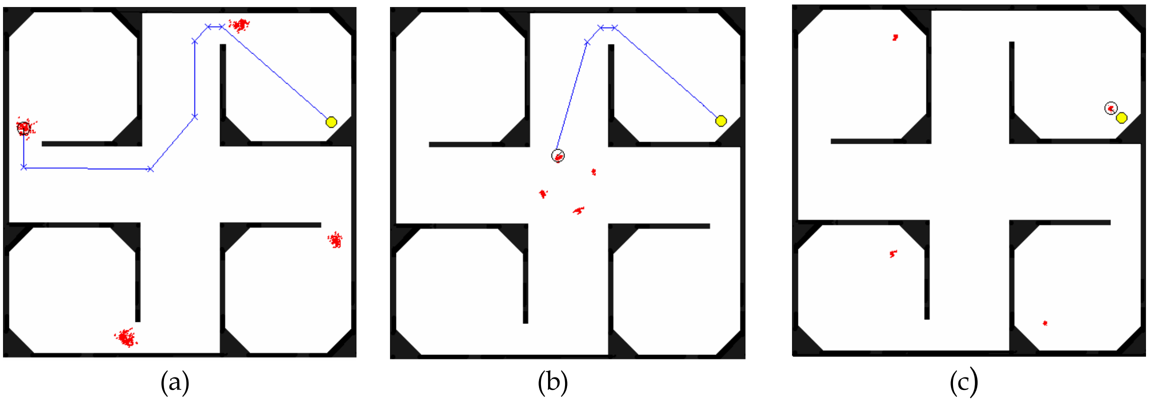


Figure 3. Localization based on CEAMCL. (a) samples at 7th second (b) samples at 16th second (c) Samples at 40th second

Figure 3 shows three inter moments when the robot ran from the center of room 1 to the goal using the CEAMCL algorithm.

To compare the localization precision of the three algorithms, we use the robot position tracked by using MCL with 5000 samples in the condition of knowing the initial position to be the reference robot position. The average estimation errors along the running time are shown in Fig 4. Since the summary in CEAMCL is based on the most likely species and the genetic operator in intra-species evolution can drive the samples to the regions with large importance factors, so localization error of CEAMCL is much lower. Although GMCL almost has the same precision as CEAMCL after some time, GMCL is much more likely to produce premature convergence in symmetric environment.

The computational time needed for each iteration with 961 initial samples on a computer with a CPU of PENIUM 800 is shown in Fig 5. Because $p_c + p_m = 1$, the computational time needed for each iteration of CEAMCL is almost twice of that of MCL with the same size of sample set. But since the sample size of CEAMCL is adaptively adjusted during the localization process, the computational time for each iteration of CEAMCL becomes less than that of MCL after some time. From Fig 4. and Fig 5. we can see that the CEAMCL can make precise localization with a small sample size. The changes of the total environment resources and the total number of samples are shown in Fig 6. From the figure we can see that the resources will be reduced when the position becomes certain, the total sample size needed for robot localization will also be reduced.

Large δ will reduce the competition between the species since there is enough resource for them. The curve of total sample size with different δ is shown in Fig 7.

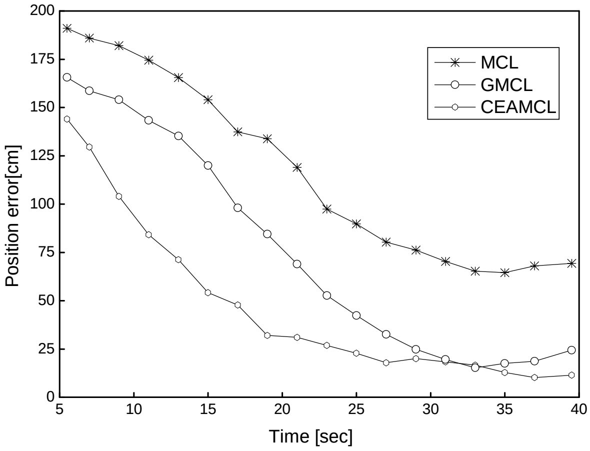


Figure 4. Estimation Error

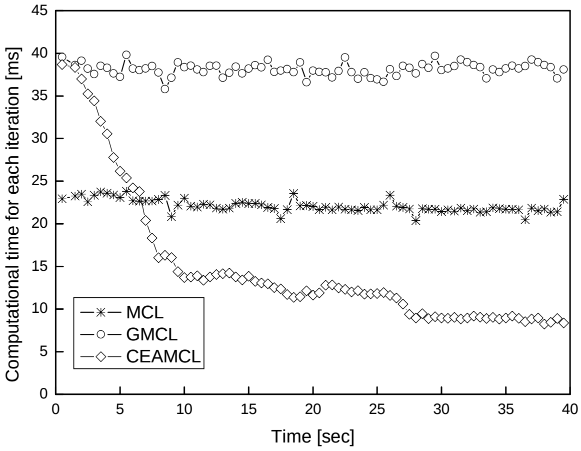


Figure 5. Computational time for each iteration

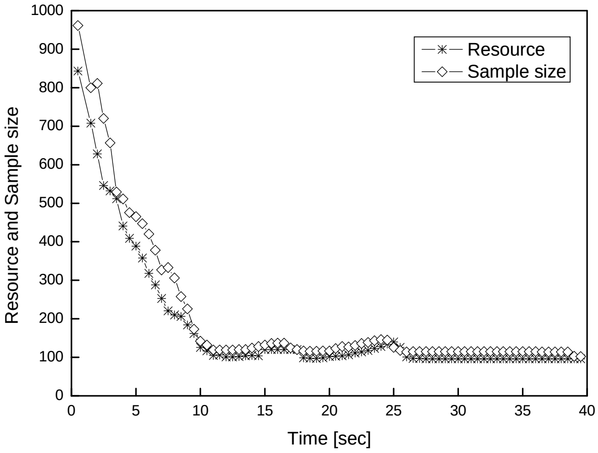


Figure 6. Change of resource and sample size

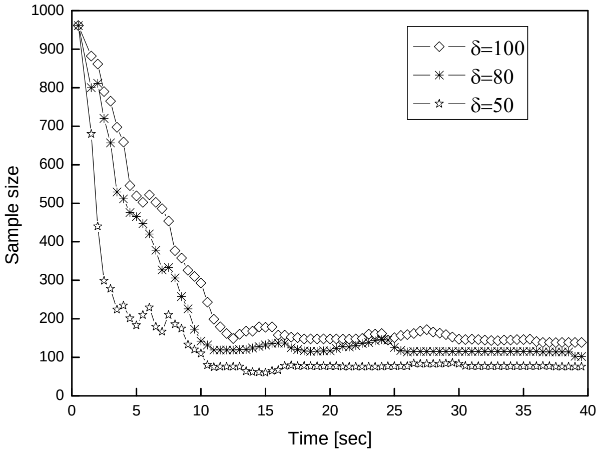


Figure 7. Effect of δ on sample size

The growth rate $r^{(i)}$ is another important parameter. Large $r^{(i)}$ will increase the rate of convergence to the species that have larger fitness. But if $r^{(i)}$ is too large it may cause premature convergence.

5. Conclusions

An adaptive localization algorithm CEAMCL is proposed in this paper. Using an ecological competition model, CEAMCL can adaptively adjust the sample size according to the total environment resource, which represents uncertainty of the position of the robot. Coevolution between species ensures that the problem of premature convergence when using MCL in highly symmetric environments can be solved. And genetic operators used for intra-species evolution can search for optimal samples in each species, so the samples can represent the desired posterior density better. Experiments prove that CEAMCL has the following advantages: (1) it can adaptively adjust the sample size during localization; (2) it can make stable localization in highly symmetric environment; (3) it can make precise localization with a small sample size.

6. References

Andrieu, C. & Doucet, A. (2002). Particle filtering for partially observed gaussian state space Models. *Journal of the Royal Statistical Society Series B (Statistical Methodology)*, Vol. 64, No. 4, pp.827-836, ISSN 00359246

Angeline, P. J. & Pollack, J. B. (1993). Competitive environments evolve better solutions for complex tasks, *Proceedings of of the Fifth International Conference on Genetic Algorithms (ICGA-1993)*, Forrest, S. (Ed.), pp. 264-270, ISBN 1558602992 , Morgan Kaufmann Pulbishers, California, USA

Burgard, W., Fox, D., Hennig, D., & Schmidt, T. (1996). Estimating the absolute position of a mobile robot using position probability grids, *Proceedings of the National Conference on Artificial Intelligence (AAAI-1996),* pp. 896-901, ISBN 0-262-51091-X, AAAI Press, Oregon, USA

Chen, M. & Zalzala, A. (1995). Safety considerations in the optimization of paths for mobilerobots using genetic algorithms, *Proceedings of First International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications*, pp. 299-304, ISBN 0852966504, IEE Press, Halifax, UK

Dellaert, F., Fox, D., Burgard, W., & Thrun, S. (1999). Monte Carlo localization for mobile robots, *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 1322-1328, IEEE Press, Michigan, USA

Duckett, T. (2003). A genetic algorithm for simultaneous localization and mapping, *Proceedings of the IEEE International Conference on Robotics and Automation*, pp, 434-439, IEEE Press, Taiwan, China

Ficici, Sevan G. & Pollack, Jordan B. (2001). Pareto Optimality in Coevolutionary Learning, *Proceedings Sixth European Conference on Artificial Life*, Eleonora, B. et al (Ed.), pp. 316-325, ISBN 3-540-42567-5, Springer Press, Prague, Czech Republic

Fox, D. (2003). Adapting the sample size in particle filters through KLD-Sampling. *International Journal of Robotic Research*, Vol. 22 No. 12, pp. 985-1004, ISSN 0278-3649

Herbert, M., Thorpe, C. & Stenz, T. (1997). *Autonomous navigtion research at Carnegie Mellon*. Kluwer Academic Publisher, ISBN 0792398335

Higuchi, T. (1997). Monte Carlo Filtering using genetics Algorithm Operators. *Journal of Statistical Computation and Simulation* 59, pp. 1-23, ISSN 0094-9655

Jensfelt, P. & Kristensen, S. (2001). Active global localization for a mobile robot using multiple hypothesis tracking. *IEEE Trans. Robotics Automation*, Vol. 17, No. 5, pp. 748-760, ISSN 1042-296X

Kaelbling, L. P., Cassandra, A. R., & Kurien, J. A. (1996). Acting under uncertainty: Discrete Bayesian models for mobile-robot navigation, *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 963-972, ISBN 0-7803-3213-X, IEEE Press, Osaka, Japan

Koller, D. & Fratkina, R. (1998). Using learning for approximation in stochastic processes, *Proceedings of the International Conference on Machine Learning*, Lorenza Saitta (Ed.), pp. 287-295, ISSN 1-55860-556-8, Morgan Kaufmann Press, Bari, Italy

Leonard, J.J. & Durrant-Whyte, H.F. (1991). Mobile robot localization by tracking geometric beacons. *IEEE Trans. Robotics Automation*, Vol. 7, No. 3, pp. 376-382

Milstein, A., Sanchez, J. N., & Wiamson, E. (2002). Robust global localization using clustered particle filtering, *Proceedings of the National Conference on Artificial Intelligence (AAAI-2002)*, pp. 581-586, ISBN 0-262-51129-0, AAAI Press, Edmonton, Canada

Moreno, L., Armingol, J. M., & Garrido, S., et al. (2002). A genetic algorithm for mobile robot localization using ultrasonic sensors. *Journal of Intelligent and Robotic Systems* 34, pp. 135-154, ISSN 0921-0296

Moriarty, D. E. & Miikkulainen, R. (1997). Forming neural networks through efficient and adaptive coevolution. *Evolutionary Computation*, Vol. 5, No. 4, pp. 373-399, ISSN 1063-6560

Potter, M. A. & De Jong, K. A. (2000). Cooperative coevolution: An architecture for evolving coadapted subcomponents. *Evolutionary Computation*, Vol. 8, No. 1, pp. 1-29, ISSN 1063-6560

Potvin, J., Duhamel, C., & Guertin, F. (1996). A genetic algorithm for vehicle routing with backhauling. *Applied Intelligence* 6, pp. 345-355, ISSN 0924-669X

Rosin, C., & Belew, R. (1997) New methods for competitive co-evolution. *Evolutionary Computation*, Vol. 5, No. 1, pp. 1-29, ISSN 1063-6560

Roumeliotis, S. I. & Bekey, G. A. (2000). Bayesian estimation and Kalman filtering: A unified framework for mobile robot localization, *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 2985-2992, ISBN 0-7803-5889-9, IEEE Press, S. Francisco, USA

Serra, J. (1982). *Image Analysis and Mathematical Morphology*. Academic Press, New York

Simmons, R. and Koenig, S. (1995). Probabilistic robot navigation in partially observable environments, *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 1080-1087, ISBN 1558603638 ,Morgan Kaufmann Press, Quebec, Canada

Thrun, S., Fox, D., Burgard, W. & Dellaert, F. (2001). Robust Monte Carlo localization for mobile robots. *Artificial Intelligence* 128, pp. 99-141, ISSN 0883-9514

Yuchang, S. & Xiaoming, C. (1996). *Common Ecology*, Beijing University Press (in Chinese), ISSN 7-301-01796-0Q52, Beijing

Autonomous Navigation of Unmanned Vehicles: A Fuzzy Logic Perspective

Nikos C. Tsourveloudis, Lefteris Doitsidis & Kimon P. Valavanis

1. Introduction

Unmanned robotic vehicles are capable of performing desired tasks in unstructured, uncertain and potentially hostile environments. They may be remotely-operated or function *(semi-) autonomously* without human intervention. However, it will long before unmanned robot vehicles function as completely autonomous entities in diverse environments. Current unmanned vehicles adhere to different levels of *autonomicity* as defined by existing technology limitations and used sensors. Important operational characteristics related to unmanned vehicle functionality (*aerial, aquatic* or *terrestrial*), include the following:

Perception: *Acquire and use knowledge about the environment and itself.* This is done by taking measurements using various sensing devices and then extracting meaningful information that should be used in all later tasks (such as *localization, planning, collision free motion control, recharging,* etc).

Intelligence: *Operate for a considerable time period without human intervention*. This is associated with the learning and inference capabilities, which of the vehicle should have to be able to adapt (its behavior or/and shape) to the environment.

Action: Travel from point A to point B. The vehicle should utilize predefined and acquired knowledge to move in dynamic environments without involving humans in the navigation loop.

In robotics, autonomy is mainly associated with navigation issues. From a conceptual point of view, autonomous navigation of robotic vehicles may be achieved via continuous interaction between *perception, intelligence* and *action*, as shown in Figure 1.

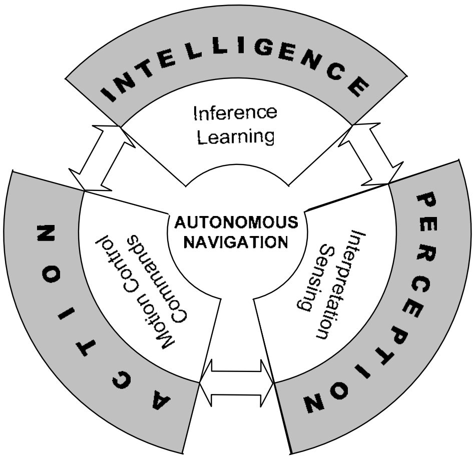


Figure 1. Autonomous navigation conceptual loop

Navigation of autonomous robotic vehicles in obstacle filled dynamic environments requires derivation and implementation of efficient real-time sensor based controllers. Effective control algorithms for autonomous navigation, should imitate the way humans are operating manned or similar vehicles. Considering the environment uncertainty that is difficult if not impossible to model, fuzzy logic is one of the most widely used mathematical tools for autonomous vehicle navigation (Driankov & Saffiotti, 2001). Fuzzy logic techniques have already been used and are being used currently for autonomous navigation of *ground* (indoors (Aguire & Gonzalez, 2000; Doitsidis, et al., 2002; Goodridge & Luo, 1994; Ishikawa, 1991; Li, 1994; Oriolo, et al., 1998; Pin & Watanabe, 1994; Tsourveloudis, et al., 2001; Tunstel, 1996) and outdoors (Hagras, et al., 2001; Seraji & Howard, 2002; Valavanis, et al., 2005), *aerial* (fixed (Doitsidis, et al., 2004; Nikolos, et al., 2003a) and rotary wings (Amaral, 2001; Hoffmann, et al., 1999; Kadmiry & Driankov, 2001; Kadmiry & Driankov, 2004; Sugeno, 1999) and *water* (surface (Vanick, 1997) or submersible (Kanakakis, et al., 2004; Kanakakis, et al., 2001)) robotic vehicles.

The wide applicability of fuzzy logic in autonomous navigation is mainly based on suitable knowledge representation of inherently vague notions achieved through fuzzy IF-THEN rules. These rules typically contain linguistic information, which describes the problem at hand very simple and fast. Further, in the majority of fuzzy logic application in navigation, a mathematical model of the dynamics of the vehicle is not needed in the design process of the motion controller. Only the problem-specific heuristic control knowledge is needed for the inference engine design. From a more practical point of view, fuzzy logic is the most appropriate modeling tool for representing imprecision and uncertainty of the sensor readings. Another reason that explains the popularity of fuzzy logic in autonomous navigation is the low computation time of the hardware implementations of fuzzy controllers which favors real-time applications.

This chapter presents implementations of fuzzy logic in the area of autonomous vehicles navigation. It discusses successful applications of collision free motion control of *ground*, *aerial* and *underwater* unmanned vehicles navigation. The common characteristic in all applications regardless of the type of vehicle is the navigation architecture used. This generic concept of fuzzy navigation architecture is discussed in the next section. Section 3 presents the implementation of the proposed generic architecture for ground, aerial and underwater robots. The chapter concludes with future research trends for unmanned vehicles.

2. Navigation Architecture

In the literature, the navigation problem is separated in two parts: *global* navigation concerned with generating a path leading to the goal point; and *local* navigation, which follows the global path avoiding collisions with obstacles. The solutions presented with the use of fuzzy logic fall more or less in the second category. (Saffiotti, 1993) discuss the problem of mixing the two essential navigation behaviors, that is, pure reactive and goal-oriented behaviors, by using fuzzy logic. Generally speaking, the approaches to fuzzy navigation in dynamic environments follow either a classical paradigm or a behavior-based paradigm. Fuzzy navigation schemes, which follow the classical paradigm, have one set of control rules that includes all situations that may arise. All rules operate at all times to generate the control law. Behavior based fuzzy navigation acknowledges that there are different types of behaviors which the autonomous vehicle must exhibit in different situations. Each behavior is given a set of rules and an inference engine is used to determine which behavior (or combination of behaviors) needs to be invoked in the current situation. In both paradigms, the "reaction" is given by a set of rules, which describe the navigation priorities.

Fuzzy inference approaches tend to de-emphasize goal-directed navigation and focus more upon handling reactive and reflexive cases. The results of the fuzzy inference controllers generally do not tend towards optimal paths. However, surprise obstacles and rapidly moving obstacles are handled with more certainty compared to methodologies in which certain performance criteria should be optimized (Tsourveloudis, et al., 2001).

Regardless of the final navigation goal or the type of vehicles, some kind of sensor data management is needed. Sensor readings provide information about the environment and the vehicle itself. These readings are almost at all times erratic, incomplete or conflicting and should be further processed in order to provide meaningful information. This information is essential for the motion commands of the vehicle. The overall architecture of the proposed navigation schema is shown in Fig. 2.

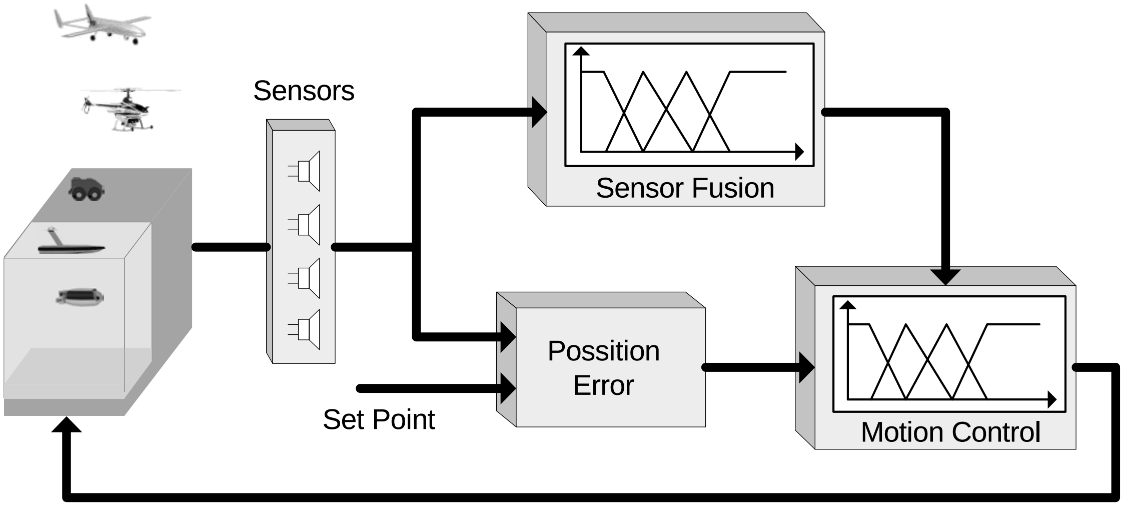


Figure 2. Architecture of the fuzzy logic navigation scheme

The *sensor fusion* module is a fuzzy logic controller which takes as input the data provided by the various sensors and delivers information for eventual obstacles in respect to vehicle's position and orientation. The interpreted obstacle information forms a collision possibility, which is send to the *motion control* module. The collision possibility together with position and/or orientation error are inputs of the motion fuzzy controller, which is responsible for the output commands to the driving devices. In further details, the first layer of the fuzzy logic inference engine performs sensor fusion from sensor readings, providing information about potential collisions in four directions, and the second layer guarantees collision avoidance with dynamic (moving) obstacles while following the desired trajectory. It has been shown (Tsourveloudis, et al., 2001), (Nikolos, et al., 2003b) that a path planning algorithm can be easily incorporated in the generic navigation architecture shown in Fig. 2

The architecture presented in Fig. 2 has been successfully applied to various unmanned vehicles as it is described in the following sections. In all these applications the basic idea of the layered fuzzy control is utilized in respect to the control demands of each robotic vehicle.

3. Physical Implementation

Some of the most difficult applications for robotics lie outdoors in dangerous and unknown environments. These include applications such as search and rescue (in land and sea), surveillance, humanitarian demining, underwater construction and mapping, environment monitoring, meteorology, agriculture and defense. Autonomous navigation of unmanned vehicles in unstructured environments is a multidiscipline and attractive challenge for researchers from academia and industry. The presentation that follows describes state-of-the-art applications of fuzzy logic that follow the architecture presented in the previous section. For most of the cases presented MATLAB have been used running either in Linux or Windows.

3.1 Ground Vehicles

The proposed navigation scheme was initially implemented on the *Nomad 200* mobile robot platform (Tsourveloudis, et al., 2001), for indoor navigation, and later on an ATRV skid steering mobile robot manufactured by *iRobot* (Doitsidis, et al., 2002).

The mobile robot ATRV-mini (shown in Fig. 3) has a ring of *24* sonar sensors that are placed around the vehicle and grouped in pairs A_i, $i=1,\ldots,12$, as shown in Fig. 3, each with an actual maximum measurement range of *2m*, returning data readings every *0.1s*. The data used for localization and calculation of the heading were produced from an odometer.

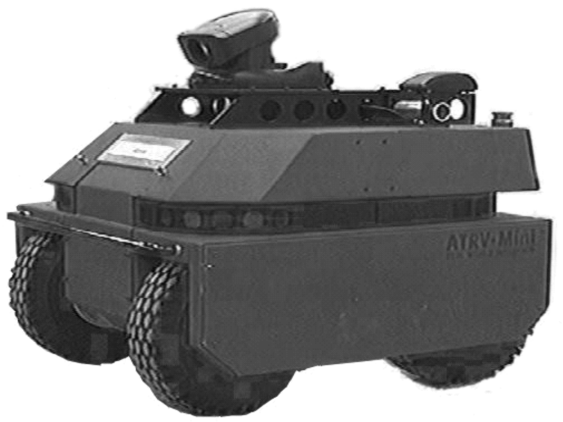


Figure 3. DAMON: The ATRV - mini of the Intelligent Systems and Robotics Laboratory, Technical University of Crete

For this robot system, a two-layer Mamdani-type controller has been designed and implemented (Doitsidis, et al., 2002). In the first layer, there are four fuzzy logic controllers responsible for obstacle detection and collision possibility calculation in the four main directions, ***front***, ***back***, ***left***, ***right***. The four controllers receive as inputs sonar sensor data and return as output the respective direction collision possibility. Data from group sensors $A_1, A_2,\ldots, A_5$ (5 inputs) and group sensors $A_7, A_8,\ldots, A_{11}$ (5 inputs) serve as inputs to the individual controllers responsible for the calculation of the ***front*** and ***back*** collision possibilities, respectively. Data from group sensors A_5, A_6, A_7 (3 inputs) and group sensors A_{11}, A_{12}, A_1 (3 inputs) serve as inputs to calculate the ***left*** and ***right*** possibilities, respectively (Fig. 4).

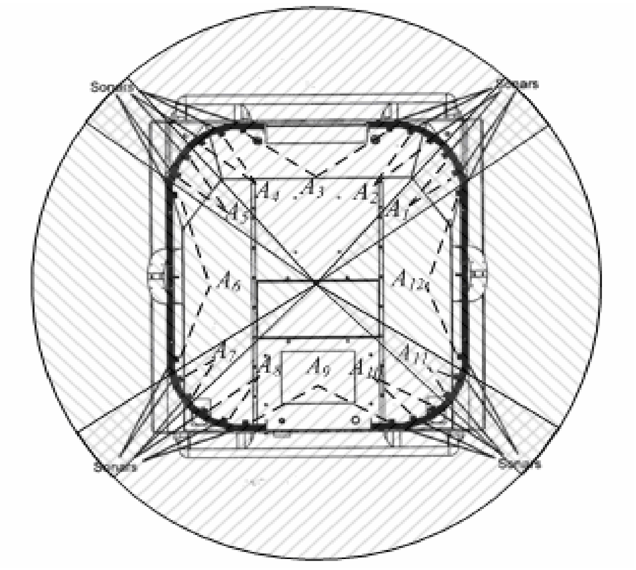


Figure 4. Sonar grouping for the ATRV-mini

The individual fuzzy controllers utilize the same membership functions to calculate the collision possibilities.

Collision possibilities are calculated using fuzzy rules of the type:

R: IF d_i *is* <$LD^{(k)}$> AND d_{i+1} *is* <$LD^{(k)}$> THEN c_j is <$LC^{(k)}$>,

where k is the rule number, d_i represents sensors group i minimum readings, $LD^{(k)}$ is the linguistic variable of the term set D ={*near, medium_distance, away*}, c_j is the collision direction and $LC^{(k)}$ the variable with term set C={*not_possible, possible, high_possibility*}.

The overall output of the first layer is calculated using the *max-min* composition between the fuzzified readings:

$$\mu_C^*(c_j) = \max \min_{d_i}[\mu_D^*(d_i), \mu_{R^{(k)}}(d_i, c_j)], \qquad (1)$$

where $\mu_D^*(d_i)$ is the minimum of the fuzzified sonar readings and $\mu_{R^{(k)}}$ is the mathematical expression of the kth navigation rule.

The input variables to the second layer fuzzy controller are: a) the four *collision possibilities* with linguistic values {*not_possible, possible, high_possibility*; b) the *angle error* with linguistic values {*Backwards_1, Hard_Left, Left, Left2, Left1 Ahead, Right1, Right2, Right, Hard_Right, Backwards_2.*} The *angle_error* takes values from –180° to 180° and it is the difference between the desired and the actual heading of the vehicle.

The output variables are: a) *translational_velocity* with linguistic variables {*back_full, back_normal, back_slow, stop, front_slow, front_normal, front_full*} b) *rotational_velocity* with linguistic variables {*right_full, right, right1, no_rotation, left1, left, left_full*}. The *translational velocity* takes values that range from –1.5m/sec to 1.5 m/sec. The *rotational velocity* takes values ranging from –3rad/sec to 3 rad/sec. The number of linguistic values for the angle error, translational and rotational velocities is chosen after conducting several experiments to ensure smooth and accurate collision free navigation.

If the value of the translational velocity is positive the vehicle moves forward; if it is negative the vehicle moves backwards. A positive rotational velocity results in vehicle turn left; a negative value in vehicle turn right. Navigation and collision avoidance are performed using rules of the type:

IF c_j is $LC^{(k)}$ AND θ is $L\Theta^{(k)}$ THEN tv is $LTV^{(k)}$ AND rv is $LTV^{(k)}$,

where k is the rule number, c_j is collision of type j, i.e., the output of the obstacle detection module, θ is the angle error, tv is the translational velocity and rv is the rotational velocity. $LC^{(k)}$, $L\Theta^{(k)}$, $LTV^{(k)}$, $RTV^{(k)}$ are the linguistic variables of c_j ,θ, tv, rv respectively. AND = min in all rules. The generic mathematical expression of the k^{th} navigation rule is:

$$\mu_{R^{(k)}}(c_j,\theta,tv,tr)=\min[(\mu_{LC^{(k)}}(c_j),\mu_{L\Theta^{(k)}}(\theta),\mu_{LTV^{(k)}}(tv),\mu_{LTR^{(k)}}(tr)]. \quad (2)$$

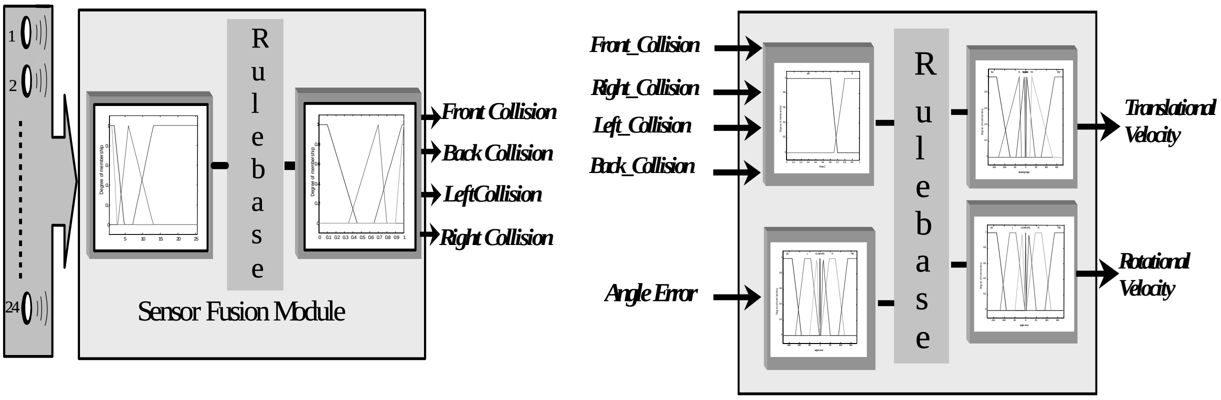
The overall navigation output is given by max-min composition:

$$\mu^{*}_{N}(tr,tv)=\max\min_{c_j,\theta}[\mu^{*}_{AND}(c_j,\theta),\mu_R(c_j,\theta,tr,tv)], \quad (3)$$

where,

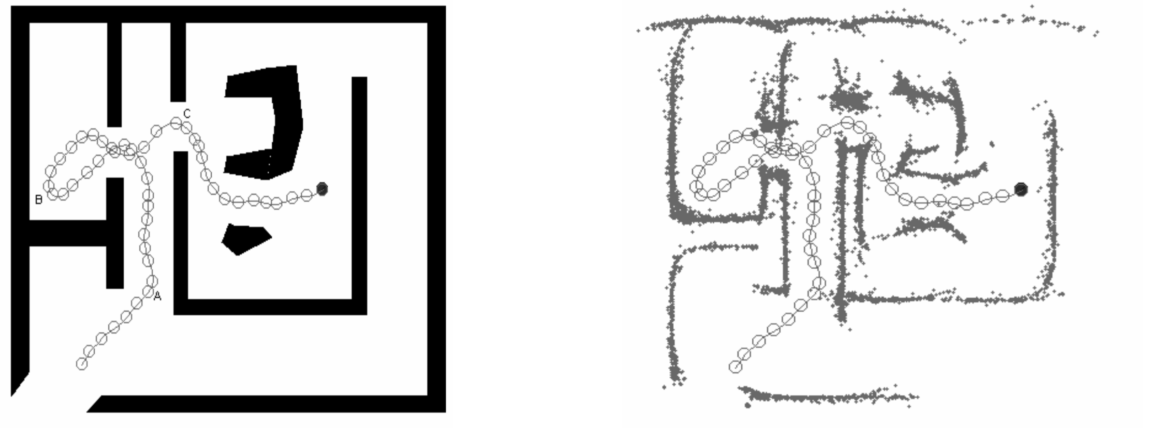
$$\mu_R(c_j,\theta,tv,tr)=\bigcup_{k=1}^{K}\mu_{R^{(k)}}(c_j,\theta,tv,tr). \quad (4)$$

The two layers of the fuzzy logic controller are presented in Fig. 5. A modification of the proposed navigation scheme for outdoors environment was implemented in an ATRV-Jr. robot equipped with a different sensor suite, including a SICK LMS 200 scanning planar laser and a GPS system (Valavanis, et al., 2005).



a) b)

Figure 5. a) The sensor fusion module, b) The motion control module



a) b)

Figure 6. Navigation in a cluttered environment a): Environment Map, b): Aggregation of sonar readings

The implementation have been done in MATLAB running on-board the actual vehicles connected with the *JMatlink class* with Java which was responsible for handling all the processes which were running on the vehicles.

It consists of four modules: the laser range filter, position detection, heading error calculation and the actual fuzzy logic robot controller. The control system receives as inputs laser, odometer and GPS data, as well as a control reference input (next waypoint or goal point). It outputs actuator commands in terms of robot rotational and translational velocity.

The fuzzy logic controller is implemented as a Mamdani-type controller similar to previous work (Tsourveloudis, et al., 2001; Doitsidis, et al., 2002). The fuzzy logic controller rule base includes the fuzzy rules responsible for vehicle control. The inference engine activates and applies relevant rules to control the vehicle. The fuzzification module converts controller inputs into information used by the inference engine. The defuzzification module converts the output of the inference engine into actual outputs for the vehicle drive system.

The fuzzy controller input from the filtered laser range block consists of a three value vector with components related to the distance of the closest object in the left sector of the scan, in the center sector and in the right sector, respectively. The sectors are presented in Fig. 7.

This information is used to calculate three collision possibilities *left*, *center*, *right* reflecting potential static / dynamic obstacles in the robot field of view, similar to the approach followed in (Tsourveloudis, et al., 2001; Doitsidis, et al., 2002) but for outdoor environments. The fourth input to the fuzzy logic controller is the robot's heading error calculated from the robot's current heading and the desired heading.

Implementation wise, each of the three aggregate range inputs includes three trapezoidal membership functions namely, *close*, *medium* and *far*. The input linguistic variables are denoted as *left distance*, *right distance* and *center distance* corresponding the left area, right area and center area sectors. The *heading error* input uses four trapezoidal membership functions and one triangular membership function. They are empirically derived based on extensive tests and experiments. Each distance input variable d_i (corresponding to left area, center area, right area) is fuzzified and expressed by the fuzzy sets C_i, *MD*, A_i referring to *close*, *medium*, and *far*. The range of the membership functions for each d_i is between 0-8 meters.

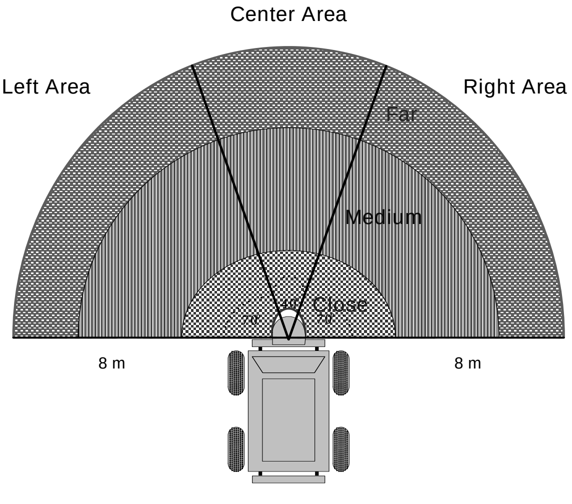


Figure 7. Laser scanner sectors

The input variable *heading error, he,* is fuzzified and expressed by the fuzzy sets *FL, L, AH, R, FR*, referring to *far left, left, ahead, right*, and *far right,* respectively. The range of the membership functions for the *heading error,* is between -180 and 180 degrees.

The fuzzy logic controller has two output variables, *translational velocity* (*tr*) implemented with two trapezoidal and one triangular membership functions, and *rotational velocity* (*rv*) implemented with four trapezoidal membership functions and one triangular membership function.

The output variable *tr* is expressed by the fuzzy sets *ST, SL, F* referring to *stop, slow*, and *fast,* while the output variable *rv* is expressed by the fuzzy sets *HRR, RR, AHR, LR, HLR* referring to *hard right, right, ahead, left, hard left.*

The output commands are normalized in a scale from 0 to 1 for the translational velocity, where 0 corresponds to complete stop and 1 to maximum speed. Rotational velocity output commands are normalized from -1 to 1, where -1 corresponds to a right turn with maximum angular velocity and 1 to a left turn with maximum angular velocity. Each fuzzy rule *j* is expressed as:

IF d_1 is D_{j1} AND d_2 is D_{j2} AND d_3 is D_{j3}AND *he* is HE_j THEN *tr* is TR_j AND *rv* is RV_j;

for *j*=1,…, number of rules. D_{ji}, is the fuzzy set for d_i in the *jth* rule which takes the linguistic value of C_i, *MD, A.* HE_j is the fuzzy set for the *he* which takes the linguistic values *FL, L, AH, R, FR.* TR_j and RV_j are the fuzzy sets for *tr* and *rv* respectively.

The generic mathematical expression of the *jth* navigation rule is given by:

$$\mu_{R(j)}(d_i, he, tr, r) = \min[\mu_{D_{ji}}(d_i), \mu_{HE(j)}(he), \mu_{TR(j)}(tr), \mu_{RV(j)}(rv)]. \tag{5}$$

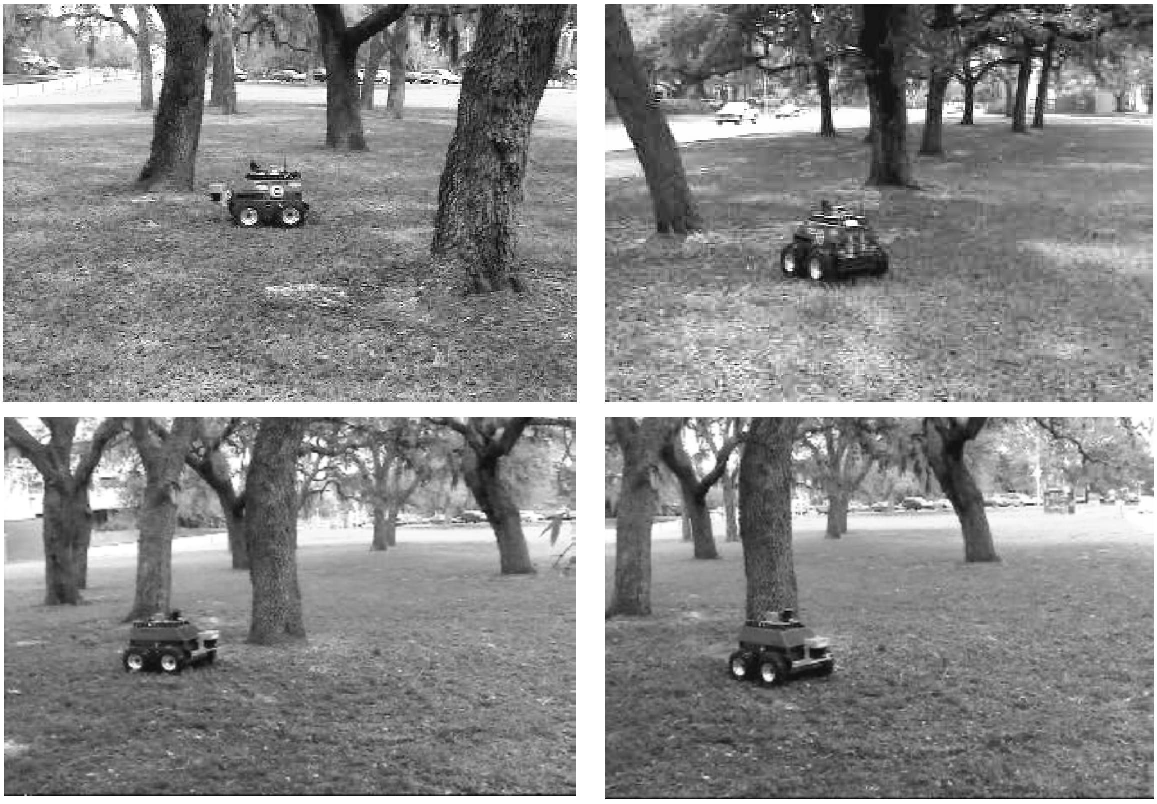


Figure 8. Unmanned Ground Vehicle navigating in an area with trees

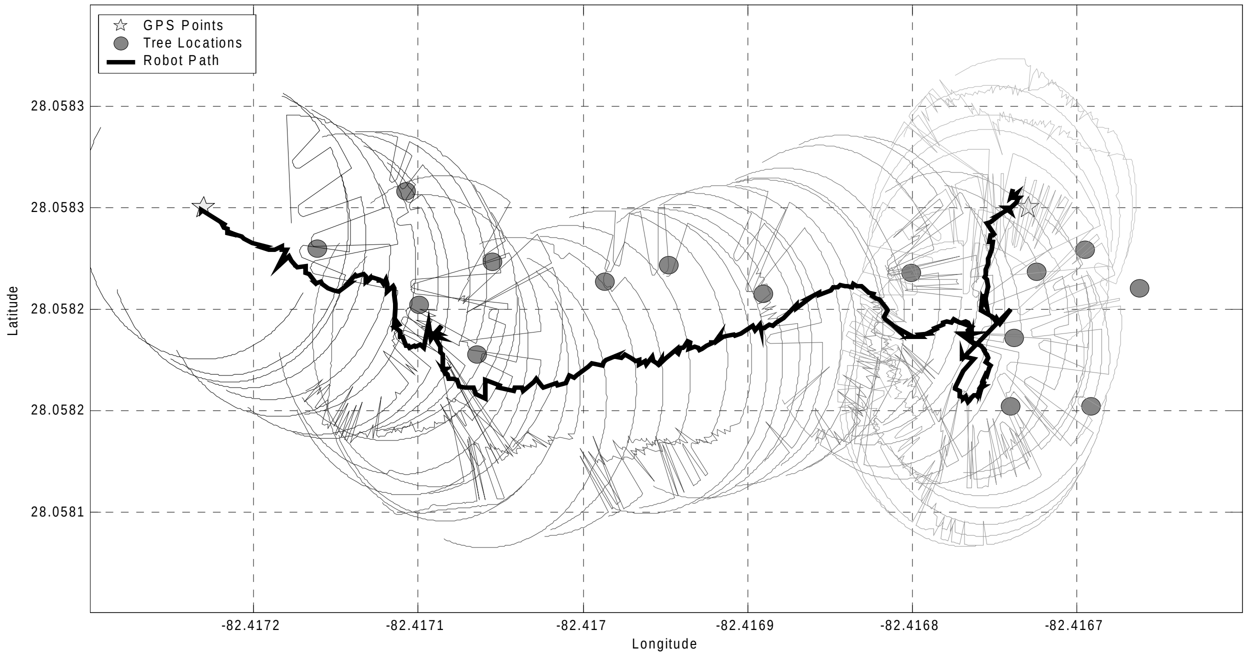


Figure 9. Robot path in an outdoor environment

The overall navigation output is given by the max-min composition and in particular:

$$\mu_{\mathrm{N}}^{*}(\mathrm{tr},\mathrm{vr}) = \max_{\mathrm{d_i},\mathrm{he}} \min[\mu_{\mathrm{AND}}^{*}(\mathrm{d_i},\mathrm{he}),\mu_{\mathrm{R}}(\mathrm{d_i},\mathrm{he},\mathrm{tr},\mathrm{rv})], \qquad (6)$$

where $\mu_{\mathrm{R}}(\mathrm{d_i},\mathrm{he},\mathrm{tr},\mathrm{rv}) = \bigcup_{\mathrm{j=1}}^{\mathrm{J}} \mu_{\mathrm{R(i)}}(\mathrm{d_i},\mathrm{he},\mathrm{tr},\mathrm{rv})$. The navigation action dictates change in robot speed and/or steering correction and it results from the deffuzification formula, which calculates the center of the area covered by the membership function computed from (6).

In order to validate the proposed scheme experiments performed in an outdoor environment which had grass, trees and some vegetation. Different set of experiments included waypoint navigation based on static predefined points while avoiding static and dynamic obstacles, raster scan of a certain area with multiple robots navigating and in the same time avoiding each other. Sample pictures of the ATRV-Jr. moving around in an area with trees are presented in Fig. 8.

Fig. 9 shows one full path traveled through an initial point to the final point, in a tree covered area while periodic laser scans are shown over the course of the robot's path.

3.2 Aerial Vehicles

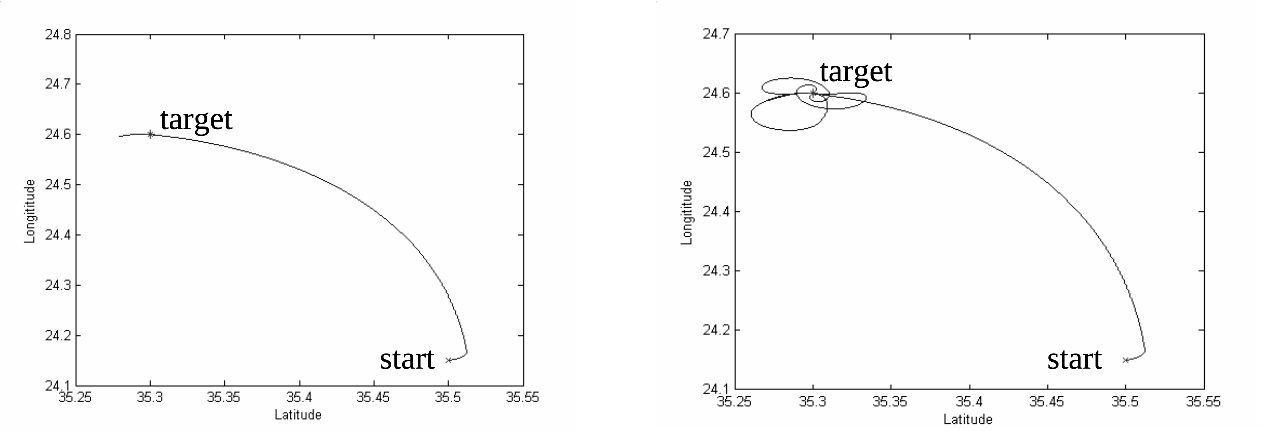
Implementations of the proposed navigation scheme for fixed wing unmanned aircrafts are presented in (Doitsidis, et al., 2004b) and in a similar approach in (Nikolos, et al., 2003a).

In (Doitsidis, et al., 2004b) a two module fuzzy logic based autonomous navigation system which is capable i) to fly through specified waypoints in a 3-D environment repeatedly, ii) to perform trajectory tracking, and, iii) to duplicate / follow another aerial vehicle. Two fuzzy logic control modules are responsible for altitude control and latitude-longitude control; when combined, they may adequately navigate the aerial vehicle. All input and output linguistic variables have a finite number of linguistic values with membership functions empirically defined.

The altitude fuzzy logic controller has three inputs, that is a) altitude error, b) change of altitude error, and, c) airspeed. The altitude error is the difference between the desired altitude and the current altitude of the airplane. The change of altitude error indicates whether the aerial vehicle is approaching the desired altitude or if it is going away from it. The airspeed is the current speed of the vehicle. Outputs are the elevators command and the throttle command, responsible for the decent and accent of the aerial vehicle. The latitude-longitude controller has as inputs the heading error and the change of heading error. The heading error is the difference between the desired and the actual heading of the airplane. The output is the roll angle of the airplane.

A simulation environment has been implemented in MATLAB. Vehicle's motion dynamics were adopted from the *Aerosim Block Set* that can be integrated in SIMULINK. The simulation test bed consists of the following subsystems: 1) Aircraft model, 2) Altitude Fuzzy Logic Controller, 3) Latitude-Longitude Fuzzy Logic Controller, and 4) Error Calculating block.

Experimental results are presented in Fig. 10. In Fig. 10a the vehicle is passing through from a certain point.



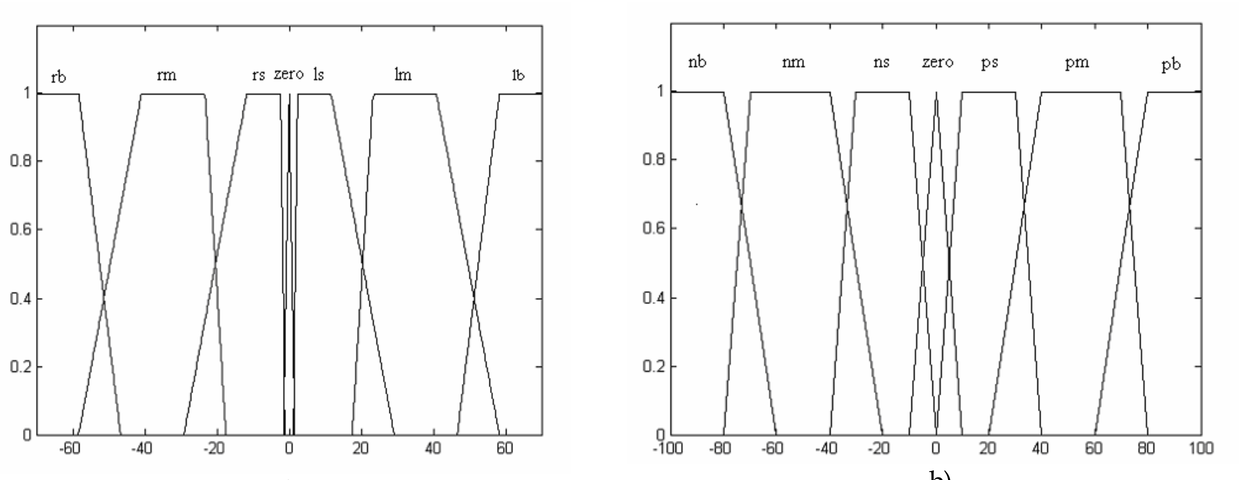
a) b)

Figure 10. Trajectories followed by the vehicle

Further, the NEARCHOS UAV, presented in Fig. 12, has been used as a test bed. The flight behavior of this UAV has been modeled in terms of simple analytic relationships, which proved very helpful in representing its actual flight in the horizontal plane. A fuzzy controller for the autonomous navigation on the horizontal plane, has been developed in (Nikolos, et al., 2003a). The controller inputs are the heading error of the aircraft and its current roll angle, while the output is the change command of the roll angle. The basic purpose of the navigation system was, to make the vehicle able to follow a predefined trajectory.

Even though the current roll angle takes values ranging from –90^0 to 90^0, the flight control system of the tested vehicle functions safely in a range from 70^0 to 70^0. The linguistic variables that represent the current roll angle are: *Right_Big (rb), Right_Medium (rm), Right_Small (rs), Zero, Left_Big (lb), Left_Medium (lm), Left_ Small (ls).* The second input to the fuzzy controller is the heading error, which is defined as the difference between the desirable and the factual direction of the aircraft. The factual direction is the heading of the aircraft, which is provided from the GPS. The desirable direction is the heading of a vector, with a starting point the current aircraft's position and ending point the desirable position. The linguistic variables that represent the heading error are: *Negative_Big (nb), Negative_Medium (nm), Negative_Small (ns), Zero, Positive_Big (pb), Positive_Medium (pm), Positive_Small (ps).* The membership functions of the input variables are presented in Fig. 11.

The desired and the actual heading direction take values ranging from 0^0 to 360^0, whereas the heading error takes values ranging from -180^0 to 180^0. However, in this implementation the heading error takes values in the region [-100^0, 100^0]. Negative (positive) values of heading error correspond to desirable right (left) roll. The linguistic variables that represent the heading error are: *Negative_Big (nb), Negative_Medium (nm), Negative_Small (ns), Zero, Positive_Big (pb), Positive_Medium (pm), Positive_Small (ps).* Experimental results about how the fuzzy logic controller performed in terms of trajectory following are presented in Fig. 13, where the continuous and discontinuous lines represent the desired and the trajectory that the fuzzy logic controller forced the vehicle to follow respectively.



a)

b)

Figure 11. Membership functions plot of input variables: a) Current Roll, b) Heading Error

Figure 12. The UAV NEARCHOS (Property of EADS – 3 SIGMA S.A)

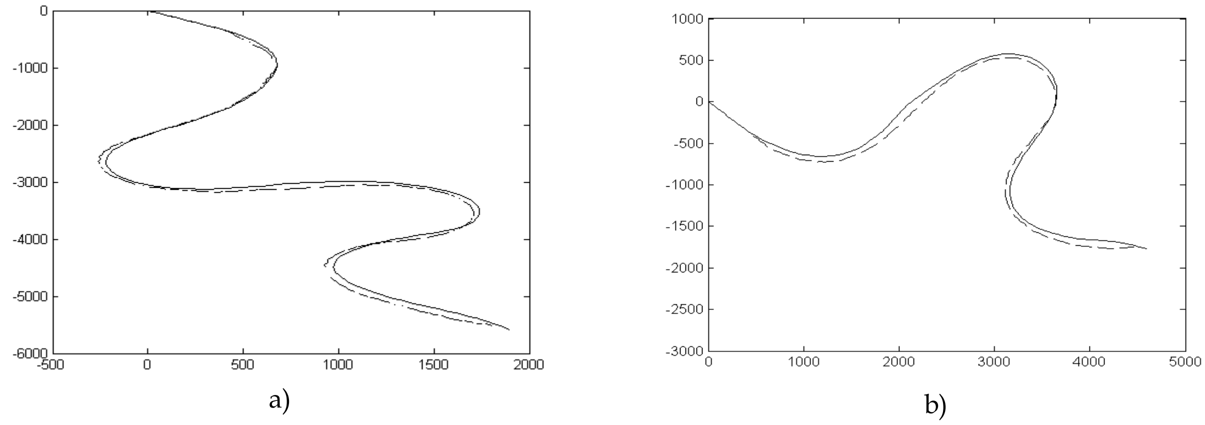


Figure 13. Trajectories followed by the UAV. Dashed lines represent the observed path while solid lines are the desired trajectories

3.3 Underwater Vehicles

Most of the difficulties in navigation of underwater vehicles (see Fig. 14) are due to the inherently uncertain nature of these environments. Fuzzy logic offers features that significantly help in addressing such problems (Kato, 1995; Tsourveloudis, et al., 1998; Kanakakis, et al., 2004). Here, we present an overview of the fuzzy logic implementations for the navigation of *Autonomous Underwater Vehicles* (AUVs) introduced in (Tsourveloudis, et al., 1998; Kanakakis, et al., 2001; Kanakakis, et al., 2004).

Some of the already known generic problems in autonomous navigation remain in the area of AUVs. These problems are, the *sensor fusion problem*: how to fuse information from different sensor modalities, or different readings from the same sensor; *the coherence problem*: how to register representations used at different levels of abstraction, and *the coordination problem*: How to coordinate the simultaneous activity of several, possibly competing behaviors such as collision avoidance and goal reaching. Additional problems in the 3-D autonomous underwater navigation are: 1) Unknown environments: Poor map and perceptual information, currents with unpredictable behaviour, and 2) Very *limited communication* options with the vehicle. Fuzzy logic navigation solutions have shown a good degree of robustness, which is crucial in the area of underwater robotics, where: 1) sonar data is unreliable, 2) mathematical models about the environment and the vehicle are usually not available, and 3) the only available navigation expertise is due to vehicle operators.

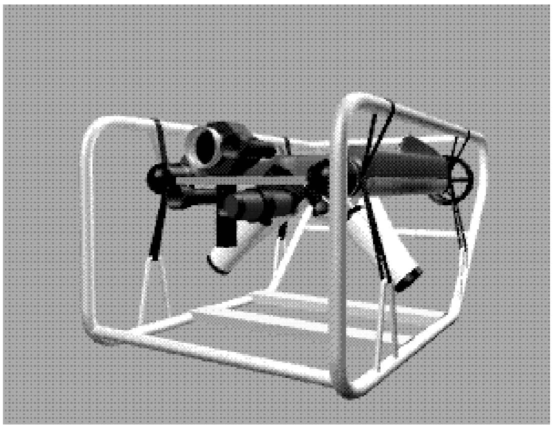


Figure 14. The underwater vehicle *Phantom S2*

The aim of underwater navigation is to guide the vehicle at a predefined point, by controlling various parameters such as pitch, yaw etc. The desired values of these parameters are the control objectives at any time instant. The fuzzy rules, which contain the navigation parameters, aim towards two goals: ideal trajectory tracking and collision avoidance. The generic expression of a fuzzy rule could be:

IF *trajectory-condition* AND *obstacle-condition* THEN *vehicle-action.*

The navigation architecture proposed for the underwater vehicles enables two layers of fuzzy controllers. These controllers can provide accurate and collision free navigation of an AUV in ocean environments, considering position accuracy with the presence of ocean currents and providing vertical stability to the vehicle. Similar to the generic architecture described in Section 2, the autonomous underwater scheme has the following modules:

- The **sensor fusion/collision avoidance module**, where the readings of the sensors of the vehicle are provided to estimate the position of the vehicle and the collision possibility in all surrounding directions. The sensor fusion module is responsible for position monitoring and obstacle detection. As AUVs operate in unknown or poorly mapped ocean environments, static or moving obstacles find themselves in the desired path of the vehicle. In these cases the vehicle should be able to use it's on board sensors to monitor its position and to detect moving or static obstacles. This implies the use of a number of different kinds of sensors, like vision cameras, laser sensors, magnetic compasses, gyroscopic mechanisms and sonar sensors. For most cases where vision is poor, sonar sensors are used to estimate an underwater environment.
- The **motion control module**, which performs low-level control of the vehicle's propellers, thrusters and fins in order to reach the determined goal point having the target surge velocity. The inputs are the goal point and the actual position and orientation, in earth-fixed coordinates, the target surge velocity and the vector of the actual vehicle velocities in body-fixed coordinates, and the sea current velocity.

Since the design of fuzzy controllers does mot require any strict modeling of the vehicle's behavior the above design is adopted for its simplicity, considering that it can be applied in all types of AUVs.

3.3.1 The Sensor Fusion/Collision Avoidance Module

The sensor fusion module outputs the collision possibility in front, right, left and back directions. The output linguistic variables are: *front_collision, right_collision, left_collision, back_collision,* taking the linguistic values *not possible, possible, high*. The collision possibilities in the four cardinal directions are computed from fuzzy rules of the type:

IF d_i *is* <$LD^{(k)}$> THEN c_j is <$LC^{(k)}$>,

and for example: IF sonar 1 *distance* is <*close*> THEN *front_collision* is <*high*>, where, k is the rule number, d_i represents the readings of the sensor i, $LD^{(k)}$ is the linguistic variable of the term set D ={*close, near, far*}, c_j is the collision of type j ($j \in$ {*not possible , possible, high*}).

A second fuzzy controller in the same module is responsible for the collision avoidance. It takes as inputs: a) *collision possibilities* with linguistic values *not possible and high*, b) *head_error* with linguistic values*: left_big, left, left_small, zero, right_small, right,* and *right_big*, and c) *pitch_error* with linguistic values *down_big, down, down _small, zero, up_small, up,* and *up_big*. The output variables are: a) *head_change* with linguistic variations, such as, *left_fast, left, left_slow, zero, right_slow, right,* and *right_fast*, b) *pitch_change* with linguistic values *down_fast, down, down _slow, zero, up_slow, up* and, *up_fast*, c) *surge_speed* with linguistic values *slow, normal,* and *high*. The collision avoidance controller consists of rules of the following form:

IF c_j is $LC^{(k)}$ AND ψ is $L\Psi^{(k)}$ AND θ is $L\Theta^{(k)}$, THEN $d\psi$ is $LD\Psi^{(k)}$ AND $d\theta$ is $LD\Theta^{(k)}$ AND u is $LDU^{(k)}$,

where, k is the rule number, c_j is the collision of type j, ψ is the heading error, θ is the pitch error, u is the vehicle's surge speed, and LC, $L\Psi$, $L\Theta$ $LD\Psi$, $LD\Theta$, LDU are the linguistic variables of c_j, ψ, θ, $d\psi$, $d\theta$, u respectively.

3.3.2 The Motion Control Module

The overall motion controller consists of the following subsystems:

- The *speed control* subsystem is responsible for the vehicle's speed by controlling its propellers revolution rate.
- The *heading control* subsystem controls the steering in the horizontal plane by controlling vehicle's head angle.
- The *depth control* subsystem controls the motion of the AUV in the vertical plane by regulating vehicle's pitch angle and depth.
- The *roll control* subsystem controls the roll parameter of the motion of the AUV.
- The *ocean current* subsystem adjusts the position of vehicle in case of undersea currents. Under the presence of a sea current, the vehicle has a drift and a deviation from the originally planned course. Although this deviation can be considered in both the speed and steering controllers, this controller adds maneuverability by modifying the steering controls. Its aim is to overcome the lateral drag by modifying the desired head and pitch angle.

The overall architecture of the fuzzy logic based navigation is shown in Fig. 15.

A vehicle control action (a fin angle, a thruster voltage or a desired propeller revolution rate) may be commanded from more than one of the above subsystems; thus, for each commanded action and during each simulation step the outputs from all subsystems form a *control vector* that controls the actual vehicle. The values of this control vector are bounded within the operational limits of vehicle servomotors to reflect reality. It should be noted that the effect of ocean currents of different velocity is taken into consideration in all phases of design and testing.

3.3.3 Simulation Results

The proposed architecture is applied to *Phoenix* AUV of the Naval Postgraduate School at Monterey, California, USA. Its dimensions and hydrodynamic model are given with clarity in the (Brutzman, 1994). The NPS-Phoenix AUV is neutrally buoyant and has a hull length of 7.3 ft. It has four paired plane surfaces (eight fins total) and four paired thrusters built in cross-body tunnels. It has two screw bi-directional propellers. Its design depth is 20 ft (6.1 m) and the hull is made of press and welded aluminum. The vehicle endurance of 90-120 min is supported by a pair of lead-acid gel batteries at speeds up to 2 ft/sec (0.61 m/sec). The behavior of the vehicle was examined under various situations including step response, ocean current, and smooth curve path. The overall performance of the controller was found to be encouraging for further research and refinement. Fig. 16 show the AUV following a rectangle saw-tooth curve in the horizontal plane and gradually descents and ascents in the vertical plane. Ocean current is present. The vehicle is simulated under ocean current with Y (earth-fixed) velocity of 0.3, 0.6, 0.8 1.0, 1.2 ft/sec. Fig. 17a, b) show the trajectories presented in Fig. 16 in vertical and horizontal planes, respectively.

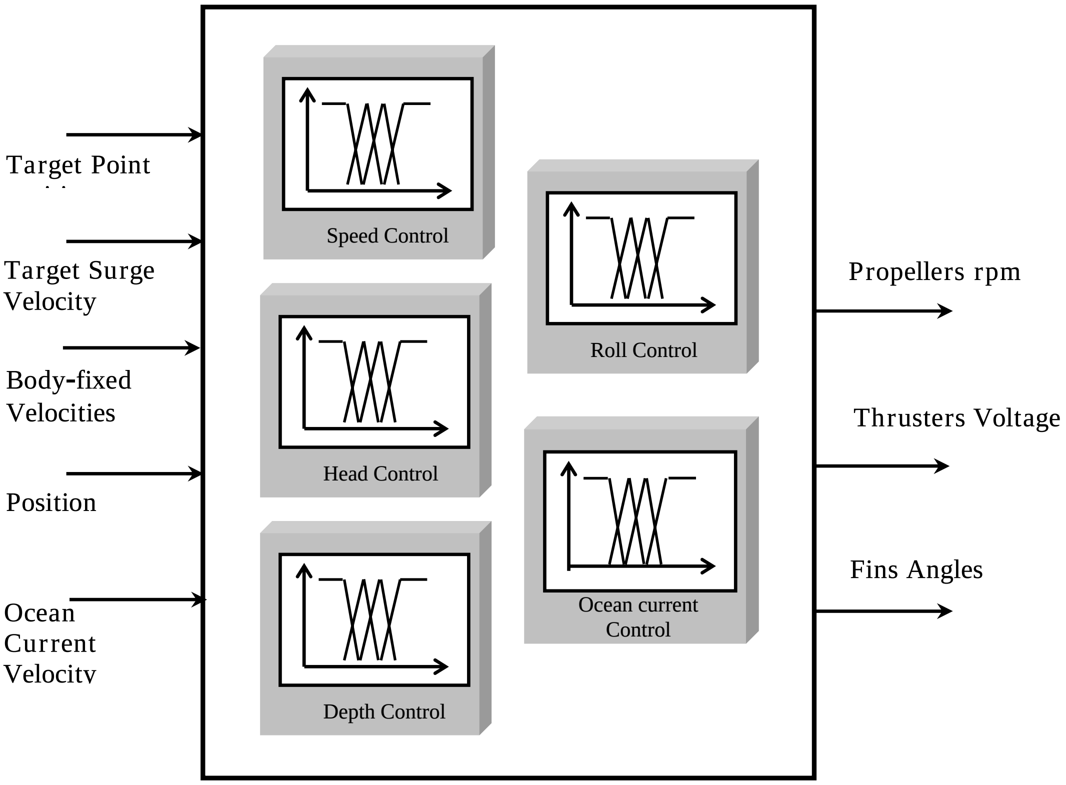


Figure. 15 The motion control module

4. Conclusions

The technology of unmanned vehicles, in all its aspects, is an exciting one, especially since it holds the promise of saving human lives by letting machines do dull, dirty or dangerous missions into high-threat environments or just unknown environments

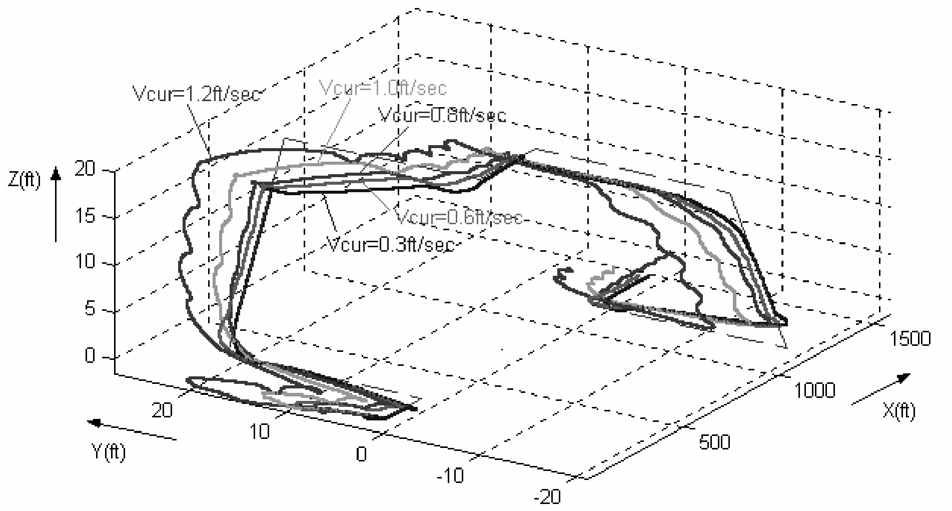
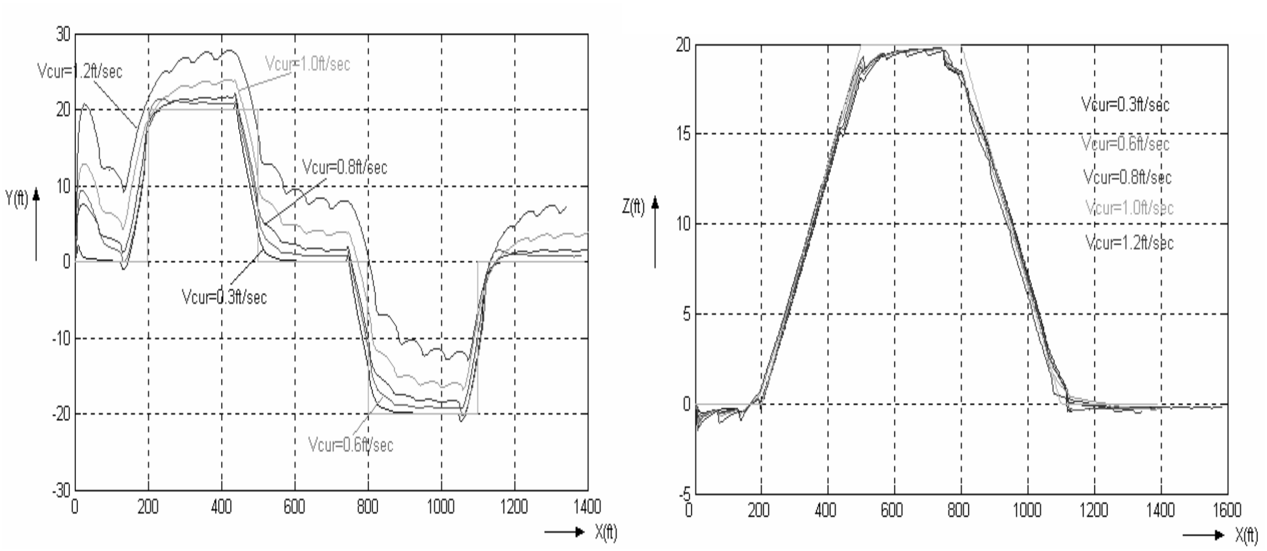


Figure 16. Simulated AUV trajectory for various ocean current velocities



a) b)

Figure 17. AUV paths for various current velocities: a) in XY plane, b) in XZ plane

This chapter shows how core research on fuzzy logic affects the advances in unmanned vehicles navigation technology and the way it can be applied to a variety of robotic vehicles, operating on ground, air or underwater. Furthermore, three key attributes for a vehicle to be considered as capable for performing autonomous navigation have been identified. *Perception* which is the ability of the vehicle to acquire knowledge about the environment and itself; *intelligence* which is the ability of a vehicle to operate for a considerable amount of time without human intervention, and *action* which is the ability of the vehicle to travel from point A to point B. How capable is a vehicle to perform these different functions is the metric to evaluate the degree of its autonomy.

Based on these attributes, fuzzy logic has been identified as a useful tool for developing controllers for vehicles so that they will be able to perform autonomous navigation. A two layer fuzzy logic controller architecture has been described. The first layer is the sensor fusion module in which the vehicle evaluates readings from various sensors and interacts with the environment.

In the second layer, which is the motion control module, the information derived from the previous layer is combined with other parameters i.e. heading, speed, altitude, position etc. This layer outputs the actual commands that will move the vehicle towards its mission.

This architecture has been proven effective in almost all types of robotic vehicles (see for example: Doitsidis, et al., 2002; Doitsidis, et al., 2004; Kanakakis, et al., 2001; Kanakakis, et al., 2004; Nikolos, et al., 2003a; Tsourveloudis, et al., 1998; Tsourveloudis, et al., 2001; Valavanis, et al., 2005).

In the next decade advances in technologies such as, Computers and Communications, Electronics, Computer-integrated manufacturing and Materials Design, will drastically support unmanned robotics to mature while dropping their costs. This will lead to a dramatic growth of unmanned robotic systems, a fact that will further affect the consumer, education, research, business and military markets.

Acknowledgement*:* L. Doitsidis is supported by "HERAKLITOS" fellowships for research from the Technical University of Crete, EPEAEK II – 88727. Kimon Valavanis has been partially supported by ONR Grants N00014-03-01-786 and N00014-04-1-0487.

5. References

Aguire, E. & Gonzalez, A. (2000). Fuzzy Behaviors for Mobile Robot Navigation: Design, Coordination and Fusion. *International Journal of Approximate Reasoning*, Vol. 25, 225-289, ISSN 0888-613X.

Amaral, T. G. B., & Crisostomo, M. M. (2001). Automatic Helicopter Control Motion using Fuzzy Logic. *Proceedings of the IEEE International Fuzzy Systems Conference*, Vol. 3, pp. 860-863, December 2001, Melbourne, Australia.

Brutzman, D, (1994).A virtual world for an autonomous underwater vehicle. Ph.D. Thesis, Naval Postgraduate School, Monterey, California, U.S.A.

Doitsidis, L.; Valavanis, K. P.; & Tsourveloudis, N. C. (2002a). Fuzzy Logic Based Autonomous Skid Steering Vehicle Navigation. *Proceedings of the IEEE International Conference on Robotics and Automation,* pp. 2171-2177, May 2002, Washington DC, U.S.A.

Doitsidis, L.; Valavanis, K. P.; Tsourveloudis, N. C. & Kontitsis, M. (2004b). A Framework for Fuzzy Logic Based UAV Navigation and Control. *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 4041-4046, April 2004, New Orleans, U.S.A.

Driankov, D. & Saffiotti, A. (2001). *Fuzzy Logic Techniques for Autonomous Vehicle Navigation*. Physica – Verlag, ISBN 3790813419, Heidelberg.

Goodridge, S. C. & Luo, R. C. (1994). Fuzzy Behavior Fusion for Reactive Control of an Autonomous Mobile Robot: MARGE. *Proceedings of the IEEE International Conference on Robotics and Automation,* pp. 1622-1627, May 1994, San Diego, U.S.A.

Hagras, H.; Callaghan, V. & Colley, M. (2001). Outdoor Mobile Robot Learning and Adaptation. *IEEE Robotics and Automation Magazine*, pp. 53-69, ISSN 1070-9932.

Hoffmann, F.; Koo, T. J. & Shakernia, O. Evolutionary Design of a Helicopter Autopilot. *Advances in Soft Computing - Engineering Design and Manufacturing, Part 3: Intelligent Control*, Springer-Verlag.

Ishikawa, S. (1991). A Method of Indoor Mobile Robot Navigation by Using Fuzzy Control. *Proceeding of the IEEE/RSJ International Workshop on Intelligent Robots and Systems,* pp. 1013-1018, November 1991, Osaka, Japan.

Kadmiry, B. & Driankov, D. (2001). Fuzzy Control of an Autonomous Helicopter. *Proccedings of the IFSA World Congress and 20th NAFIS International Conference,* pp. 2797-2802, July 2001, Vancouver, Canada.

Kadmiry, B. & Driankov, D. (2004). A fuzzy gain-scheduler for the attitude control of an unmanned helicopter. *IEEE Transactions on Fuzzy Systems*, Vol. 12, 502-515.

Kanakakis, V.; Tsourveloudis, N. C. & Valavanis, K. P. (2001). Design and testing of a fuzzy logic controller for underwater vehicles. *Proceedings of the IARP International Workshop on Underwater Robotics for Sea Exploration and Environmental Monitoring,* Rio de Janeiro, Brazil.

Kanakakis, V.; Valavanis, K. P. & Tsourveloudis, N. C. (2004). Fuzzy-Logic Based Navigation of Underwater Vehicles. *Journal of Intelligent and Robotic Systems*, Vol. 40, 45-88.

Kato, N.; (1995). Applications on fuzzy algorithm to guidance and control of underwater vehicles. Yuh. J. (ed), Underwater Robotic Vehicles: Design and Control, TSI Press.

Li, W. (1994). Fuzzy Logic-based 'Perception-Action' Behavior Control of a Mobile Robot in Uncertain Environment. *Proceedings of the 3rd IEEE Conference on Fuzzy Systems,* pp. 1626-1631, March 1994, San Francisco, U.S.A.

Nikolos, I. K.; Doitsidis, L.; Christopoulos, V. N. & Tsourveloudis, N. C. (2003a). Roll Control of Unmanned Aerial Vehicles using Fuzzy Logic. *WSEAS Transactions on Systems*, Vol. 4, 1039-1047.

Nikolos, I. K.; Valavanis, K. P.; Tsourveloudis, N. C. & Kostaras, A. N. (2003b). Evolutionary algorithm based offline/online path planner for UAV navigation. *IEEE Transactions on Systems, Man and Cybernetics, Part B*, Vol. 33, 898 – 912.

Oriolo, G.; Ulivi, G. & Vindittelli, M. (1998). Real-time Map Builting and Navigation for Autonomous Robots in Unknown Environments. *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 28, 316-333.

Pin, F. G. & Watanabe, Y. (1994). Navigation of Mobile Robots Using Fuzzy Logic Behaviorist Approach and Custom Design Fuzzy Inference Boards. *Robotica,* Vol. 12.

Saffiotti, A.; Ruspini, E. H. & Konoligr, K. (1993). Blending Reactivity and Goal-Directedness in a Fuzzy Controller. *Proceedings of the Second IEEE Conference on Fuzzy Systems*, pp. 134-139, March 1993, San Francisco, U.S.A.

Seraji, H. & Howard, A. (2002), Behavior-Based Robot Navigation on Challenging Terrain: A Fuzzy Logic Approach. *IEEE Transactions on Robotics and Automation*, Vol. 18, 308-321.

Sugeno, M. (1999). Development of an Intelligent Unmanned Helicopter. *Fuzzy Modeling and Control: Selected Works of M. Sugeno*, H. T. Nguyen and N. R. Prasad (eds), CRC Press, Boca Raton.

Tsourveloudis, N. C., Gracanin, D., Valavanis, K. P. (1998). Design and testing of navigation algorithm for shallow water autonomous underwater vehicle. *Proceedings of the IEEE OCEANS 98*, Nice, France.

Tsourveloudis, N. C.; Valavanis, K. P. & Hebert, T. (2001). Autonomous Vehicle Navigation Utilizing Electrostatic Potential Fields and Fuzzy Logic. *IEEE Transactions on Robotics and Automation*, Vol. 17, 490-497.

Tunstel, E. (1996). Mobile Robots Autonomy Via Hierarchical Fuzzy Behavior. *Proceedings of the 6th International Symposium on Robotics and Manufacturing,* pp. 837-842, May 1996, Montpelier, France.

Valavanis. K. P.; Doitsidis. L.; Long. M. T. & Murphy. R. R (2005).Validation of a Distributed Field Robot Architecture Integrated with a MATLAB Based Control Theoretic Environment: A Case Study of Fuzzy Logic Based Robot Navigation", *IEEE Robotics and Automation Magazine,* (In press).

Vaneck, T.W. (1997). Fuzzy Guidance Controller for an Autonomous Boat. *IEEE Control Systems Magazine*, Vol. 17. 43-51.

White, B. A.; Blumel, A. L. & Hughes, E. J., (2000). A Robust Fuzzy Autopilot Design Using Multi-Criteria Optimization. *International Journal of Fuzzy Systems*, Vol. 2, pp. 129-138.

-IV-

Adaptive and Learning Systems

Integrating Behaviors for Mobile Robots An ethological Approach

Jose Maria Canas Plaza & Vicente Matellan Olivera

1. Introduction

Robots available nowadays in the everyday markets can be divided into two major groups. First, those that are oriented to a single task, as vacuum cleaners (roomba[1], robocleaner[2], ...), lawn mowers (automower[3], robomower[4], ...), etc; and second, those oriented to various tasks, as robotic pets (Aibo[5], Necoro[6],...), museum guiders, or research platforms (Pioneer, Koala, etc.). In order to get service robots or personal assistants we need to improve the abilities of the second ones. These abilities have to do with their ability to smartly combine their basic skills to obtain behaviors that are more complex.

The generation of autonomous behavior is a very complex issue. Within a multi-goal system, like service robots, the focus is more on the integration than on the control of perception algorithms. The organization becomes the critical issue; robustness and flexibility are key features. As animals can do it, we can see no compelling reason why robots could not, but more research on robot architectures is needed to reach an acceptable performance.

Several paradigms have been historically proposed for behaviour generation in robots. These paradigms are also known as architectures. Dynamic and uncertain environments forced the evolution from symbolic AI (Nilsson 1984) to reactive and behavior based systems (Brooks 1986; Arkin 1989).

The behaviors based systems adapt smoothly. They have no anticipation and no state, but they have shown poor scalability for complex systems. Hybrid architectures have been predominant since mid 90s (Simmons 1994; Konolige 1998; Bonasso 1997), mainly those three-tiered ones that add two layers to behavior based ones, usually a sequencer, and a deliberator.

Several architectures have been explored after the hybrid three-tiered architectures became the de facto standard. In particular, many reviews of the hierarchy principle have been proposed in last years (Arkin 2003, Saffiotti 2003, Nicolescu 2002, Behnke 2001, Bryson 2001), trying to overcome subsumption limitations.

Our contribution in this area is a novel hierarchical approach named JDE. This new hierarchical approach, ethologically inspired, is based on the selective activation of

[1] http://www.roombavac.com
[2] http://www.robocleaner.de
[3] http://www.automower.com
[4] http://www.friendlyrobotics.com
[5] http://www.aibo-europe.com
[6] http://www.necoro.com

schemas, which generates the dynamic hierarchies that govern the robot behavior. In section 2 we describe its major characteristics.

Besides the conceptual fundamentals, every architecture has to be implemented in software. In section 3 we present some of the existing software platforms for the development of robotic software. JDE also provides a software infrastructure to ease the development of robotics software. In section 4 this software architecture is described in detail.

2. JDE Architecture

In JDE terminology a "behavior" is considered the close combination of perception and actuacion in the same platform. Both are splited in small units called schemas (Arkin 1989). *Perceptive schemas* build some piece of information concerning the environment or the robot itself, and *actuation schemas* make control decisions considering such information like speeding up the motors or moving some joint. JDE also proposes the schemas can be combined in a dynamic hierarchy to unfold the behavior repertoire of the robot accordingly to current goals and environment situation. Foundations, and further details of JDE can be found in the PhD dissertation (Cañas 2003).

2.1 Schemas

JDE follows the Arkin's definition (Arkin 1998) "a *schema* is the basic unit of behavior from which complex actions can be constructed; it consists of the knowledge of how to act or perceive as well as the computational process by which it is enacted''.

They are thought as continuous feedback units, similar to teleo-reactive programs in (Nilsson 1997) and are close to control circuitry, which continuously update its outputs from its inputs. In JDE each schema has a time scale and a state associated. They can be switched on and off at will and they accept some modulation parameters. The schemas are thought as iterative processes with a goal, which continuous execution allows them to build and keep updated some stimuli or to develop some behavior.

There are two types of schemas, perceptive and actuation ones. Each *perceptive schema* builds some information piece about the environment or the robot itself, which we'll call a stimulus, and keeps it updated and grounded. It can take as input the value of sensor readings, or even stimuli elaborated by other schemas. It also accepts modulating parameters. Perceptive schemas can be in "slept" or in "active" state.

Each *actuation schema* takes control decisions in order to achieve or maintain its own goals, taking into account the information gathered by different sensors and the associated perceptive schemas. The output of an actuation schema typically orders commands to actuators, and can also activate other schemas, both perceptive and actuation ones. Such new schemas are regarded as its children, and the parent schema often modulates them through mentioned parameters.

Actuation schemas can be in several states: "slept", "checking", "ready" and "active", closely related to how action selection is solved in JDE. Control schemas have preconditions, which implicitly describe in which situations such schema is applicable and may achieve its goals.

The algorithm running inside the schema contains all the task-knowledge about how to perceive relevant items and how to act, whatever technique be used. JDE architecture only imposes the interface: selective activation (on-off) and parameters for modulation, as long as it affects the way all the pieces are assembled together into the system. Continuous (iterative) execution is assumed, actively mapping its inputs into its outputs, regardless how it is achieved: case-based actuation, fuzzy controller, crisp function, finite-state-automata, production sytem, etc.

2.2 Hierarchy

All schemas that are awake (those in "checking", "ready" or "active" states) run concurrently, similar to the distribution found in behavior-based systems. To avoid incoherent behavior and contradictory commands to actuators JDE proposes hierarchical activation as the skeleton of the collection of schemas. It also claims that such hierarchical organization, in the ethological sense, provides many other advantages for roboticists like bounded complexity for action selection, action-perception coupling and distributed monitoring. All of them without losing the reactivity needed to face dynamic and uncertain environments. We consider that these features make easier the development of versatile and flexible artificial behavior systems.

In JDE there is hierarchy as long as one schema can activate other schemas. An actuation schema may command to actuators directly or may awake a set of new child schemas. These children will execute concurrently and they will achieve in conjunction the father's goal while pursuing their own. Actually, that's why the father awoke such schemas, and not others. A continuous control competition between all the brothers determines whether each child schema will finally get the "active" state or remains silent in "checking" or "ready" state. Only one winner, if any, pass to "active" state and is allowed to send commands to the actuators or spring their own child schemas. The father also activates the perceptive schemas needed to resolve the control competition between its actuation children and the information needed for them to work and take control decisions. This recursive activation of perceptive and actuation schemas conforms a schema hierarchy.

For instance, in the figure 1 perceptive schemas are squares and actuation ones are circles. Schema 1 activates the perceptive children 3 and 4 to build relevant information, and awakes the actuation schemas 5, 6, and 7. The 6 wins the control competition and then activates 11, 17, 14, and 15.

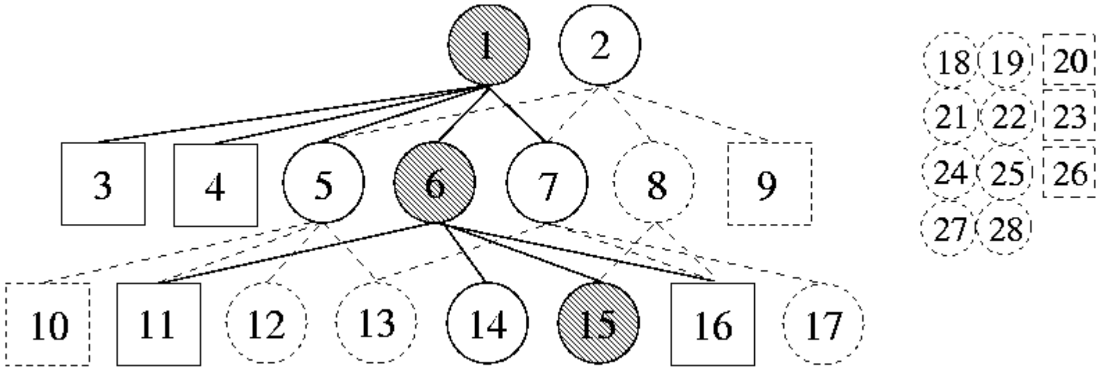


Figure 1. Schema hierarchy in JDE

The last one, 15, gains control and really sends commands to the actuators. All winning schemas have been represented in dark circles. All the dashed schemas are in "slept" state and do not consume computing power at all, they are not relevant to the current context.

Once the father has awaken their children it keeps itself executing, continuously checking its own preconditions, monitoring the effects of its current kids, modulating them appropriately and keeping them awake, or maybe changing to other children if they can face better the new situation. At its abstraction level, the father has the entire behavior context to react accordingly to changes in situation. With continuous modulation, the father may adjust the system actuation to small changes in situation, if needed. Changing the children schema lets the hierarchy to reconfigure itself, in response to more significant variations in environment. This way the activation flows from the root adapting the system accordingly to the dynamic situation. Bigger changes may cease the satisfaction of the father's preconditions, and all its children are deactivated, forcing the reconfiguration of the hierarchy at a higher level.

There is no privileged number of levels in JDE, it depends on the complexity of the task at hand and may evolve as new situations are encountered. Higher levels tend to be more abstract and with slower time cycle. The layer concept is weak in JDE, as they are dynamic and not always with the same units inside. The schema is the unit, more than the layer.

2.3 Action Selection

JDE decomposes the whole Action Selection Mechanism (ASM) into several simpler action selection contests. At each level of the hierarchy, there is a winner-takes-all competition among all actuation schemas of such level. There is only one winner at a time, and that is the only one allowed to send commands to the actuators or even spring more child schemas.

Before a given schema wins the control, it has to go through several states. First, when the parent awakes it, it passes from "slept" to "checking" state. Second, the schema promotes from "checking" to "ready" when its preconditions match current situation.

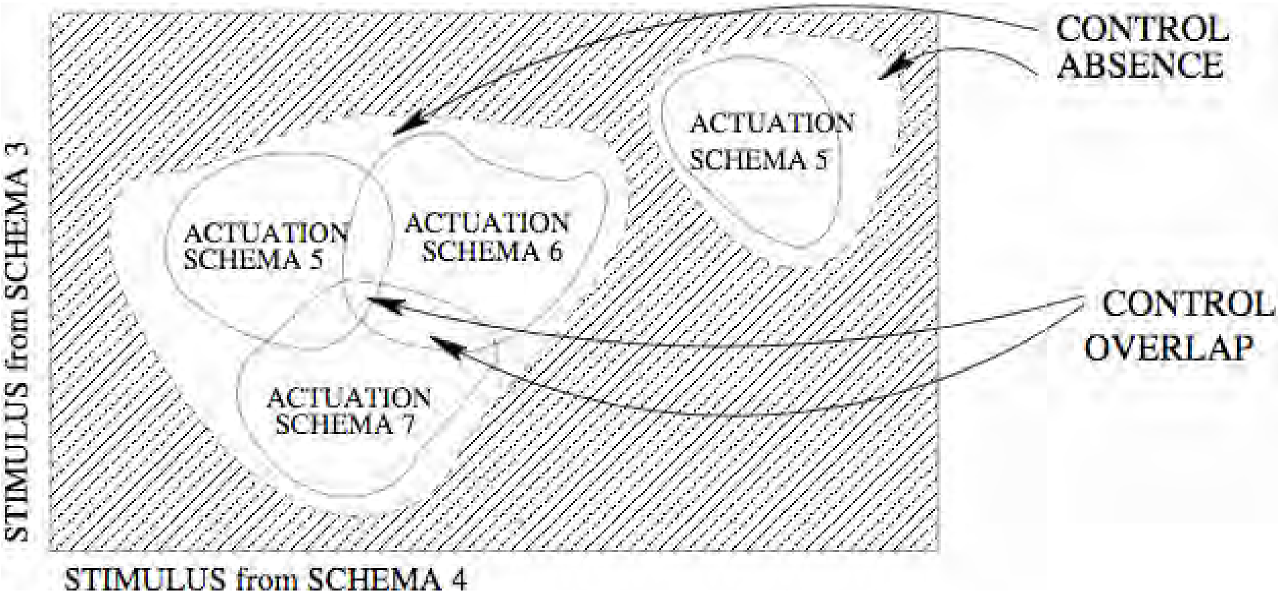


Figure 2. Activation regions in all plausible situations of such context

The preconditions are seen here as ethological triggers (Tinbergen 1951), and preconditions of teleo-reactive programs (Nilsson 1997). They define the set of situations in which such schema is applicable and may achieve its goal, which in JDE is named the *activation region* of such schema. Third, typically, the preconditions of only one schema will hold and it will move from "ready" to "active" state. In case of several (or none) "ready" brothers, the parent is called for choosing one winner, using a fixed priority rule o even a more flexible arbitration.

For instance, in figure 2 the activation regions of schemas 5, 6, and 7 of figure 1 are displayed. They are defined over the space of possible values for stimuli built by perceptive schemas 3 and 4. Impossible combinations are displayed in shadow. The situation is a point in such space and it may fall inside an activation region, such schema will promote to "ready" or outside. This picture is similar to state-space diagrams in (Arkin 2003). The father, schema 1, modulates the activation regions of its children through parameters in order to get more or less disjoint and exhaustive regions. Uncovered or overlap areas explicitly solved in the ad-hoc arbitration when needed.

The JDE architecture is goal oriented and situated. It is goal oriented because the winner lies in the set of schemas already awaken by the father, and no others. It is situated because the environment chooses among them the most suitable to cope with current situation. It is also fast: a new action selection takes place at every child iteration, which allows timely reaction to environment changes.

This approach reduces the complexity, because in a population of *n* schemas there is no need to resolve $O(n^2)$ interactions, but only a number of small competitions where typically 3 or 4 schemas compete at a time for control. In addition, such competitions arise in the context of the parent schema being "active", so the possible situations are bounded.

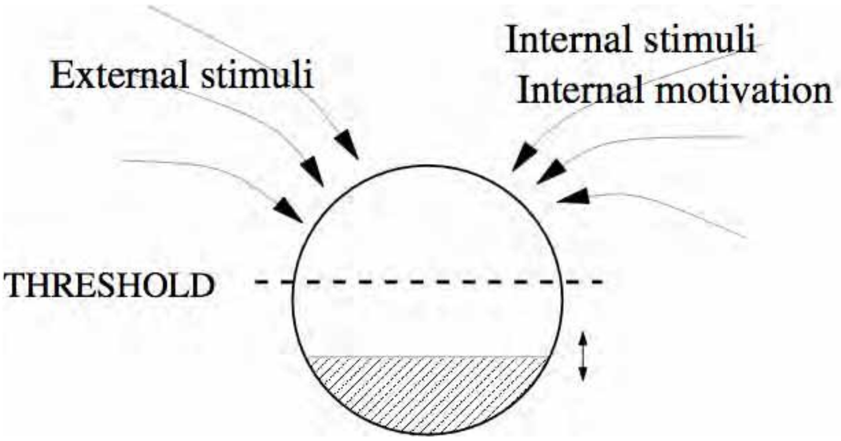


Figure 3. Additive combination of stimuli in JDE

The preconditions can be thought not as the crisp evaluation of the truth of a single condition, but as a thresholded continuous variable which accumulates the contribution of different stimuli to the trigger itself, as can be seen in figure 3. This way it provides room for the additive combination of stimuli and the triggered response observed in the Innate Release Mechanism of animals (Lorenz 1981). Some of such stimuli may be internal ones or motivations (like thirst, fear, hunger, sexual appetite, etc.) with their own dynamics. Several robotics works have explored the effect of such internal variables in action selection (Tyrrell 1993, Arkin 2003).

2.4 Perception

Perception has traditionally deserved little interest in the community devoted to research on robotics architectures, but we think that it is an essential part of behavior generation systems, as it provides *all* the information on which an autonomous robot must make its control decisions. Perception is relevant both in action selection, data for triggering conditions (which actuation schema will actually gain control), and in the normal execution of actuation schemas (which actuation decision, or motor value will such schema order to actuators).

The active schemas may change as time passes, so the perception output is a dynamic collection of stimuli more than a general world model. In JDE, perception is task oriented. Actually, it is tightly coupled to action, as the parent schema awakes both the actuation children, and, at the same time, the perceptive schemas which build the relevant information.

Symbols and state are allowed in JDE. The activation of one schema indicates the internal predisposition to such stimulus, as some computing power is devoted to search for and update its internal symbols. Continuous execution of perceptive schemas keeps their symbols grounded just to perceive them when they appear. The stimuli may appear in reality but if no internal process exists attending to it, the robot will miss it. The event has just not existed for the robot.

Hierarchy naturally offers an attention mechanism, and so JDE perception is selective and situated. Only the data relevant to the current context are searched for, as only such perceptive schemas are "active". No computational resources are spent building stimuli not needed now, because the control schemas that need them and the associated perceptive schemas are "slept", which is the default state. In contrast, in Brooks system (Brooks 1986) all the automata are always active.

This selective perception in JDE is very convenient because the number of relevant stimuli of the environment, which need to be internalized, grows when the whole system is going to exhibit a wide range of behaviors. Perception costs computing time, which is always limited. This all-time perception of all potential stimuli does not scale to complex and versatile systems, especially if vision is involved.

An attention mechanism is clearly needed. Again, this requirement is not needed if the robot is going to do a single task, but is essential if a full set of behaviors is going to be integrated in a single system.

Sensor fusion and complex stimuli (Arkin98) may be generated in JDE as perceptive schema may activate perceptive children and fuse their outputs into a compound stimulus.

2.5 Behavior Patterns in JDE

JDE can be used to generate usual behavior patterns found in the robotics literature. For instance, a fixed sequence can be generated with states inside a given schema, and the simple time passing as the triggering condition from one state to the next one.

Taxias (Lorenz 1981) can be easily generated in JDE as the conjunction of two schemas: the perceptive one characterizes the key stimulus, its position, etc.; and the actuation schema decides what to do. For instance, a tracking behavior uses negative feedback, and an obstacle avoidance behavior uses a positive one.

Flexible behavior sequences are identified in animals (Lorenz 1981) and in robotics. In order to implement them trigger condition can be used in JDE. These conditions allow proceeding with the next action in the sequence. For instance, if a sequence of three behaviors want to be implemented, the third schema will really carry on the action that achieves the father's goal. The stimulus that has to be present to promote this schema will be produced by a second schema, that is, the execution of the second schema increases the chance of the stimulus to appear, so it is an appetitive behavior. The same applies to the second with its stimulus and the first.

In this way, JDE hierarchy is not intended as a task-decomposition in primitive steps, which must be executed in sequence, but as a predisposition. There is no a priori strict commitment to a single course of action. Actually the father doesn't fully activate their actuation children, just awakes them. Which one really gains control at every instant depends on the situation, so the sequence adapts its unfolding order to the environmental conditions.

Like reactive planning (Nilsson 1997, Firby 1992), many courses of action are valid and all of them are contained in the behavior specification as a collection of child schemas.

This flexibility allows the system to take advantage of environment opportunities and to face little contingencies.

3. Programming Real Robots

All the previously described patterns have to be implemented in a real robot, which means that they have to be implemented in a particular programming language, adapted to different robotic platforms, etc. This is the other sense of the word "architecture" when applied to robotics, the software infrastructure. Different research groups, manufacturers, etc. have developed different software architectures, in this section we will review their main characteristics.

3.1 Implementing a Robotic Architecture

The ultimate goal of robotic architectures is to be able to make working robots. This requires making choices about which software components use, and how to configure them. This decision is conditioned by several requirements:

- Mobile robots works in real world domains where real-time is required, may be not hard real-time, but al least soft real-time. Besides, there are several types of sensors and actuators, as well as interfaces that the programmer must know.

- Typical robotics applications must manage various sources of information (sensors) and pursuing various goals at the same time.

- User interface in robotics applications is another major concern in robotics, not only from the human computer interaction point of view, in the interaction of robots with users; but for the debugging of the pa.

- Robotic software is increasingly distributed, as Woo et al. (Woo 2003) have pointed out. It is usual that robotic applications have to communicate with other processes, in the same machine or in other computers.

- There are few knowledge about how generate and organize robotic behavior. How to organize it is still a research issue and there is no general guides about.

- There are no open standards for robotic software developent (Utz 2002, Montemerlo 2003, Cote 2004). In other computer science fields there libraries that programmer can use. However, in robotics there are few reusable software, most programs have to be built from scratch. This is due to the heterogeneity of robotic hardware and to the immaturity of the robotic industry.

The way robots are programmed has been changing. Historically the robots were unique, they were not produced in series, and programs were built directly using device drivers; operating systems were minimal. Application programs read sensor data directly from sensors using the drivers, and wrote the commands directly on the motors using the library provided by the robot manufacturer.

The reduction in the number of manufacturers, and their consolidation, as well as the work of many research groups, have made possible the appearance of development platforms that simplify the development of software, as well as different tools.

3.2 Development Tools

Creating software for mobile robots is in essence as any other application development. Programmer has to write the program in a particular programming language, compile it, link it with the libraries, and finally execute it in the on-board computers. In many cases, all this process is performed in a PC, using cross compilers.

There are some specific programming languages for robotics, as for instance TDL (Task Description Language) (Simmons 1998) or RAP (Reactive Action Packages) (Firby 1994). However, they have not been a success, and usual programming languages (C, C++, Java, etc.) are the dominant ones.

Another set of useful tools is the simulators. Early simulators were little realistic; only simulate flat worlds populated by static obstacles. Nowadays simulators have been greatly improved. There are 3-D simulators like Gazebo[7], or able to simulate vision (Lozano 2001), or to include many robots in the same world, as Player/Stage[7]. Besides, many robotic manufacturers include a simulator for their robots, as for instance, EyeSim for the EyeBot, Webots for Khepera and Koala, etc. However, we think that general simulators are better option. Among the open source simulators not affiliated to any manufacturer, we think that SRISim[8] and Stage[7] are the most relevant.

3.3 Robotic Software Development Platforms

In many areas of software development "middleware" has appeared as a way to simplify software development. Middleware provides predefined data structures, communication protocols, sincronization mechanisms, etc. In the robotics software different platforms have appeared (Utz 2002).

Hattig et al. (Horswill 2003) have pointed out that uniform access to hardware is the first step towards software reuse in the robotics industry. This characteristic can be found in many platforms, but each one implements it in its own way. For example, in ARIA the API for accessing sensors and actuators has been made up by a set of methods, while in Player/Stage or in JDE has been implemented as a protocol between the applications and servers.

JDE software infrastructure will be described in next chapter. In order to compare it with other alternatives available, we will analyze in this section some software development platforms available nowadays for the robotics programmers. We will analyze both

[7] http://playerstage.sourceforge.net

[8] http://www.ai.sri.com/~konolige/saphira

commercial platforms, such as ARIA from ActivMedia, OPEN-R[9] from Sony; as well as, open source efforts as Player/Stage, MIRO, Orocos (Bruyninckx 2001), or CARMEN.

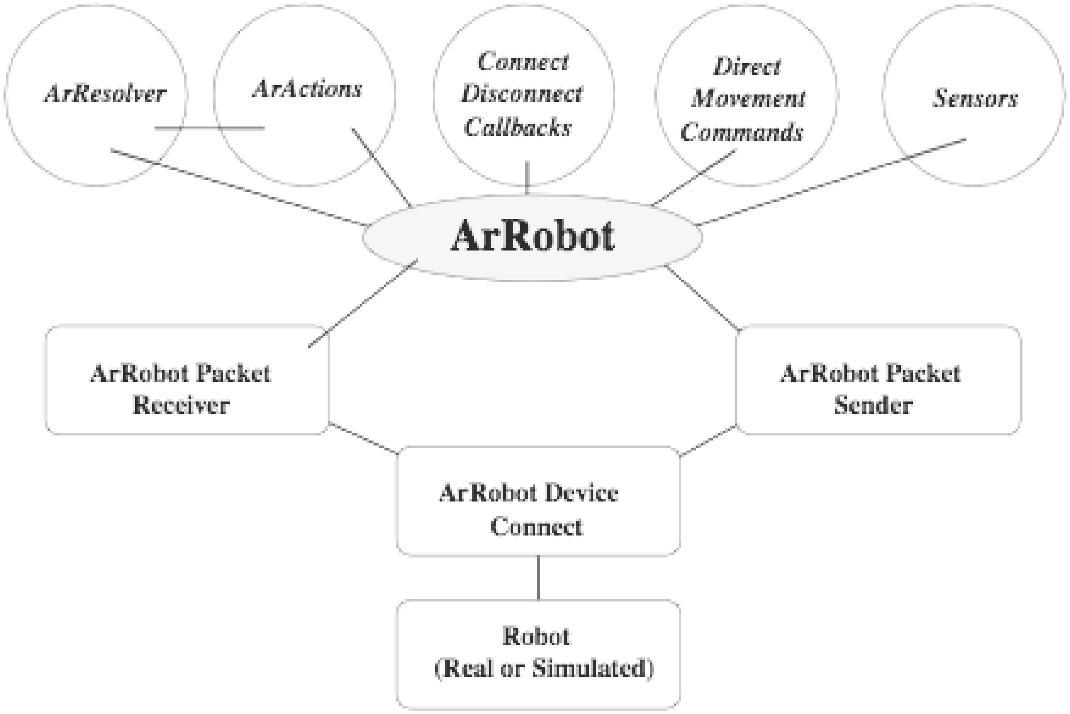


Figure 4. Architecture of the ARIA API

ARIA

ActivMedia has become the leader in the robotics research manufacturer business after the shutdown of Nomadic and RWI. ARIA (Activmedia Robotics Interface for Applications) (ActivMedia 2002) is the development environment distributed by ActivMedia with its robots (Pioneer, Peoplebot, Amigobot, etc.).

Underlying the API offered by ARIA, there is a client-server design. ActivMedia robots are managed by a micro-controller that implements the server. The application written using ARIA connects to this server using a set of predefined messages (protocol) through a serial port.

ARIA offers an object-oriented environment (applications have to be written in C++). It offers multitasking support and communications facilities. In this case, Aria objects are not distributed; they are placed in the computer that is physically connected to the robot. Anyway, ARIA offers the ArNetworking class to manage communications.

Hardware access in Aria is made through a collection of classes as shown in figure 4. Main class is ArRobot where methods as ArRobot::setVel commands a translational velocity and ArRobot::setRotVel the rotational one. Packet Receiver and Packet Sender, manage the sending and reception of packages through the serial port. Other classes as ArSonarDevice or ArSick contain methods for accessing those devices.

ARIA is offered for Linux and Win32 and it is distributed under GPL license. It also includes some basic behaviors as obstacle avoidance, but more complex ones are offered as proprietary separate libraries: MAPPER for managing maps, ARNL (Robot Navigation

[9] http://www.openr.aibo.com

and Localization) for navigation algorithms, ACTS (Color-Tracking Software) to identify objects, etc.

MIRO

Miro[10] is a distributed object-oriented platform developed at Ulm University and distributed under GPL license. It uses CORBA as the underlying standard for object distribution, to be exact TAO.
Miro is composed by three levels: devices, services, and classes. The first one, "Miro Device Layer", is the one devoted to access hardware devices; it offers different objects for every device. That is, sensors and actuators are accessed through methods of different objects, depending on the hardware. For example, RangeSensor defines an interface for sensors such as sonar, infrared, laser, etc. DifferentialMotion contains the methods to move the robot, etc. Second layer, "Miro Services Layer", provides descriptions of the sensors and actuators as CORBA IDLs, so they can be accessed remotely by objects running in other computers, independently of the operating system.
Third level, "Miro Class Framework" groups the tools for visualization and logs management, as well as general use modules, as map building, path planning, or behavior generation, etc.
An application developed using Miro is a collection of local and remote objects. Each one runs in its own computer and all of them communicate using the infrastructure of the platform. Objects can be written in any language that supports the CORBA standard. Miro itself has been written in C++.
There are other platforms using distributed objects, and CORBA. One of the better known is Orocos (Open RObot COntrol Software), the open-source project funded by the European Union.

OPEN-R

Open-R is the API released by Sony for programming the Aibo robot. Aibo is a robotic pet, whose main sensor is a camera situated in the head, and whose actuators are the four legs, the tail and the neck. It is an autonomous robot based in a 64 bit RISC processor, able to communicate using an integrated WiFi card.
The operating system used in this robot is Aperios (previously know as Apertos, and Muse). It is a real-time object oriented operating system. It is proprietary software, and Sony has just release the Open-R API. Open-R offers just a C++ interface, and applications can consist in one or more Open-R objects (Martín 2004). Application objects can use a set of basic objects, and that can send and receive messages among them. Each object runs in its own thread and control flow is event-based.
Among the basic objects, OvirtualRobotComm lets applications access to images and joints through services as Effectors, that moves the dog joints, or switches the LEDs on and off; Sensor that reads the position of the joits; or OfbkImageSensor to access the recorded images. ANT object manages the TCP/IP stack to let applications communicate outside the robot, offering the Send and Receive services.

[10] http://smart.informatk.uni-ulm.de/MIRO

Development environment using Open-R is a PC using a cross compiler for the Aibo RISC. Programs developed are written in a memory stick that it is inserted in the robotic dog.

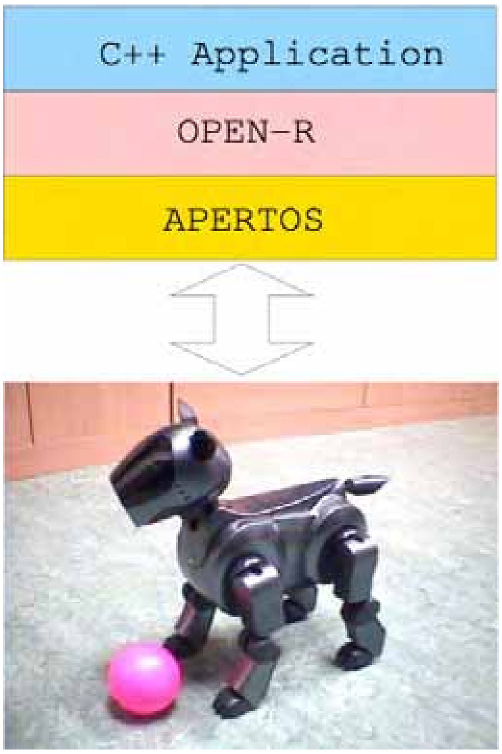


Figure 5. Software architecture for programming Sony Aibo robot

Player/Stage/Gazebo

Another open-source platform is Player/Stage/Gazebo (PSG). Initially developed at Southern California (Gerkey 2003), nowadays is an open-source project. Nowadays it support different robots (Aibo, Pioneer, Segway, etc.), and the simulator included makes him a complete platform.

In PSG sensors and actuators are managed as files (Vaughan 2003) as in Unix operating system. Five basic operations can be performed on those files: open, close, read, write, and configure. Every type of device is defined in PSG by an interface; for example, the ultrasonic sensors of two different robots will be an instance of the same interface.

PSG is based on client-server design. Any application has to establish a dialogue using TCP/IP with the server Player. This idea let applications be really independent (i.e. they can be written in any programming language) and imposes minimal requirements on their architecture.

PSG is oriented to offer an abstract interface of the hardware of the robots, not to offer common blocks. However, these blocks can be added defining new messages for the protocol. For instance, probabilistic localization has been added as another interface localization, that provides multiple hypotheses. This new interface overwrites the traditional position based just in odometry.

There are other well-know platforms, as for instance Mobility, the one developed by RWI (B21, B14 are classical robots); Evolution Robotics sells its ERSP; CARMEN is offered by Carnegie Mellon university; etc. We have described the previous four just to show the reader the major alternatives faced when developing JDE software infrastructure described in next section.

4. JDE Software Infrastructure

Once we have seen existing platforms, we will describe in this section the characteristics of JDE software infrastructure. Current JDE implementation consists of the infrastructure software (JDE servers), a collection of behavior specific schemas (JDE basic schemas), and auxiliary tools and libraries.

A typical robot control program in JDE programming environment is made up of a collection of several concurrent asynchronous threads, corresponding to the JDE schemas. Over basic schemas there may be perceptive and actuation schemas. Perceptive schemas make some data processing to provide information about the world or the robot. Actuation schemas make decisions in order to reach or maintain some goal. They order motor commands or activate new schemas, because the schemas can be combined forming hierarchies. The underlying theoretical foundations have been described in section 2.

Figure 6 summarizes this architecture. Two JDE servers (OTOS and OCULO) are in the middle, represented as two circles. These servers access the robot sensors and actuators (in the low part of the figure 4) using specialized drivers, and provide the data to clients (small circles in the top of the figure 4) through a protocol named *JDE Protocol*.

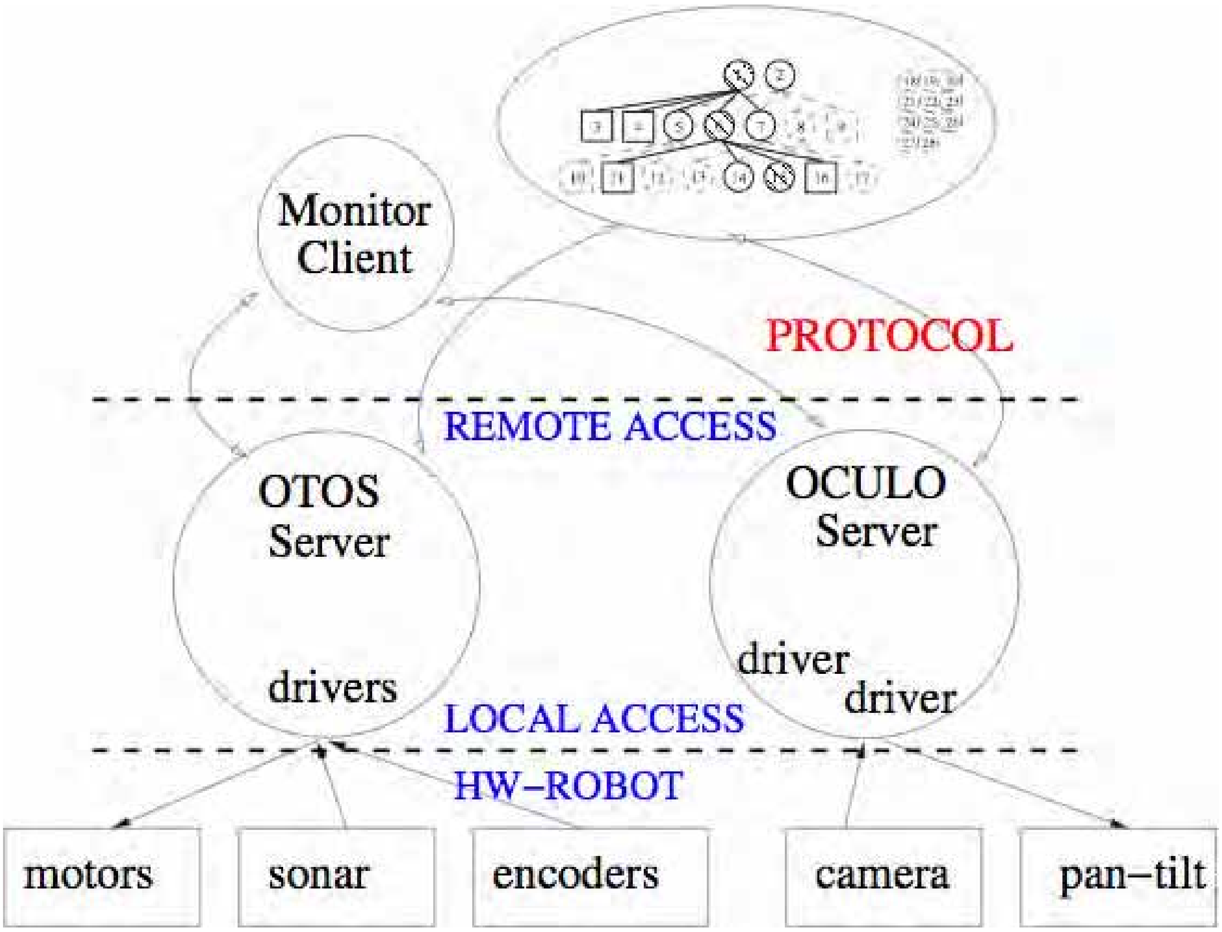


Figure 6. JDE software architecture. OTOS and OCULO servers

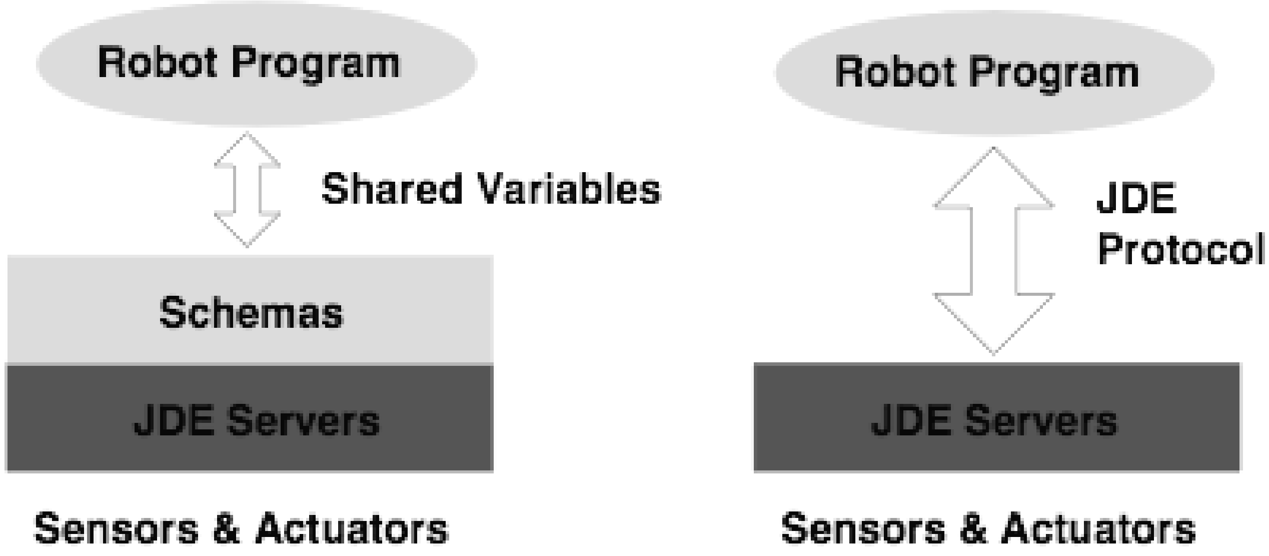


Figure 7. Programming options using JDE

Developers can build their applications using directly these servers; they just need to implement the protocol to communicate with the servers.JDE also provides some auxiliary clients, as for instance a monitor of the sensors (circle at the top left of the figure). Others clients implement the schemas in which JDE is based on.

In order to facilitate the implementation of JDE schemas, the distribution provides some basic schemas that interact with the servers, offering sensor data as shared variables, which is much more simple than implementing the protocol. The release also includes the skeleton of the schemas to ease its use.

In summary, JDE software implementation offers two options to develop programs, directly using JDE servers, or using the service schemas. The second one is the recommended one both because is easier, and because is closer to the conceptual ideas behind JDE. Figure 7 shows these two options.

4.1 JDE Servers

Robot application gets sensor readings and order actuator commands sending network messages to the JDE servers. There are subscription messages for short sensor readings like odometer, sonar or laser. Those data are continuously sent from the server to the subscribed clients. Large readings, like camera images, are sent only on demand. There are also actuation messages from clients to servers. All these messages make up a protocol that settles a hardware abstraction layer and provides language and operating system independence to the robot applications.

OTOS server manages proximity sensors, such as sonar, laser, infrared, and tactile sensors. It also manages propioceptive sensors such as odometers, and battery power. It also sends messages to the robot actuators.

The protocol to communicate with OTOS servers is mainly based on direct subscription, that is, once the client has connected it can subscribe to a particular sensor and the sender periodically sends the new data.

OTOS server has been implemented using five different threads, as shown in figure 8. One is devoted to attend new clients; a second one serves already connected clients. The other ones are devoted to listen measurements for the different sensors. Each client can subscribe to those sensor it is interested on.

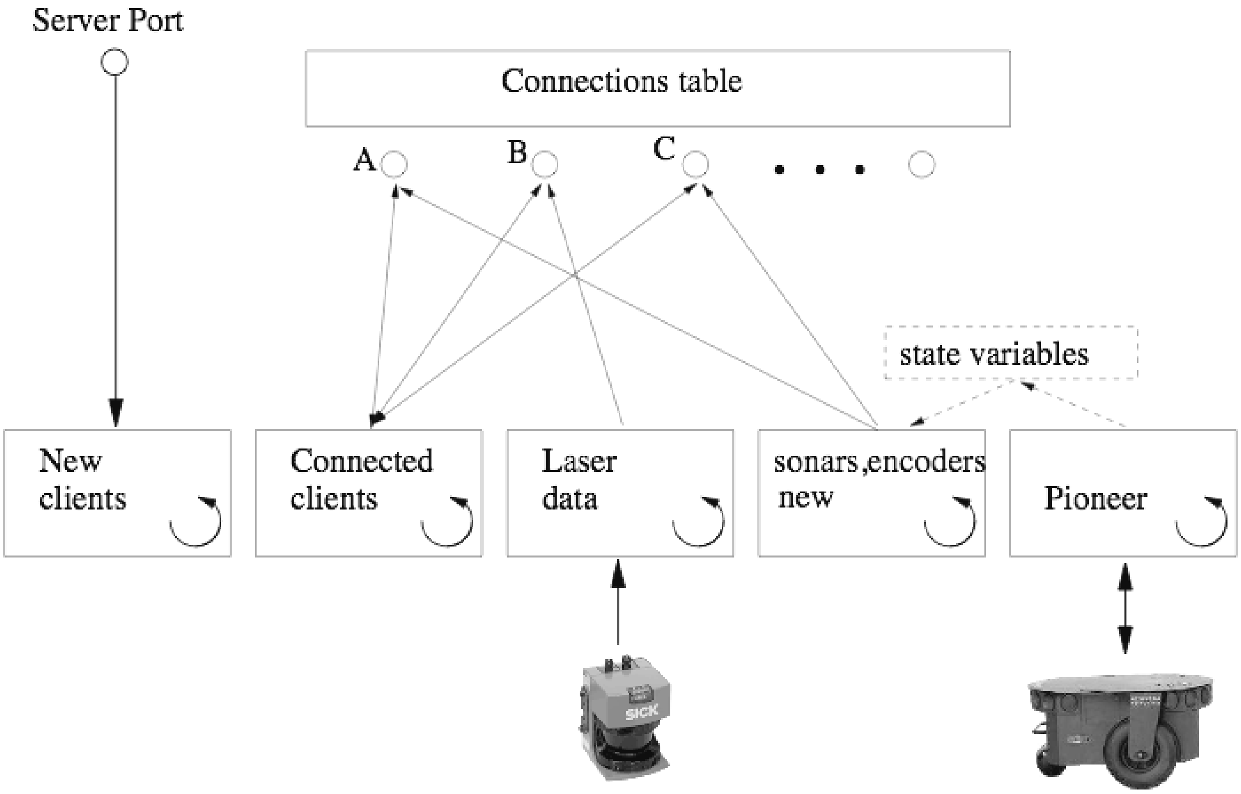


Figure 8. OTOS implementation using 5 different threads

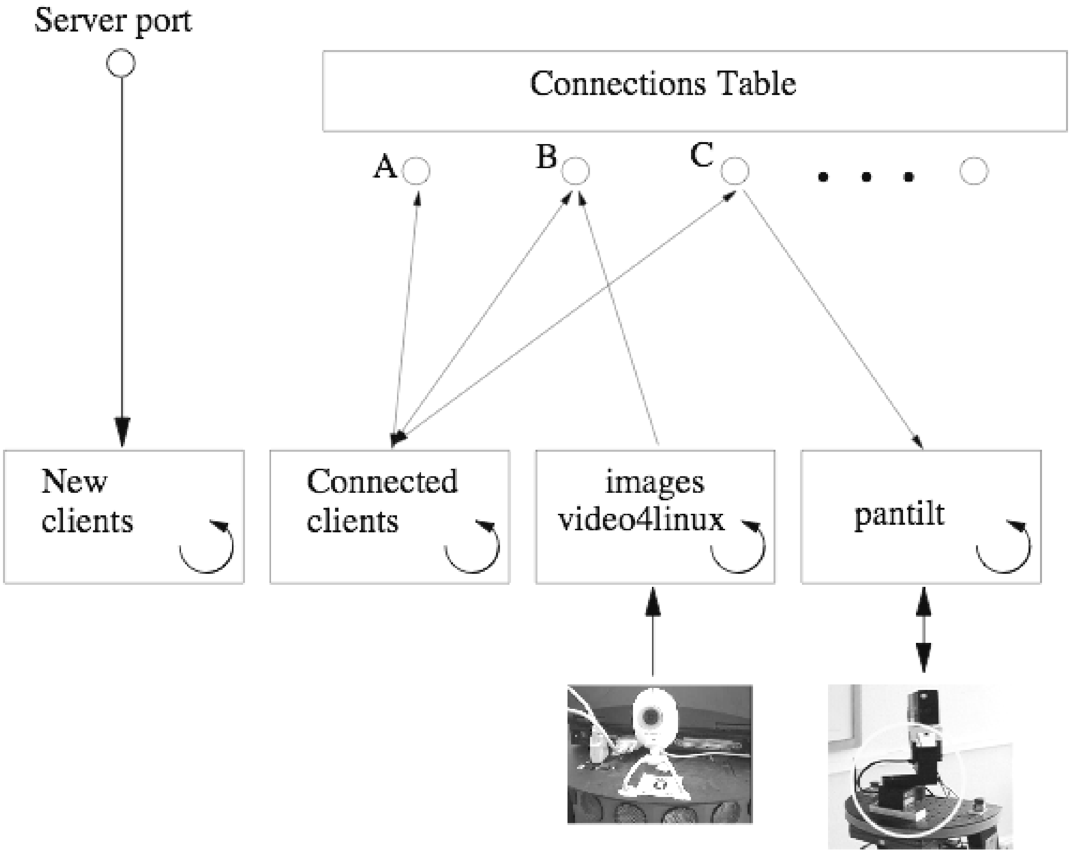


Figure 9. OCULO implementation using 4 different threads

OCULO server manages the functions related to the camera and the pan-tilt unit. It lets clients send messages to point the camera, and it sends them the images flow. In this case the images are send under demand, that is, each time the client needs an image, it has to ask for it.

OCULO has been implemented using 4 threads, as shown in figure 9. As in the OTOS case, two threads are devoted to attend new clients and to interact with already connected ones. The other two manage the interaction with the camera and the pan-tilt unit. Communication with the cameras has been implemented using the video4linux standard.

Developing an application over JDE servers is the more flexible way of using JDE, however it is the hardest way of using it. The alternative is to use the service schemas provided in the distribution.

4.2 Service Schemas

From the theoretical point of view of JDE, everything should be implemented as a schema. It is not just a fundamentalism issue; it is a practical one. It is easier to implement complex behavior using schemas than as traditional programs.

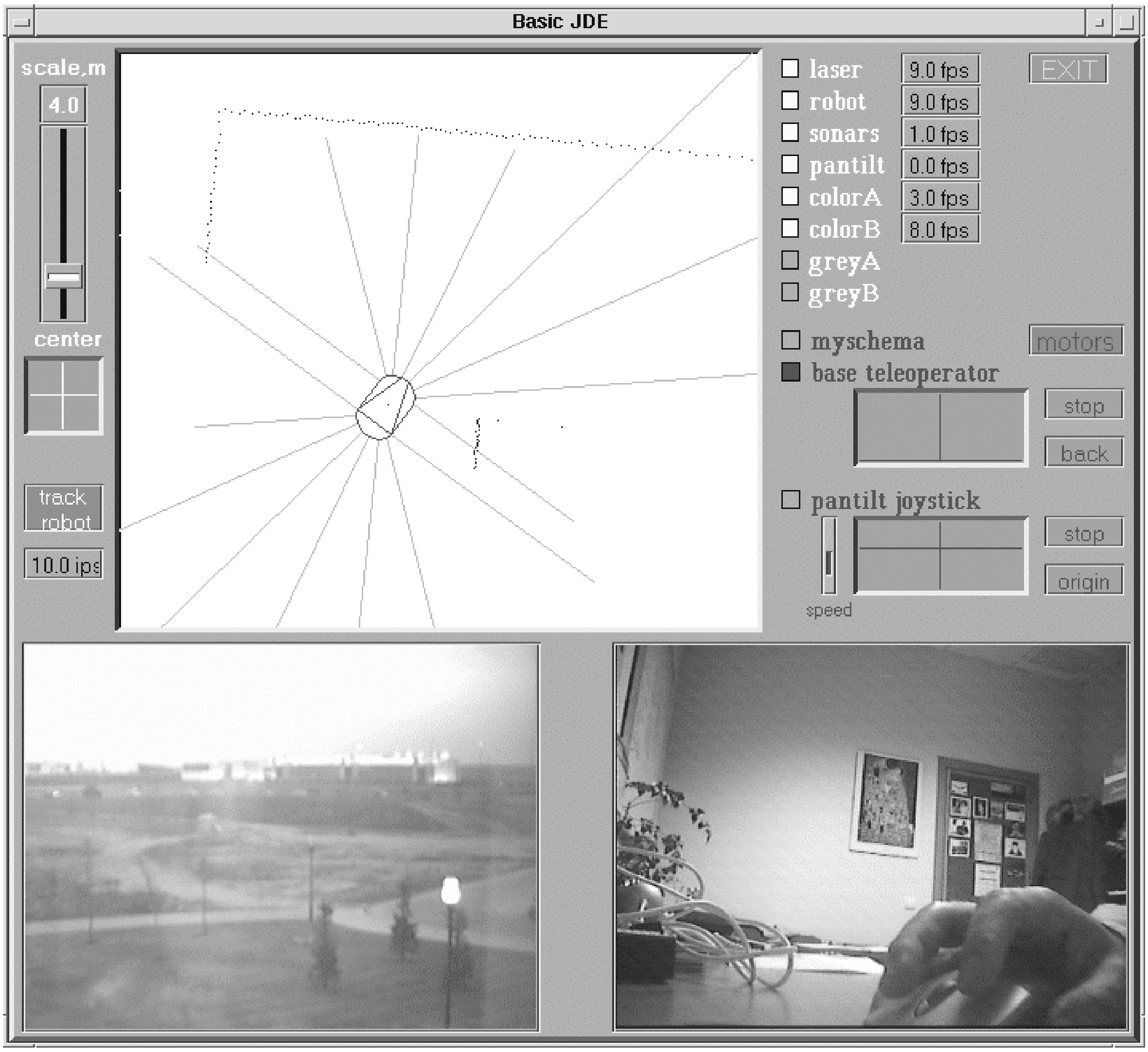


Figure 10. JDE GUI connected to a Pioneer and two different cameras (off-board robot)

JDE distribution includes schema skeletons to make easy the construction of new schemas, and it also includes completely implemented schemas, named "service schemas" to facilitate the interaction with JDE servers.

Getting sensor measurements in JDE using schemas is as simple as reading a local variable, and ordering motor commands as easy as writing an actuator variable.

A set of basic schemas updates those sensor variables with fresh readings and implements such actuator variables. They connect to real sensors and actuators, directly on local devices or through remote socket servers. Pioneer and B21 robots, SRIsim and Stage simulators, video4linux cameras, firewire cameras, Directed Perception pan-tilt units, and SICK laser scanners are fully supported in current version of JDE.

Five *service schemas* are included in the JDE distribution. "Pan-tilt-motors" and "base-motors" periodically translate the actuation variables (like translation *v* and rotation *w* speeds) into actuation messages to the servers. "Oculo-sensations" and "otos-sensations" receive network messages from the servers and update the variables that store corresponding sensor readings. Finally, "gui-xforms" displays sensor readings, internal states and allows the explicit activation from the GUI (Figure 10). It is useful for debugging and monitoring. Applications in JDE can be written, either using the JDE servers, or using the schemas. In both cases, some auxiliary tools are available.

4.3 Auxiliary Tools

A very useful client named *Record* has also been included in JDE release. This client stores all sensorial data in a file, as shown in the left part of the figure 8. This client is used together with the *Replay* client. This client replaces the OTOS and OCULO server reproducing off-line the data stored by *Record* client. This couple is very useful for instance when we want to compare different algortihms, or to debug a schema. The *Replay* server can also reproduce data at different speeds (1x, 2x, etc.). Both clients are based on time stamps with a resolution of microseconds.

Another useful client interacting with OTOS y OCULO server is the *Monitor*. This client continiously shows the values received from the sensors for a human operator. This client also provides a visual joystick to teleoperate the robot.

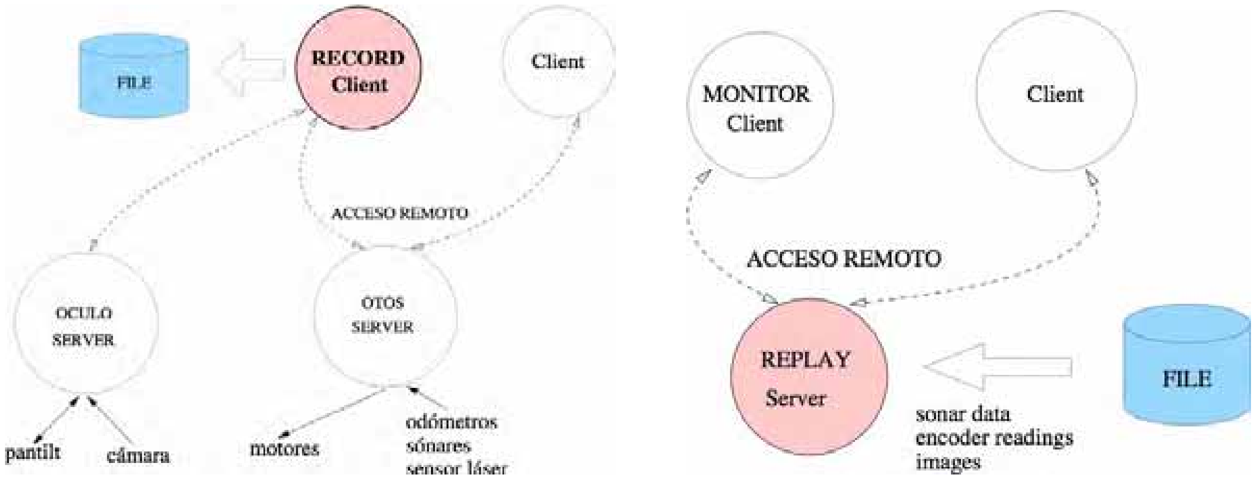


Figure 11. Record client (left) and Replay server (right) for off-line work

Various clients of this type can be used simultaneously. In the same way, it can be used concurrently with other clients that were implementing the behavior of the robot. This monitor can be switched on and off according to the needs of the debugging, saving computer power or resources if needed.

Auxiliary libraries have been also written for grid manipulations (*gridslib*) and fuzzy control (*Fuzzylib*), to make easier the behavior generation. In the same way, various tools have been developed to ease the development of the behaviors.

For instance, *ReplayServer* provides sensor data (sonar, laser, encoders, images, etc.) previously recorded through the same protocol, which is very convenient for off-line work. *HSItuner* in the same way can be used to tune the values of color filter in HSI space.

4.4 Implementing JDE Schemas

JDE schemas have been implemented as independent kernel threads (we have used Pthreads library), so they run concurrently. At the initialization time, all threads are started but immediately stopped to "slept" state, waiting for someone to awake them. The parent thread can resume the execution of a child thread or require it to stop, providing room for the selective activation.

All schemas assume iterative execution. The frequency of the iterations determines the schema time scale, and it is enforced with intentional delays between consecutive iterations. So, the computing power demand comes in periodic bursts.

The iterative execution is a discrete approximation of the continuous feedback of the theoretical model, and a good one if period is fast enough compared to the events in the surroundings (Nilsson 1994).

The schemas communicate each others through selective activation, stimuli, and modulation parameters. The last two are implemented through shared variables. As long as several schemas may access to such data concurrently, locks are used to protect them and avoid race conditions. Each actuation schema includes a *preconditions function*, which at every iteration checks whether such schema should promote to "ready" state or not. In addition, the parent provides each of its children a *list of brothers* and a pointer to the *arbitration function*.

Every kid checks its own preconditions and its brothers' state to detect any control collision or absence. If its own preconditions are not satisfied, it searches whether there is another "active" brother. If not, then it calls the arbitration function to resolve the control absence. If its preconditions hold, then it searches whether there is another "ready" or "active" brother, and in such a case calls the arbitration function to resolve that control collision. If it is the only "ready" or "active" child, it simply executes its specific code for that iteration.

The arbitration function is already implemented in the father's code and the caller is indicated whether it must promote to "active" or remain in "ready" state. This way the arbitration is configured by the father, but it takes place when a child needs it.

5. Conclusions and Further Research

An architecture for behaviour generation named JDE has been presented. Its distinguish characteristics are its hierarchical nature and the use of schemas as the basic building blocks. Its hierarchy model is taken from ethology and intended as a predisposition of the system to perceive certain stimuli and to response in a certain way to them.

JDE explicitly introduces perception into the architectural model, closely tied to action, offering also a selective attention mechanism. The action selection also takes benefit from the hierarchical organization as it is decomposed into several small competitions. This bounds its complexity and allows flexible coordination of the schemas.

One of the weak points in this architecture is that it lacks of high level reasoning that allows predictions about the future. There is no explicit symbolic deliberation in the classical sense. So, all relevant data for the behaviour must be anticipated by the designer, in particular the corresponding stimuli and perceptive schemas have to be allocated in advance. All relevant situations must be anticipated by the designer and so the corresponding actuation schemas to deal with them allocated in advance. In a similar way, perception in JDE can be seen as propositional calculus, but no as first order predicate logic.

We are currently working in the introduction of active perception into JDE. The proposal consists in a perceptive schema having actuation children whose goal is to help in the perception process of its father. More schemas to increase the behaviors pool and a new JDE software implementation are also under development.

We are committed to general platforms, we think that the only that claims made by robotic researchers can be checked is if it can be reproduced. In this way JDE has support for general protocols (for instance video4linux) and it has been developed first on B21 robots and later for ActivMedia family of robots (Pioneer, etc.) using Aria. In the same way, we think that the only way to really check the developments is by releasing programs as *libre*[11] software.

All the code developed to implement JDE has been developed in C language for GNU/Linux machines, and is released under GPL. There are GNU/Linux Debian packages available for i386 architectures in http://gsyc.escet.urjc.es/robotica/software.

It can be directly installed in a Debian based Linux using apt-get adding the following lines to the sources.list file:

```
deb http://gscy010.dat.escet.urjc.es/debian dist main
deb-src http://gscy010.dat.escet.urjc.es/debian dist main
```

6. References

ActivMedia (2002). ARIA Reference Manual (v. 1.1.10).

J.S. Albus (1999). The engineering of mind. *Information Sciences*, Vol. 117, no. 1-2, pp. 1-18.

R.C. Arkin, M. Fujita, T. Takagi and R. Hasegawa (2003), An ethological and emotional basis human-robot interaction, *Robotics and Autonomous Systems*, vol 42, pp. 191-201.

R.C. Arkin (1989), Motor schema-based mobile robot navigation, *International Journal of Robotics Research*, vol 8, no 4, pp. 92-112.

R. Arkin (1998), *Behavior-based robotics*, MIT Press, 1998.

S. Behnke and R. Rojas (2001), A hierarchy of reactive behaviors handles complexity, *Balancing reactivity and social deliberation in multi-agent systems*, LNCS 2103, pp. 125-136

R.P. Bonasso, R.J. Firby, E. Gat, D. Kortenkamp, D.P. Miller and M.G. Slack (1997), Experiences with an architecture for intelligent reactive agents, *Journal of Experimental and Theoretical Artificial Intelligence*, vol. 9, no. 2, pp. 237-256.

[11] *Libre* is the word in Spanish for free as in “free speech“, to avoid misunderstanding in English with “free beer“

R.A. Brooks (1986), A robust layered control system for a mobile robot, *IEEE Journal of Robotics and Automation*, vol. 2, no. 1, March, pp. 14-23.

H. Bruynincks (2001), Open Robot Control Software: the OROCOS project, *Proceedings of the 2001 IEEE/RJS International Conference on Intelligent Robots and Systems (IROS-2001)*, vol. 3, pp. 2523-2528.

J.J. Bryson and L.A. Stein (2001), Modularity and design in reactive intelligence'', in *Proceedings of the 17th Int. Joint Conf. on Artificial Intelligence*, pp. 1115-1120.

J.M Cañas, and V. Matellán (2002). "Dynamic schema hierarchies for an autonomous robot". Advances in Artificial Intelligence (IBERAMIA-02). Lecture Notes in Artificial Intelligence (LNAI-2527), pp. 903-912.

J.M. Cañas(2003), *Jerarquía dinámica de esquemas para la generación de comportamiento artificial*, PhD dissertation, Universidad Politécnica de Madrid.

C. Cote, D. Létourneau, F. Michaud, J. M. Valin, Y. Brosseau, C. Raïevsky, M. Lemay, and V. Tran (2004). "Code reusability tools for programming mobile robots". *Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-04).*

R. J. Firby (1992), Building symbolic primitives with continuous control routines'', in *Proceedings of the Int. Conf. on AI Planning Systems*, AIPS'92, pp. 62-69.

R. J. Firby (1994), "Task networks for controlling continuous processes". Proceedings of the 2nd International Conference on AI Planning Systems, pp. 49-54.

B. P. Gerkey, R. Vaughan, and A. Howard (2003). "The Player/Stage project: tools for multi-robot and distributed sensor systems". *Proceedings of the 11th International Conference on Advanced Robotics (ICAR-03)*, pp. 317-323.

M. Hattig, I. Horswill , and J. Butler (2003), "Roadmap for mobile robot specifications", *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-2003)*, vol. 3, pp. 2410-2414.

I. Horswill (1997), Visual architecture and cognitive architecture, *Journal of Experimental and Theoretical Artificial Intelligence*, vol. 9, no. 2, 1997, pp. 277-292.

K. Konolige and K.L. Myers (1998), The Saphira architecture for autonomous mobile robots, *Artificial Intelligence and Mobile Robots: case studies of successful robot systems*, MIT Press, pp. 211-242.

K. Lorenz (1981), *Fundations of Ethology*, Springer Verlag, New York, Wien; 1981.

M.A. Lozano, I. Iborra, D. Gallardo (2001). "Entorno Java para la simulación de robots móviles". Actas de la IX Conferencia de la Asociación Española para la Inteligencia Artificial CAEPIA-01. pp.1249-1258.

F. Martín, R. González, J.M. Cañas, and V. Matellán (2004), "Programming model based on concurrent objects for the Aibo robot", *Actas de las XII Jornadas de Concurrencia y Sistemas Distribuidos*, pp. 367-379.

M. Montemerlo, N. Roy and S. Thrun. "Perspectives on standardization in mobile robot programming: The Carnegie Mellon Navigation CARMEN Toolkit". *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-03)*. Vol. 3, pp. 2436-2441.

M.N. Nicolescu and M.J. Mataric (2002), ``A hierarchical architecture for behavior-based robots'', in *Proceedings of the 1st Int. Joint Conf. on Autonomous Agent*s and Multiagents Systems, AMAAS'02, pp. 227-233.

N. J. Nilsson (1984), Shakey the robot, SRI Technical Report 323, April 1984.

N. J. Nilsson (1997), Visual architecture and cognitive architecture, *Journal of Experimental and Theoretical Artificial Intelligence*, vol. 9, no. 2, pp. 277-292.

A. Saffiotti and Z. Wasik (2003), Using hierarchical fuzzy behaviors in the RoboCup domain, *Autonomous Robotic Systems*, Springer, 2003, pp. 235-262.

R.G. Simmons (1994), Structured control for autonomous robots, *IEEE Journal of Robotics and Automation*, vol. 10, no. 1, February, pp. 34-43.

R.G. Simmons, D. Apfelbaum (1998). "A Task Description Language for robot control". Proceedings of the 1998 IEEE/RJS International Conference on Intelligent Robots and Systems (IROS-98). Vol. 3. pp.1931-1937.

N. Tinbergen (1951), *The study of instinct*, Oxford University Press, London; 1951.

T. Tyrrell (1994), The use of hierarchies for action selection, *Journal of Adaptive Behavior*, vol. 2, no. 4, pp. 307-348.

H. Utz, S. Sablatnög, S. Enderle, and G. Kraetzschmar (2002), "MIRO – Middleware for mobile robot applications, *IEEE Transactions on Robotics and Automation*. Special Issue on Object-Oriented Distributed Control Architectures. Vol. 18, no. 4, pp. 493-497.

R. T. Vaughan, B. P. Gerkey, and A. Howard (2003). "On device abstractions for portable, reusable robot code". Proceedings of the 2003 IEEE/RJS International Conference on Intelligent Robots and Systems (IROS-2003). Vol. 3 pp. 2121-2127.

E. Woo, B. A. MacDonald, and F. Trépanier. "Distributed mobile robot application infrastructure". *Procceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-03)*. Las Vegas (USA), Vol 3, pp. 2410-2414.

Stabilization of Fuzzy Takagi - Sugeno Descriptor Models; Application to a Double Inverted Pendulum

Thierry Marie Guerra, Sebastien Delprat & Salim Labiod

1. Introduction

Takagi-Sugeno (TS) fuzzy models (Takagi & Sugeno 1985) have been widely used in the context of control and observation of nonlinear models, see for example (Tanaka & Wang 2001) and the references therein. They consist in a collection of linear models blended together with nonlinear membership functions. Their capability to represent exactly in a compact set of the state variables a nonlinear model makes them attractive for control and observation (Taniguchi et al. 2001). The stability and the stabilization of such models (including performances and/or robustness considerations) are mainly investigated through Lyapunov functions (Chen et al. 2000, Joh et al. 1997, Tanaka et al. 1996, Tanaka et al. 1998, Tong et al. 2002, Tuan et al. 2001, Zhao 1995). These ones are most of the time quadratic ones, nevertheless interesting results can also be found using piecewise quadratic functions (Feng 2003, Johansson et al. 1999) or non quadratic Lyapunov functions (Blanco et al. 2001, Guerra & Vermeiren 2004, Tanaka et al. 2001). At last, TS fuzzy descriptors have been studied for the stabilization and the observation points of view and some results are given in (Guerra et al. 2004, Tanaka & Wang 2001, Taniguchi et al. 2000). Most of the time an interesting way to solve the different problems addressed is to write the obtained conditions in a LMI form (Linear Matrix Inequalities) (Boyd et al. 1994).

There is a systematic way, called the sector nonlinearity approach (Tanaka & Wang 2001) to go from a nonlinear model affine in the control to a Takagi Sugeno fuzzy model. The number of linear models r of the nonlinear TS fuzzy model grows exponentially, i.e. in 2^{nl}, with nl the number of nonlinearities to be treated (Tanaka & Wang 2001, Taniguchi et al. 2001). In the stabilization framework, the conditions of stabilization will only depend on the linear models of the TS models, i.e. the nonlinear membership functions blending the linear models together are not used. Of course, this remains in conservative results. Let us also point out that the number of conditions to be solved in the LMI problems is directly related to the number r of linear models of the TS fuzzy model. We can emphasize that the more r is big, the more the results obtained will be conservative. Thus it is of high interest to obtain a TS representation of nonlinear models with a reduced number of rules. Another remark can be formulated. For some models, mechanical ones for example, a descriptor form can be interesting to keep a TS model structure closed to the nonlinear one. In some cases, we will show that it allows reducing the number of rules and therefore it can improve, in an interesting way, the results. Nevertheless using this specific descriptor structure, it is then necessary to derive conditions of stabilization. Some results can be found in (Tanaka & Wang 2001, Taniguchi et al. 2000). Hence, this work focuses on continuous TS fuzzy models using a descriptor form and the stabilization results will be studied through a quadratic Lyapunov function.

The chapter is organized as follows. The first part gives the notations used and the material necessary to derive the results. It includes some specific matrix properties and basic properties of TS models. The second part presents the statement of the problem using TS models in a descriptor form and an example to show their interest. The third part gives the main result obtained for stabilization. The goal is to use matrix properties in order to reduce the conservatism of the basic conditions. A first academic example is given to show the different results. At last the application of this methodology to the double-inverted pendulum is presented.

2. Notations and Material

Let us consider positive scalar functions $h_i(\cdot) \geq 0$, $i \in \{1,\ldots,r\}$ and $v_k(\cdot) \geq 0$ $k \in \{1,\ldots,e\}$ satisfying the convex sum property:

$$\sum_{i=1}^{r} h_i(\cdot) = 1,\ \sum_{k=1}^{e} v_k(\cdot) = 1 \tag{1}$$

With such functions and some matrices of appropriate dimensions Y_i we define the following notations: $Y_h = \sum_{i=1}^{r} h_i(z(t)) Y_i$, $Y_{hh} = \sum_{i=1}^{r}\sum_{j=1}^{r} h_i(z(t)) h_j(z(t)) Y_{ij}$, $Y_v = \sum_{k=1}^{e} v_k(z(t)) Y_k$,

$$Y_{vh} = \sum_{k=1}^{e}\sum_{i=1}^{r} v_k(z(t)) h_i(z(t)) Y_{ik}, \text{ and so on}$$

As usual, a star $(*)$ in a symmetric matrix indicates a transpose quantity. Congruence of a symmetric definite positive matrix $P = P^T > 0$ with a full rank matrix Y corresponds to the following quantity: $YPY^T > 0$.

We will also use the following lemma. It is a slightly modified version of a property given in (Peaucelle et al. 2000) and also used in the context of Takagi-Sugeno fuzzy models stabilization (Guerra et al. 2003).

Lemma 1:

Let P, Y, Γ, Φ and Ψ be matrices of appropriate dimensions the two following properties are equivalent.

$$P^T\Gamma^T + \Gamma P + Y < 0 \tag{2}$$

$$\text{It exists } \Phi \text{ and } \Psi \text{ such that:} \begin{bmatrix} \Phi^T\Gamma^T + \Gamma\Phi + Y & (*) \\ P - \Phi + \Psi^T\Gamma^T & -\Psi - \Psi^T \end{bmatrix} < 0 \tag{3}$$

Proof:

(3) implies (2): the result is obtained using the congruence with $\begin{bmatrix} I & \Gamma \end{bmatrix}$.

(2) implies (3): As $P^T\Gamma^T + \Gamma P + Y < 0$, it always exists an enough small ε^2 such that:

$$P^T\Gamma^T + \Gamma P + Y + \frac{\varepsilon^2}{2}\Gamma^T\Gamma < 0 \tag{4}$$

Using the Schur's complement, (4) is equivalent to:

$$\begin{bmatrix} P^T\Gamma^T + \Gamma P + Y & \varepsilon^2\Gamma^T \\ \varepsilon^2\Gamma & -2\varepsilon^2 I \end{bmatrix} < 0 \tag{5}$$

If we choose $\Phi = P$ and $\Psi = \varepsilon^2 I$, then the first inequality of (2) holds.

Remark 1:

If Φ or Ψ are under constraint, the equivalence is not more true.

Most of the LMI problems encountered for TS stabilization can be resumed in the following way. For a given k, with Υ_{ij}^{k}, $i, j \in \{1,\ldots,r\}$ expressions being independent from time, find the best conditions, i.e. in the sense of reducing the conservatism, to the problem:

$$\sum_{i=1}^{r}\sum_{j=1}^{r} h_i\left(z(t)\right) h_j\left(z(t)\right)\left(\Upsilon_{ij}^{k} + \Upsilon_{ji}^{k}\right) < 0 \tag{6}$$

Several results are available, going from the very simple one (Tanaka et al. 1998):

$$\Upsilon_{ii}^{k} \le 0,\ \Upsilon_{ij}^{k} + \Upsilon_{ji}^{k} \le 0,\ i, j \in \{1,\ldots,r\},\ j > i \tag{7}$$

to very specific matrix transformations (Kim & Lee 2000, Teixeira et al. 2003, Liu & Zhang 2003). Let us point out that whatever the relaxations are they can be used on the Υ_{ij}^{k}. We will just give the one of (Liu & Zhang 2003) that seems to be a good compromise between complexity and number of variables involved in the LMI problem. The work presented in (Teixeira et al, 2003) can also be quoted, but it implies a serious increase of the number of variables involved in the problem.

Lemma 2 (Liu & Zhang 2003):

With Υ_{ij}^{k}, $i, j \in \{1,\ldots,r\}$ matrices of appropriate dimension, (6) holds if there exist matrices $Q_{ii}^{k} > 0$ and $Q_{ij}^{k} = \left(Q_{ji}^{k}\right)^{T}$, $i, j \in \{1,\ldots,r\}$ $j > i$ such that the following conditions are satisfied:

$$\Upsilon_{ii}^{k} + Q_{ii}^{k} < 0 \tag{8}$$

$$\Upsilon_{ij}^{k} + \Upsilon_{ij}^{k} + Q_{ij}^{k} + Q_{ji}^{k} \le 0 \tag{9}$$

$$Q^{k} = \begin{bmatrix} Q_{11}^{k} & (*) & & (*) \\ Q_{12}^{k} & Q_{22}^{k} & & \\ \vdots & & \ddots & (*) \\ Q_{1r}^{k} & \cdots & Q_{(r-1)r}^{k} & Q_{rr}^{k} \end{bmatrix} > 0 \tag{10}$$

The models under consideration in this chapter are the so-called Takagi-Sugeno's ones (Takagi & Sugeno 1985). They correspond to linear models blended with nonlinear functions (12). They can represent exactly a large class of affine nonlinear models in compact region of the state space (Tanaka & Wang 2001, Taniguchi et al. 2001). From a nonlinear model with $x(t)$ the state, $u(t)$ the input vector and $y(t)$ the output vector:

$$\begin{cases} \dot{x}(t) = f\left(x(t)\right) + g\left(x(t)\right)u(t) \\ y(t) = h\left(x(t)\right) \end{cases} \tag{11}$$

there exists a systematic way called the sector nonlinearity approach (Tanaka & Wang 2001) to put it into a TS form (see example 1 hereinafter):

$$\begin{cases} \dot{x}(t) = \sum_{i=1}^{r} h_i\left(z(t)\right)\left(A_i x(t) + B_i u(t)\right) = A_z x(t) + B_z u(t) \\ y(t) = \sum_{i=1}^{r} h_i\left(z(t)\right)\left(C_i x(t)\right) = C_z x(t) \end{cases} \tag{12}$$

with r the number of linear models, $z(t)$ a vector which depends linearly or not on the state, $h_i\left(z(t)\right) \ge 0$, $i \in \{1,\ldots,r\}$ nonlinear functions verifying the convex sum property (1).

The number r of linear models grows exponentially according to the number of nonlinearities to be treated in the model (11) (Tanaka et al. 1998). Note also that the TS representation of (11) is not unique (Taniguchi et al. 2001).

In order to stabilize this kind of models, classically the control law used is the Parallel Distributed Compensation (PDC) (Wang et al. 1996). The expression of this control law is given by:

$$u(t) = -\sum_{i=1}^{r} h_i\left(z(t)\right) F_i x(t) = -F_z x(t) \tag{13}$$

Basic results of stabilization of TS models with a PDC control law can be found in (Wang et al. 1996).

At last looking at the problem (6), the results do not depend on the nonlinear functions $h_i\left(z(t)\right)$ and then can lead to a strong conservatism. Thus it is of high interest to find new ways to reduce this conservatism. One way can be to use other Lyapunov functions (Feng 2003, Guerra & Vermeiren 2004, Johansson et al. 1999). The way explored in this chapter is to use a descriptor form of TS fuzzy models.

3. Statement of the Problem

Let us consider a fuzzy descriptor model as (Taniguchi & al. 2001):

$$\sum_{k=1}^{e} v_k\left(z(t)\right) E_k \dot{x}(t) = \sum_{i=1}^{r} h_i\left(z(t)\right)\left(A_i x(t) + B_i u(t)\right), \text{ or:}$$

$$\begin{cases} E_v \dot{x}(t) = A_h x(t) + B_h u(t) \\ y(t) = C_h x(t) \end{cases} \tag{14}$$

In the following we suppose that the problem is always well formulated, hypothesis 1.

Hypothesis 1:

$$\text{For all } z(t),\ \sum_{k=1}^{e} v_k\left(z(t)\right) E_k \neq 0 \tag{15}$$

Defining $x^*(t) = \left[x^T(t), \dot{x}^T(t)\right]^T$, the system (14) can be written as:

$$\begin{cases} E^* \dot{x}(t) = A_{hv}^* x^*(t) + B_h^* u(t) \\ y(t) = C_h^* x^*(t) \end{cases} \tag{16}$$

where:

$$E^* = \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix},\ A_{ik}^* = \begin{bmatrix} 0 & I \\ A_i & -E_k \end{bmatrix},\ B_i^* = \begin{bmatrix} 0 \\ B_i \end{bmatrix},\ C_i^* = \begin{bmatrix} C_i & 0 \end{bmatrix}.$$

Consider a modified PDC control law (Taniguchi et al. 2000):

$$u(t) = -\sum_{i=1}^{r}\sum_{k=1}^{e} h_i\left(z(t)\right) v_k\left(z(t)\right) F_{ik} x(t) \tag{17}$$

then, introducing (17) in (16), with $F_{hv}^* = \begin{bmatrix} F_{hv} & 0 \end{bmatrix}$ leads to:

$$\begin{cases} E^* \dot{x}(t) = \left(A_{hv}^* - B_h^* F_{hv}^*\right) x^*(t) \\ y(t) = C_h^* x^*(t) \end{cases} \tag{18}$$

According to the work of (Taniguchi & al. 2001) the following theorem conditions ensure the fuzzy descriptor to be quadratically stable.

Theorem 1:

The fuzzy descriptor model (18) is quadratically stable if there exists a common matrix X such that:

$$E^{*}X = X^{T}E^{*} \geq 0 \tag{19}$$

$$\left(A_{hv}^{*} - B_{h}^{*}F_{hv}^{*}\right)^{T} X + X^{T}\left(A_{hv}^{*} - B_{h}^{*}F_{hv}^{*}\right) < 0 \tag{20}$$

Proof:

It is straightforward considering the following Lyapunov candidate function:$V\left(x^{*}(t)\right) = x^{*T}(t)E^{*T}Xx^{*}(t)$.

The goal is now to propose LMI conditions for ensuring to find X and the gains F_{ik}. Let us define: $X = \begin{bmatrix} X_1 & X_2 \\ X_3 & X_4 \end{bmatrix}$, condition (19) implies: $X_1 = X_1^T \geq 0$ and $X_2 = 0$. Then condition (20) can be written as:

$$\begin{bmatrix} 0 & A_h^T - F_{hv}^T B_h^T \\ I & -E_v^T \end{bmatrix}\begin{bmatrix} X_1 & 0 \\ X_3 & X_4 \end{bmatrix} + \begin{bmatrix} X_1 & X_3^T \\ 0 & X_4^T \end{bmatrix}\begin{bmatrix} 0 & I \\ A_h - B_h F_{hv} & -E_v \end{bmatrix} < 0 \tag{21}$$

With X_4 non-singular, we have: $X^{-1} = \begin{bmatrix} X_1^{-1} & 0 \\ -X_4^{-1}X_3X_1^{-1} & X_4^{-1} \end{bmatrix} = \begin{bmatrix} P_1 & 0 \\ P_3 & P_4 \end{bmatrix}$, and after congruence with X^{-T} we obtain:

$$\begin{bmatrix} P_1 & P_3^T \\ 0 & P_4^T \end{bmatrix}\begin{bmatrix} 0 & A_h^T - F_{hv}^T B_h^T \\ I & -E_v^T \end{bmatrix} + \begin{bmatrix} 0 & I \\ A_h - B_h F_{hv} & -E_v \end{bmatrix}\begin{bmatrix} P_1 & 0 \\ P_3 & P_4 \end{bmatrix} < 0 \tag{22}$$

Let us rewrite (22) as:

$$\begin{bmatrix} P_3^T + P_3 & (*) \\ P_4^T + A_h P_1 - B_h F_{hv} P_1 - E_v P_3 & -P_4^T E_v^T - E_v P_4 \end{bmatrix} < 0 \tag{23}$$

Let us define with $M_{ik} = F_{ik} P_1$:

$$\Upsilon_{ij}^{k} = \begin{bmatrix} P_3^T + P_3 & (*) \\ P_4^T + A_i P_1 - B_i M_{jk} - E_k P_3 & -P_4^T E_k^T - E_k P_4 \end{bmatrix} \tag{24}$$

The following theorem gives the result.

Theorem 2:

Let us consider TS the fuzzy descriptor model (18), the Υ_{ij}^{k} defined in (24). The TS fuzzy descriptor with control law (17) is quadratically stable if there exists matrices: $P_1 = P_1^T > 0$, P_3, P_4 regular, M_{ik}, such that for each $k \in \{1,\ldots,e\}$ and $i, j \in \{1,\ldots,r\}$, $j > i$ the conditions given equation (7) hold. Moreover the gains of the control law are given by $F_{ik} = M_{ik}P_1^{-1}$.

Remark 2:

As stated previously, any usual relaxation can be used. For example with the one presented before (Liu & Zhang 2003) the result will be: the TS fuzzy descriptor with control law (17) is quadratically stable if there exists matrices: $P_1 = P_1^T > 0$, P_3, P_4 regular, M_{ik}, $Q_{ii}^{k} > 0$ and $Q_{ij}^{k} = \left(Q_{ji}^{k}\right)^T$ such that for each $k \in \{1,\ldots,e\}$ and each $i, j \in \{1,\ldots,r\}$, $j > i$ the conditions given equations (8), (9) and (10) hold. Note also that the number of LMI conditions obtained in both case (excepted (10)) is: $e \cdot \frac{r \cdot (r+1)}{2}$.

Remark 3:

These first conditions include those presented in (Taniguchi et al. 2000).

To show the interest of this descriptor form formulation we will study a first academic example.

Example 1:

Consider the following nonlinear model with $x(t)=\begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix}$ the state vector:

$$E(x(t))\cdot \dot{x}(t) = A\cdot \dot{x}(t) + B(x_1(t))\cdot u(t) \tag{25}$$

with: $A=\begin{bmatrix} -1 & -1 \\ 2 & 6 \end{bmatrix}$, $E(x(t))=\begin{bmatrix} \dfrac{1}{1+x_1^2(t)} & -1 \\ 1 & \dfrac{1}{1+x_2^2(t)} \end{bmatrix}$ and $B(x_1(t))=\begin{bmatrix} 2+\cos(x_1) \\ 1 \end{bmatrix}$.

A TS descriptor in the form of (14) can be obtained. For that, note that there are two nonlinearities in $E(x(t))$ that leads to $e=4$ and one in the right side of (25) which gives $r=2$. To explicit the way to obtain a TS form, we consider the function $f(x)=\dfrac{1}{1+x^2}$. For $x\in\Re$ it is easy to check that $f(x)$ belongs to $[0,1]$ then we can write: $f(x)=w_1(x).1+w_2(x).0$ with $w_1(x)=\dfrac{1}{1+x^2}$ and $w_2(x)=\dfrac{x^2}{1+x^2}$. Note that the $w_i(x)$ $i=1,2$ are positive functions and satisfy the convex sum property: $w_1(x)+w_2(x)=1$. Thus using that decomposition for $E(x(t))$ leads to four models, i.e.:

$$E(x)=w_1(x_1)w_1(x_2)\begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}+w_1(x_1)w_2(x_2)\begin{bmatrix} 1 & -1 \\ 1 & 0 \end{bmatrix}+w_2(x_1)w_1(x_2)\begin{bmatrix} 0 & -1 \\ 1 & 1 \end{bmatrix}w_2(x_1)w_2(x_2)\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

. Of course, this way of taking into account the nonlinearities can be applied to all bounded nonlinearities (Morère 2001, Tanaka et al. 1998). Then the number of conditions involved in the LMI problem is $e\cdot\dfrac{r\cdot(r+1)}{2}=12$.

The conditions of theorem 2 give a solution using MATLAB LMI toolbox, this ensures the stabilization of the TS descriptor.

Considering now a classical TS model (12) for (25) will impose to invert $E(x(t))$, i.e.:

$$\dot{x}(t)=E^{-1}(x(t))A\cdot\dot{x}(t)+E^{-1}(x(t))B(x_1(t))\cdot u(t) \tag{26}$$

Note that $E^{-1}(x(t))=\begin{bmatrix} \dfrac{1+x_1^2}{(1+x_1^2)\cdot(1+x_2^2)+1} & \dfrac{(1+x_1^2)\cdot(1+x_2^2)}{(1+x_1^2)\cdot(1+x_2^2)+1} \\ \dfrac{-(1+x_1^2)\cdot(1+x_2^2)}{(1+x_1^2)\cdot(1+x_2^2)+1} & \dfrac{1+x_2^2}{(1+x_1^2)\cdot(1+x_2^2)+1} \end{bmatrix}$, then after some easy but fastidious calculus it can be shown that the four nonlinearities: $1+x_1^2$, $1+x_2^2$, $\dfrac{1}{(1+x_1^2)\cdot(1+x_2^2)+1}$ and $2+\cos(x_1)$ have to be treated to obtain a TS model. This will give a TS model with $r=2^4=16$ linear models and then a LMI problem with 136 LMI! For example, considering a compact set of the state variable: $x_1(t), x_2(t) \in [-6, 6]$ no solution was obtained, even using the relaxation of (Liu & Zhang 2003).

This example clearly shows that keeping the TS form close to the nonlinear model can be helpful. The goal of the next section is to try to reduce the conservatism of the conditions obtained in the theorem 2.

4. Main Result

With the change of variable: $M_{hv} = F_{hv}P_1$, let us rewrite (22) in the following way:

$$\begin{bmatrix} P_1 & P_3^T \\ 0 & P_4^T \end{bmatrix}\begin{bmatrix} 0 & A_h^T \\ I & -E_v^T \end{bmatrix} + \begin{bmatrix} 0 & I \\ A_h & -E_v \end{bmatrix}\begin{bmatrix} P_1 & 0 \\ P_3 & P_4 \end{bmatrix} + \begin{bmatrix} 0 & -M_{hv}^T B_h^T \\ -B_h M_{hv} & 0 \end{bmatrix} < 0 \tag{27}$$

Let us consider the property described in lemma 1, equation (2), i.e. $P^T\Gamma^T + \Gamma P + Y < 0$, with $P = \begin{bmatrix} P_1 & 0 \\ P_3 & P_4 \end{bmatrix}$, $\Gamma = \begin{bmatrix} 0 & I \\ A_h & -E_v \end{bmatrix}$ and $Y = \begin{bmatrix} 0 & -M_{hv}^T B_h^T \\ -B_h M_{hv} & 0 \end{bmatrix}$ then using the equivalence with (3) and $\Phi = \begin{bmatrix} \Phi_1 & \Phi_2 \\ \Phi_3 & \Phi_4 \end{bmatrix}$, $\Psi = \begin{bmatrix} \Psi_1 & \Psi_2 \\ \Psi_3 & \Psi_4 \end{bmatrix}$, condition (22) is satisfied if:

$$\begin{bmatrix} \Phi_3^T + \Phi_3 & (*) & (*) & (*) \\ \Phi_4^T + A_h\Phi_1 - E_v\Phi_3 - B_hM_{hv} & \Phi_2^T A_h^T + A_h\Phi_2 - \Phi_4^T E_v^T - E_v\Phi_4 & (*) & (*) \\ P_1 - \Phi_1 + \Psi_3^T & -\Phi_2 + \Psi_1^T A_h^T - \Psi_3^T E_v^T & -\Psi_1^T - \Psi_1 & (*) \\ P_3 - \Phi_3 + \Psi_4^T & P_4 - \Phi_4 + \Psi_2^T A_h^T - \Psi_4^T E_v^T & -\Psi_3^T - \Psi_2 & -\Psi_4^T - \Psi_4 \end{bmatrix} < 0 \tag{28}$$

Note that the expression (28), due to the term B_hM_{hv} will be at least a triple sum: $\sum_{i=1}^{r}\sum_{j=1}^{r}\sum_{k=1}^{e} h_i(z)h_j(z)v_k(z)$. As Φ and Ψ are unspecified matrices, their degrees of freedom can be extended to this triple sum in the following way:

$$\Phi = \begin{bmatrix} \Phi_{1hv} & \Phi_{2hv} \\ \Phi_{3hh} & \Phi_{4hh} \end{bmatrix} = \sum_{i=1}^{r} h_i(z) \begin{bmatrix} \sum_{k=1}^{e} v_k(z)\Phi_{1ik} & \sum_{k=1}^{e} v_k(z)\Phi_{2ik} \\ \sum_{j=1}^{r} h_j(z)\Phi_{3ij} & \sum_{j=1}^{r} h_j(z)\Phi_{4ij} \end{bmatrix} \tag{29}$$

$$\Psi = \begin{bmatrix} \Psi_{1hv} & \Psi_{2hv} \\ \Psi_{3hh} & \Psi_{4hh} \end{bmatrix} = \sum_{i=1}^{r} h_i(z) \begin{bmatrix} \sum_{k=1}^{e} v_k(z)\Psi_{1ik} & \sum_{k=1}^{e} v_k(z)\Psi_{2ik} \\ \sum_{j=1}^{r} h_j(z)\Psi_{3ij} & \sum_{j=1}^{r} h_j(z)\Psi_{4ij} \end{bmatrix} \tag{30}$$

Remark 4: If the fuzzy descriptor shares the same input matrices, i.e. $B_i = B$, $i \in \{1, \ldots, r\}$ then, of course, $\Phi = \begin{bmatrix} \Phi_{1v} & \Phi_{2v} \\ \Phi_{3h} & \Phi_{4h} \end{bmatrix}$ and $\Psi = \begin{bmatrix} \Psi_{1v} & \Psi_{2v} \\ \Psi_{3h} & \Psi_{4h} \end{bmatrix}$.

A new expression for (28) is then:

$$\begin{bmatrix} \Phi_{3hh}^T + \Phi_{3hh} & (*) & (*) & (*) \\ \begin{pmatrix} \Phi_{4hh}^T + A_h\Phi_{1hv} \\ -E_v\Phi_{3hh} - B_hM_{hv} \end{pmatrix} & \begin{pmatrix} \Phi_{2hv}^T A_h^T + A_h\Phi_{2hv} \\ -\Phi_{4hh}^T E_v^T - E_v\Phi_{4hh} \end{pmatrix} & (*) & (*) \\ P_1 - \Phi_{1hv} + \Psi_{3hh}^T & -\Phi_{2hv} + \Psi_{1hv}^T A_h^T - \Psi_{3hh}^T E_v^T & -\Psi_{1hv}^T - \Psi_{1hv} & (*) \\ P_3 - \Phi_{3hh} + \Psi_{4hh}^T & P_4 - \Phi_{4hh} + \Psi_{2hv}^T A_h^T - \Psi_{4hh}^T E_v^T & -\Psi_{3hh}^T - \Psi_{2hv} & -\Psi_{4hh}^T - \Psi_{4hh} \end{bmatrix} < 0 \tag{31}$$

Let us define:

$$\Upsilon_{ij}^{k} = \begin{bmatrix} \Phi_{3ij}^{T} + \Phi_{3ij} & (*) & (*) & (*) \\ \Phi_{4ij}^{T} + A_i\Phi_{1jk} - E_k\Phi_{3ij} - B_iM_{jk} & \Phi_{2jk}^{T}A_i^{T} + A_i\Phi_{2jk} - \Phi_{4ij}^{T}E_k^{T} - E_k\Phi_{4ij} & (*) & (*) \\ P_1 - \Phi_{1jk} + \Psi_{3ij}^{T} & -\Phi_{2jk} + \Psi_{1jk}^{T}A_i^{T} - \Psi_{3ij}^{T}E_k^{T} & -\Psi_{1jk}^{T} - \Psi_{1jk} & (*) \\ P_3 - \Phi_{3ij} + \Psi_{4ij}^{T} & P_4 - \Phi_{4ij} + \Psi_{2jk}^{T}A_i^{T} - \Psi_{4ij}^{T}E_k^{T} & -\Psi_{3ij}^{T} - \Psi_{2jk} & -\Psi_{4ij}^{T} - \Psi_{4ij} \end{bmatrix} \quad (32)$$

Theorem 3:

Let us consider the fuzzy descriptor model (18) and the Υ_{ij}^{k} defined in (32). The fuzzy descriptor with control law (17) is quadratically stable if there exist matrices $P_1 = P_1^T > 0$, P_3, P_4, M_{ik}, Φ and Ψ defined in (29) and (30) such that for each $k \in \{1,\ldots,e\}$ and $i, j \in \{1,\ldots,r\}$, $j > i$ the conditions (7) (or considering also matrices $Q_{ii}^{k} > 0$ and $Q_{ij}^{k} = \left(Q_{ji}^{k}\right)^T$ $i, j \in \{1,\ldots,r\}$, $j > i$ the conditions (8), (9) and (10)) are satisfied. Moreover the gains of the control law are given by: $F_{ik} = M_{ik}P_1^{-1}$.

Lemma 3:

For any fuzzy descriptor (14), if the conditions of theorem 2 are satisfied then those of theorem 3 are also satisfied.

Proof:

In the proof, the exponent $^{(1)}$ stands for the first approach (theorem 2), the exponent $^{(2)}$ stands for the second one (theorem 3). Suppose that the conditions of theorem 2 are satisfied. Then there exists $P_1 = P_1^T > 0$, P_3, P_4, M_{ik}, $Q_{ii}^{k(1)} > 0$ and $Q_{ij}^{k(1)} = \left(Q_{ji}^{k(1)}\right)^T$ satisfying (8), (9) and (10) (if no relaxation is chosen the proof follows the same path). We keep the same matrices $P_1 = P_1^T > 0$, P_3, P_4, M_{ik}, for the theorem 3 and fix $Q_{ii}^{k(2)} = \begin{bmatrix} Q_{ii}^{k(1)} & 0 \\ 0 & \varepsilon^2 I \end{bmatrix}$, $Q_{ij}^{k(2)} = \begin{bmatrix} Q_{ij}^{k(1)} & 0 \\ 0 & 0 \end{bmatrix}$. Then directly, $Q^{k(2)} > 0$ if and only if: $\begin{bmatrix} Q^{k(1)} & 0 \\ 0 & \varepsilon^2 I_{n\times r} \end{bmatrix} > 0$ which is clearly satisfied as $Q^{k(1)} > 0$. Fix also: $\Phi_{1ik} = P_1$, $\Phi_{2ik} = 0$, $\Phi_{3ij} = P_3$, $\Phi_{4ij} = P_4$, $\Psi_{2ik} = \Psi_{3ij} = 0$, $\Psi_{1ik} = \Psi_{4ij} = \varepsilon^2 I$, $i, j \in \{1,\ldots,r\}$, $j > i$, $k \in \{1,\ldots,e\}$, (32) can be written as:

$$\Upsilon_{ij}^{k(2)} = \begin{bmatrix} P_3 + P_3^T & (*) & 0 & (*) \\ A_iP_1 - B_iM_{jk} - E_kP_3 + P_4^T & -E_kP_4 - P_4^TE_k^T & (*) & (*) \\ 0 & \varepsilon^2 A_i^T & -2\varepsilon^2 I & 0 \\ \varepsilon^2 I & -\varepsilon^2 E_k^T & 0 & -2\varepsilon^2 I \end{bmatrix} \quad (33)$$

or defining: $\Gamma_i^k = \begin{bmatrix} 0 & I \\ A_i & -E_k \end{bmatrix}$: $\Upsilon_{ij}^{k(2)} = \begin{bmatrix} \Upsilon_{ij}^{k(1)} & (*) \\ \varepsilon^2\left(\Gamma_i^k\right)^T & -2\varepsilon^2 I_{2n} \end{bmatrix}$ (34)

Then:

$$\Upsilon_{ii}^{k(2)} + Q_{ii}^{k(2)} = \begin{bmatrix} \Upsilon_{ii}^{k(1)} + Q_{ii}^{k(1)} & (*) \\ \varepsilon^2\left(\Gamma_i^k\right)^T & -\varepsilon^2 I_{2n} \end{bmatrix} \quad (35)$$

$$\Upsilon_{ij}^{k(2)}+\Upsilon_{ji}^{k(2)}+Q_{ij}^{k(2)}+Q_{ji}^{k(2)}=\begin{bmatrix}\Upsilon_{ij}^{k(1)}+\Upsilon_{ji}^{k(1)}+Q_{ij}^{k(1)}+Q_{ji}^{k(1)} & (*)\\ \varepsilon^2\left(\Gamma_i^k+\Gamma_j^k\right)^T & -4\varepsilon^2 I_{2n}\end{bmatrix} \tag{36}$$

Applying Schur's complement leads to:

$$(35)\Leftrightarrow\ \Upsilon_{ii}^{k(1)}+Q_{ii}^{k(1)}+\varepsilon^2\Gamma_i^k\left(\Gamma_i^k\right)^T<0 \tag{37}$$

$$(36)\Leftrightarrow\ \Upsilon_{ij}^{k(1)}+\Upsilon_{ji}^{k(1)}+Q_{ij}^{k(1)}+Q_{ji}^{k(1)}+\frac{\varepsilon^2}{4}\left(\Gamma_i^k+\Gamma_j^k\right)\left(\Gamma_i^k+\Gamma_j^k\right)^T<0 \tag{38}$$

As (8), (9) are verified for the theorem 2, then it always exists an enough small ε^2 such that (37) and (38) are satisfied.

To show the interest of this new result we will study a second academic example.

Example 2:

This example is constructed in a way that theorem 2 conditions fail to obtain a solution. Consider the following nonlinear model in a descriptor form with $x(t)=\begin{bmatrix}x_1(t)\\x_2(t)\end{bmatrix}$ the state vector:

$$E\left(x(t)\right)\cdot\dot{x}(t)=A\left(x_1(t)\right)\cdot\dot{x}(t)+B\left(x_1(t)\right)\cdot u(t) \tag{39}$$

The different matrices are given by:

$$A\left(x_1(t)\right)=\begin{bmatrix}-28.7 & 45.2\\ -14.7-\dfrac{\sin\left(x_1(t)\right)}{x_1(t)}\cdot 47.4 & -19.9\end{bmatrix},\ E\left(x(t)\right)=\begin{bmatrix}48.9-\cos\left(x_1(t)-x_2(t)\right)\cdot 41.8 & 33.5\\ -0.1 & -20.7\end{bmatrix}$$

and $B\left(x_1(t)\right)=\begin{bmatrix}-40\cdot\dfrac{\sin\left(x_1(t)\right)}{x_1(t)}\\ 5\end{bmatrix}$.

All the nonlinearities being bounded, following the same path as presented for the example 1, we obtain the TS model in a descriptor form:

$$\begin{cases}E_v\dot{x}(t)=A_h x(t)+B_h u(t)\\ y(t)=C_h x(t)\end{cases} \tag{40}$$

With: $A_1=\begin{bmatrix}-28.7 & 45.2\\ -4.272 & -19.9\end{bmatrix}$, $A_2=\begin{bmatrix}-28.7 & 45.2\\ -62.1 & -19.9\end{bmatrix}$, $B_1=\begin{bmatrix}8.8\\5\end{bmatrix}$, $B_2=\begin{bmatrix}-40\\5\end{bmatrix}$, $E_1=\begin{bmatrix}91.8 & 33.5\\ -0.1 & -20.7\end{bmatrix}$ and $E_2=\begin{bmatrix}8.2 & 33.5\\ -0.1 & -20.7\end{bmatrix}$. Using theorem 3 conditions allows obtaining the following solution.

Matrices : $P_1 = \begin{bmatrix} 4.117 & -0.249 \\ -0.249 & 1.058 \end{bmatrix}$, $P_3 = \begin{bmatrix} -5.196 & -0.729 \\ 1.254 & -59.511 \end{bmatrix}$ and $P_4 = \begin{bmatrix} 217.55 & 18.175 \\ 1966.3 & -1206.9 \end{bmatrix}$, and for the gains:

$$\begin{aligned} F_{11} &= \begin{bmatrix} -1.445 & -7.69 \end{bmatrix}, \quad F_{12} = \begin{bmatrix} -0.913 & -2.708 \end{bmatrix} \\ F_{21} &= \begin{bmatrix} -2.49 & -20.65 \end{bmatrix}, \quad F_{22} = \begin{bmatrix} -0.99 & -4.85 \end{bmatrix} \end{aligned} \tag{41}$$

The non linear model (39) and the obtained control law (17) with the gains (41) have been implemented using the MATLAB/SIMULINK software. An example of simulation is presented figure 1. Considering the initial condition vector $x(0) = \begin{bmatrix} -5 & 5 \end{bmatrix}^T$, the convergence of the state vector and the evolution of the control signal are presented figure 1.

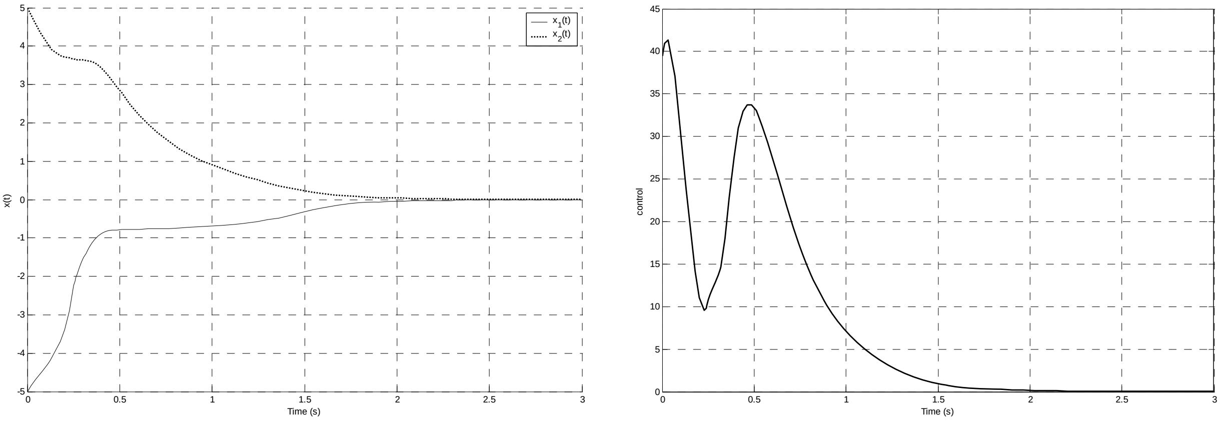


Figure 1. Stabilization of the non linear model (39) using control law (17) and initial conditions $x(0) = \begin{bmatrix} -5 & 5 \end{bmatrix}^T$.

5. Application to a Double-Inverted Pendulum

We consider the well-known double-inverted pendulum application. It is composed with a cart with two poles. Both of the poles are free in rotation around their axis as shown figure 2. The goal is to keep the angles $\theta(t)$ and $\beta(t)$ - respectively the angle between the first pole and the vertical and the second one and the vertical – around 0 and to ensure the tracking of the cart position $p(t)$. The different variables useful and the values chosen for the model description are resumed table 1.

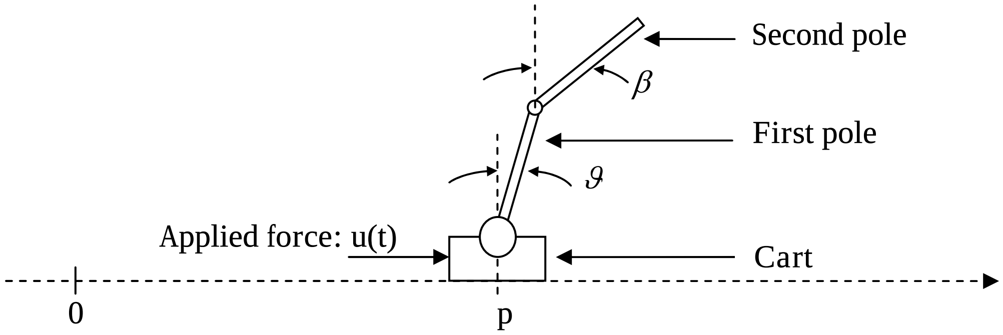


Figure 2. Representation of a double-inverted pendulum on a cart

Notation	value	Description
M	$30\ \mathrm{Kg}$	Mass of the cart
J_1	$5\cdot10^{-3}\ \mathrm{Kg}\cdot\mathrm{m}^2$	Inertia of the first pole
m_1	$0.2\ \mathrm{Kg}$	Mass of the first pole
l_1	$10\ \mathrm{cm}$	Half-length of the first pole
k_1	$0.1\ \mathrm{N}\cdot\mathrm{s}\cdot\mathrm{m}^{-1}$	First joint friction (viscous)
J_2	$8\cdot10^{-3}\ \mathrm{Kg}\cdot\mathrm{m}^2$	Inertia of the second pole
m_2	$0.3\ \mathrm{Kg}$	Mass of the second pole
l_2	$15\ \mathrm{cm}$	Half-length of the second pole
k_2	$0.1\ \mathrm{N}\cdot\mathrm{s}\cdot\mathrm{m}^{-1}$	Second joint friction (viscous)
g	$9.81\ \mathrm{m}\cdot\mathrm{s}^{-2}$	Gravity
f	$1\ \mathrm{m}\cdot\mathrm{s}^{-1}$	Friction
$\mathrm{u}(\mathrm{t})$		Force to apply on the cart
$\theta(\mathrm{t})$		Angle for the first pole
$\beta(\mathrm{t})$		Angle for the second pole
$\mathrm{p}(\mathrm{t})$		Position of the cart

Table 1. Variables useful for the double-inverted pendulum

According to the Euler Lagrange equations the following model can be obtained (Morère 2001):

$$\begin{aligned}(\mathrm{M}+\mathrm{m}_1+\mathrm{m}_2)\cdot\ddot{\mathrm{p}}+(\mathrm{m}_1+2\cdot\mathrm{m}_2)\cdot\mathrm{l}_1\cdot\cos(\theta)\cdot\ddot{\theta}+\mathrm{m}_2\cdot\mathrm{l}_2\cdot\cos(\beta)\cdot\ddot{\beta}=\\ -\ \mathrm{f}\cdot\dot{\mathrm{p}}+(\mathrm{m}_1+2\cdot\mathrm{m}_2)\cdot\mathrm{l}_1\cdot\dot{\theta}^2\cdot\sin(\theta)+\mathrm{m}_2\cdot\mathrm{l}_2\cdot\dot{\beta}^2\cdot\sin(\beta)+\mathrm{u}\end{aligned}\tag{42}$$

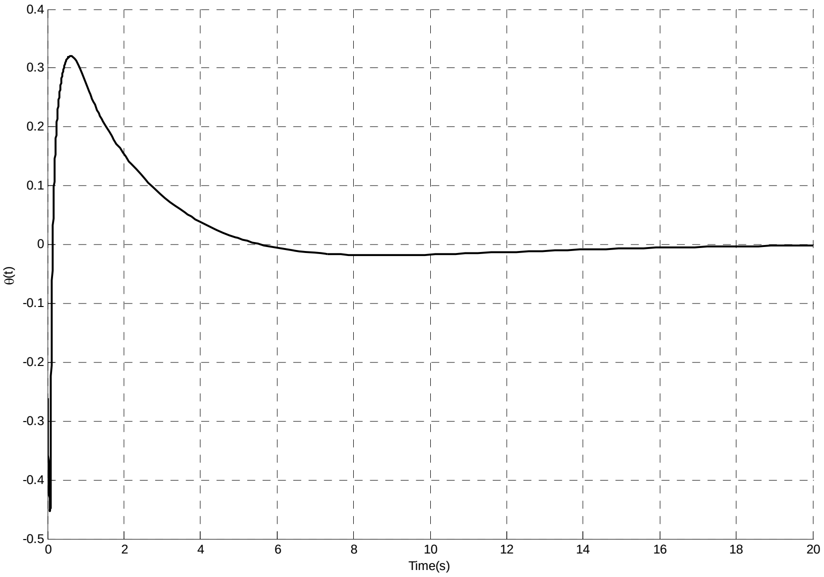
$$\begin{aligned}(\mathrm{m}_1+2\cdot\mathrm{m}_2)\cdot\mathrm{l}_1\cdot\cos(\theta)\cdot\ddot{\mathrm{p}}+(\mathrm{m}_1\cdot\mathrm{l}_1^2+4\cdot\mathrm{m}_2\cdot\mathrm{l}_1^2+\mathrm{J}_1)\cdot\ddot{\theta}+2\cdot\mathrm{m}_2\cdot\mathrm{l}_1\cdot\mathrm{l}_2\cdot\cos(\theta-\beta)\cdot\ddot{\beta}=\\ -2\cdot\mathrm{m}_2\cdot\mathrm{l}_1\cdot\mathrm{l}_2\cdot\dot{\beta}^2\sin(\theta-\beta)+(\mathrm{m}_1+2\cdot\mathrm{m}_2)\cdot\mathrm{g}\cdot\mathrm{l}_1\cdot\sin(\theta)-\mathrm{k}_1\cdot\dot{\theta}\end{aligned}\tag{43}$$

$$\begin{aligned}\mathrm{m}_2\cdot\mathrm{l}_2\cdot\cos(\beta)\cdot\ddot{\mathrm{p}}+2\cdot\mathrm{m}_2\cdot\mathrm{l}_1\cdot\mathrm{l}_2\cdot\cos(\theta-\beta)\cdot\ddot{\theta}+(\mathrm{m}_2\cdot\mathrm{l}_2^2+\mathrm{J}_2)\cdot\ddot{\beta}=\\ +2\cdot\mathrm{m}_2\cdot\mathrm{l}_1\cdot\mathrm{l}_2\cdot\dot{\theta}^2\cdot\sin(\theta-\beta)+\mathrm{m}_2\cdot\mathrm{g}\cdot\mathrm{l}_2\cdot\sin(\beta)-\mathrm{k}_2\cdot\dot{\beta}\end{aligned}\tag{44}$$

With the state vector $\mathrm{x}(\mathrm{t})=\begin{bmatrix}\dot{\mathrm{p}} & \dot{\theta} & \dot{\beta} & \ddot{\mathrm{p}} & \ddot{\theta} & \ddot{\beta}\end{bmatrix}^{\mathrm{T}}$ we have the following descriptor form of the double-inverted pendulum:

$$\begin{bmatrix}1 & 0 & 0 & 0 & 0 & 0\\ 0 & 1 & 0 & 0 & 0 & 0\\ 0 & 0 & 1 & 0 & 0 & 0\\ 0 & 0 & 0 & \mathrm{M}+\mathrm{m}_1+\mathrm{m}_2 & (\mathrm{m}_1+2\mathrm{m}_2)\mathrm{l}_1\cos(\theta) & \mathrm{m}_2\mathrm{l}_2\cos(\beta)\\ 0 & 0 & 0 & (\mathrm{m}_1+2\cdot\mathrm{m}_2)\mathrm{l}_1\cos(\theta) & \mathrm{m}_1\mathrm{l}_1^2+4\mathrm{m}_2\mathrm{l}_1^2+\mathrm{J}_1 & 2\mathrm{m}_2\mathrm{l}_1\mathrm{l}_2\cos(\theta-\beta)\\ 0 & 0 & 0 & \mathrm{m}_2\mathrm{l}_2\cos(\beta) & 2\mathrm{m}_2\mathrm{l}_1\mathrm{l}_2\cos(\theta-\beta) & \mathrm{m}_2\mathrm{l}_2^2+\mathrm{J}_2\end{bmatrix}\begin{bmatrix}\dot{\mathrm{p}}\\ \dot{\theta}\\ \dot{\beta}\\ \ddot{\mathrm{p}}\\ \ddot{\theta}\\ \ddot{\beta}\end{bmatrix}$$

$$=\begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -\mathrm{f} & 0 & 0 \\ 0 & \left(\mathrm{m}_1+2\mathrm{m}_2\right)\mathrm{g}\mathrm{l}_1\dfrac{\sin(\theta)}{\theta} & 0 & 0 & -\mathrm{k}_1 & 0 \\ 0 & 0 & \mathrm{m}_2\mathrm{g}\mathrm{l}_2\dfrac{\sin(\beta)}{\beta} & 0 & 0 & -\mathrm{k}_2 \end{bmatrix}\begin{bmatrix} \mathrm{x} \\ \theta \\ \beta \\ \dot{\mathrm{x}} \\ \dot{\theta} \\ \dot{\beta} \end{bmatrix}+\begin{bmatrix} 0 \\ 0 \\ 0 \\ \mathrm{H}_4 \\ \mathrm{H}_5 \\ \mathrm{H}_6 \end{bmatrix}+\begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}\mathrm{u} \tag{45}$$



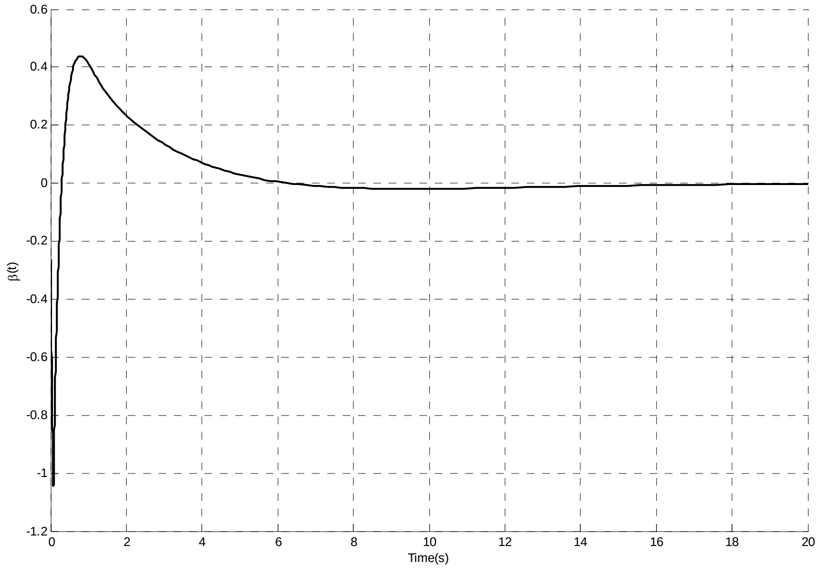


Figure 3. Simulation of the double-inverted pendulum. Upper figure: first pole angle $\Theta(\mathrm{t})$, lower figure: second pole angle $\beta(\mathrm{t})$

With: $H_4 = (m_1 + 2m_2) l_1 \dot{\theta}^2 \sin(\theta) + m_2 l_2 \dot{\beta}^2 \sin(\beta)$, $H_5 = -2m_2 l_1 l_2 \dot{\beta}^2 \sin(\theta - \beta)$ and $H_6 = 2m_2 l_1 l_2 \dot{\theta}^2 \sin(\theta - \beta)$

Or in a compact writing:

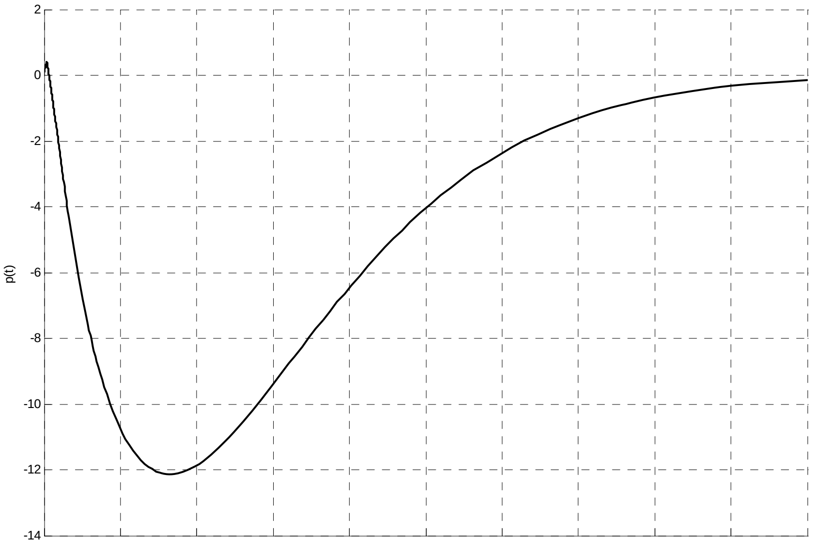
$$\begin{aligned} E_v \dot{x}(t) &= A_h x(t) + B_h u(t) + H \\ y(t) &= C_h x(t) \end{aligned} \tag{46}$$

Remark 5: In this simplified version the term H is neglected. In a more general framework, this is due to the fact that for output stabilization, i.e. with an observer, there exists a separation principle in the case where the premise vector $z(t)$ is measurable. As $\theta(t)$ and $\beta(t)$ are measurable, neglecting H allows using this principle. Let us also say that several ways to use a complete TS descriptor model are possible. One is to consider the H part as bounded uncertainties and then use robust conditions of stabilization; nevertheless this was not the purpose of this chapter.

For E_v it is necessary to take into account three nonlinearities: $\cos(\theta)$, $\cos(\beta)$ and $\cos(\theta - \beta)$. That means $e = 2^3 = 8$ functions v_k. For A_h 2 nonlinearities $\frac{\sin(\theta)}{\theta}$ and $\frac{\sin(\beta)}{\beta}$, that means $r = 2^2 = 4$ functions h_i. Conditions of theorem 2 were then performed to obtain the control law.

An example of simulation is presented in the next figures with the initial condition: $\theta(0) = -20°$ and $\beta(0) = -15°$.

Figure 3 shows the evolution of the angles $\theta(t)$ and $\beta(t)$, figure 4 the evolution of the cart position $p(t)$ and the control law. The three first figures are simulated on $20s$, the last figure, the control law, is zoomed on the first $0.25s$.



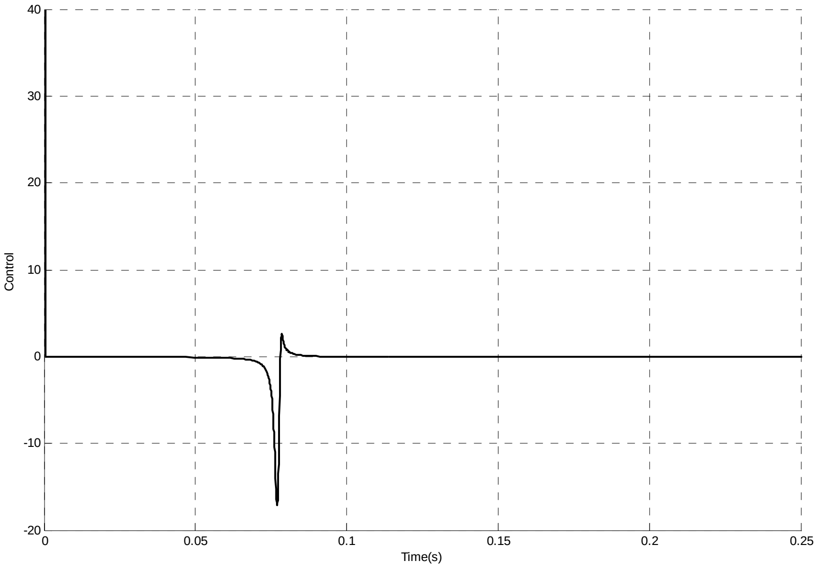


Figure 4. Simulation of the double-inverted pendulum. Upper figure: cart position *p(t),* lower figure: control law evolution (zoomed on the first 0.25s)

6. Conclusion

The chapter focused on TS fuzzy models in descriptor form stabilization. As for classical TS models, they can be obtained in a systematic way using the sector nonlinearity approach. Their main interest is to remain close to the nonlinear model. Thus, in some cases the number of models involved can be highly reduced in comparison to a classical TS model representing the same nonlinear model. Hence, the results obtained with the conditions of stabilization can allow reducing in a large way the conservatism of preceding classical results. Nevertheless, if we want to outperform basic conditions of stabilization for TS models in the descriptor form, a way is to use specific matrix transformation. The application of matrix transformation allowed outperforming the results. At last we presented the application to the well-known double inverted pendulum in simulation.

To go further, robustness can be easily introduced in the different conditions. It can be done using classical bounded uncertainties. The regulator problem could be also investigated.

7. References

Y. Blanco, W. Perruquetti, & P. Borne (2001). Non quadratic stability of nonlinear systems in the Takagi–Sugeno form. Proceedings of European Control Conference, Porto, Portugal.

S. Boyd, L. El Ghaoui, E. Feron & V. Balakrishnan (1994), Linear Matrix Inequalities in system and control theory. SIAM, Philadelphia, PA

B.S. Chen, C.S. Tseng, & H.J. Uang (2000). Mixed fuzzy output feedback control design for nonlinear dynamic systems: an LMI approach. IEEE Trans. Fuzzy Systems, 8(3):249–265.

G. Feng (2003). Controller synthesis of fuzzy dynamic systems based on piecewise Lyapunov functions. IEEE Trans. Fuzzy Systems, 11(5):605–612.

T.M. Guerra, M. Ksontini & F. Delmotte (2003). Some new relaxed conditions of quadratic stabilization for continuous Takagi-Sugeno fuzzy models. Proceedings of IEEE CESA'03 Lille, France

T.M. Guerra, K. Guelton & S. Delprat (2004). A class of nonlinear observers in descriptor form: LMI based design with applications in biomechanics, Proceedings of the Workshop IFAC/AFNC'04, Oulu, Finland

T.M. Guerra & L. Vermeiren (2004). LMI-based relaxed non quadratic stabilization conditions for non-linear systems in the Takagi-Sugeno's form. Automatica, 40(5):823–829.

J. Joh, R. Langari, E.T. Jeung, & W.J. Chung (1997). A new design method for continuous Takagi–Sugeno fuzzy controller with pole placement constraints: an LMI approach. IEEE Trans. Fuzzy Systems, 5(3):72–79.

M. Johansson, A. Rantzer, & K.E. Arzen (1999). Piecewise quadratic stability of fuzzy systems. IEEE Trans. Fuzzy Systems, 7:713–722.

E. Kim & H. Lee (2000). New Approaches to Relaxed Quadratic Stability Condition of Fuzzy Control Systems, IEEE Transactions on Fuzzy Systems, 8(5) 523-533

X. Liu & Q. Zhang (2003). New approaches to H_∞ controller designs based on fuzzy observers for T-S fuzzy systems via LMI, Automatica, 39(9) 1571-1582

Y. Morère (2001). Control laws synthesis for Takagi-Sugeno fuzzy models. PhD Dissertation, LAMIH, Univ. de Valenciennes et du Hainaut-Cambrésis (in French)

D. Peaucelle, D. Arzelier, O. Bachelier & J. Bernussou (2000). A new robust D-stability condition for real convex polytopic uncertainty. Systems and Control letters, 40 (1), 21-30

T. Takagi & M. Sugeno (1985). Fuzzy identification of systems and Its applications to modeling and control, IEEE Trans. Systems Man and Cybernetics 15(1) 116-132.

K. Tanaka, T. Hori, T. Taniguchi & H.O. Wang (2001). Stabilization of nonlinear systems based on fuzzy Lyapunov function, Workshop IFAC/AFNC, Valencia, Spain

K. Tanaka, T. Ikeda, & H.O. Wang (1996). Robust stabilization of a class of uncertain nonlinear systems via fuzzy control: Quadratic stability, H1 control theory and linear matrix inequalities. IEEE Transactions on Fuzzy Systems, 4(1):1–13.

K. Tanaka, T. Ikeda, & H.O. Wang (1998). Fuzzy regulators and fuzzy observers: relaxed stability conditions and LMI-based designs. IEEE Trans. Fuzzy Systems, 6(2):1– 6.

K. Tanaka & H.O. Wang (2001). Fuzzy control systems design and analysis. A linear matrix inequality approach. John Wiley & Sons, New York.

T. Taniguchi, K. Tanaka & H.O. Wang (2000). Fuzzy descriptor systems and nonlinear model following control. IEEE Transactions on Fuzzy Systems 8(4) 442-452

T. Taniguchi, K. Tanaka, H. Ohtake & H.O. Wang (2001). Model construction, rule reduction and robust compensation for generalized form of Takagi-Sugeno fuzzy systems. IEEE Transactions on Fuzzy Systems 9(4), 525-537

M.C.M. Teixeira, E. Assunçao & R.G. Avellar (2003). On relaxed LMI-based design for fuzzy regulators and fuzzy observers. IEEE Transactions on Fuzzy Systems 11(5) 613-623

S. Tong, T. Wang, & H.X. Li (2002). Fuzzy robust tracking control for uncertain nonlinear systems. International Journal of Approximate Reasoning, 30:73–90.

H.D. Tuan, P. Apkarian, T. Narikiyo, & Y. Yamamoto (2001). Parameterized linear matrix inequality techniques in fuzzy control system design. IEEE Trans. Fuzzy Systems, 9(2):324–332.

H.O. Wang, K. Tanaka & M. Griffin (1996). An approach to fuzzy control of nonlinear systems: stability and design issues. IEEE Trans. on Fuzzy Systems 4(1) 14-23

J. Zhao (1995). Fuzzy logic in modeling and control. PhD dissertation, CESAME, Louvain la Neuve, Belgium, 1995.

Adaptive Control of Nonlinear Dynamics Systems Based on RBF Network

Ho Dac Loc, Nguyen Thi Phuong Ha & Luong Van Lang

1. Introduction

As it was mentioned in (Narenda et al., 1990; Ching-Teng Lin et al., 1996), many adaptive control techniques, such as self-tuning control, self-tuning PID, etc. have been developed and successfully implemented in many applications. More recently, many adaptive control systems which use artificial intelligent techniques are developed to deal with increasing complexity in control systems, such as non-linearity, unexpected load disturbances, variable time delay, etc. In many direct adaptive control approaches, the actual system error is usually use to tune the controller parameters. Several adaptive control strategies also make use of predictive models in formulating their adaptation laws. A large number of these intelligent adaptive control systems use neural network and fuzzy logic, and also combination of other new paradigms (Kosko B., 1994; Lewis F.L., 1995).

In this chapter, we will develop an adaptive controller based on RBF network for unknown nonlinear dynamic system. Design of adaptive controller is based on the theory optimal control. Some applications of the proposed methodology are introduced. The simulation results made by matlab showed that the synthesized adaptive control system have good performance. 2. Statement of the problem

Let us consider the nth-order nonlinear dynamic system in the form:

$$\begin{cases} \dot{x}_1 = f_1(\underline{x}) \\ \dot{x}_2 = f_2(\underline{x}) \\ \cdots\cdots\cdots\cdots\cdots\cdots \\ \dot{x}_n = f_n(\underline{x}) + u \\ y = x_1, \end{cases} \tag{1}$$

where $\underline{x} = (x_1, x_2, ..., x_n)^T$ is the state vector; $f_i, i = \overline{1,n}$ are nonlinear continuous functions; y is the output signal; u is the control signal from controller.

The control objective is a determination of a feedback control u such that the following conditions are met:

the close-loop system must be globally stable.

b) $$\lim_{t\to\infty}|y_m - y| = \lim_{t\to\infty}|y_m - x_1| = \lim_{t\to\infty}|e| \tag{2}$$

where y_m is the output reference signal.

If the functions $f_i, i = \overline{1,n}$ are known, then the following control law can ensure the control objective (Kolesnikov A.A, 1994):

$$\hat{u} = -\left[\frac{\partial\Psi(\underline{x})}{\partial x_n}\right]^{-1}\left[\frac{1}{T}\varphi(\Psi) + \sum_{i=1}^{n}\frac{\partial\Psi(\underline{x})}{\partial x_i} f_i(\underline{x})\right] \tag{3}$$

where T is a positive constant; $\Psi(\underline{x})$ is a differentiable function of state variables and $\Psi(0)=0$; $\varphi(\Psi)$ is a continuous differentiable function responding the following conditions : a) $\varphi(0)=0$; b) $\varphi(\Psi)*\Psi > 0 \quad \text{for} \quad \forall\Psi \neq 0$.

The control law (3) cannot be implemented since: a) $f_i, i = \overline{1,n}$ are unknown; b) part of state vector is not measured variables. There are many practical situations where both a) and b) are true. The adaptive controller developed in this chapter is suitable for these situations.

3. Design of the adaptive controller

The basic architecture of the adaptive controller is a standard radial basic function neural network, plus an adaptive block which adjusts the parameters of the controller.

To begin, let $\Psi(\underline{e}) = e_1 + \alpha e_2$, where $\underline{e} = \left(e, \dot{e}\right)^T = (e_1, e_2)^T$; α is positive constant; $\varphi(\Psi) = \Psi$. Using (1), we have :

$$\begin{aligned} \frac{\partial\Psi}{\partial x_1} &= -1 - \alpha\frac{\partial f_1(\underline{x})}{\partial x_1} \\ \frac{\partial\Psi}{\partial x_2} &= -\alpha\frac{\partial f_1(\underline{x})}{\partial x_2} \\ &\ldots\ldots\ldots\ldots\ldots\ldots \\ \frac{\partial\Psi}{\partial x_n} &= -\alpha\frac{\partial f_1(\underline{x})}{\partial x_n} \end{aligned} \tag{4}$$

Substituting (4) into (3), we have:

$$\hat{u} = \left(\alpha\frac{\partial f_1}{\partial x_n}\right)^{-1}\left[\Psi - \left(1 + \alpha\frac{\partial f_1(\underline{x})}{\partial x_1}\right)f_1(\underline{x}) \quad - \alpha\frac{\partial f_1(\underline{x})}{\partial x_2} f_2(\underline{x}) - \alpha\frac{\partial f_1(\underline{x})}{\partial x_n} f_n(\underline{x})\right] \tag{5}$$

If the functions $f_i, \; i = \overline{1,n}$ are known, then using control law of (5) applied to (1), we can write the following differentiable equation for x_1:

$$\ddot{x}_1 = \sum_{i=1}^{n}\frac{\partial f_1(\underline{x})}{\partial x_i}\dot{x}_i \quad = \frac{1}{\alpha}\Psi - \frac{1}{\alpha}f_1(\underline{x}) \quad = \frac{1}{\alpha}\left(y_m - x_1 - \alpha\dot{x}_1\right) - \frac{1}{\alpha}\dot{x}_1 \tag{6}$$

or

$$\ddot{x}_1 + \left(1 + \frac{1}{\alpha}\right)\dot{x}_1 + \frac{1}{\alpha}(x_1 - y_m) = 0 \tag{7}$$

Using $z = x_1 - y_m$, we have:

$$\ddot{z} + a_1 \dot{z} + a_0 z = 0 \tag{8}$$

where $a_1 = 1 + \frac{1}{\alpha}; \quad a_0 = \frac{1}{\alpha}$. Solving the differential equation (8), we have:

$$z(t) = C_1 \exp\left[-\frac{1}{2}\left(a_1 - \sqrt{a_1^2 - 4a_0}\right)t\right] + C_2 \exp\left[-\frac{1}{2}(a_1 + \sqrt{a_1^2 - 4a_0} \cdot t\right] \tag{9}$$

where C1 and C2 are constants, which implies that $\lim\limits_{t\to\infty} |e| = 0$ - main control objective.
Our purpose is to design a RBF neural network to approximate the optimal control of (5). The RBF network with two inputs, Gaussian basic functions and normalized output can be described as:

$$u = \frac{\sum_{j=1}^{L} \bar{\lambda}^j \left(\prod_{i=1}^{2} \mu_{A_i^j}(e_i)\right)}{\sum_{i=1}^{L}\left(\prod_{i=1}^{2} \mu_{A_i^j}(e_i)\right)} = \underline{\lambda}^T \frac{\prod_{i=1}^{2} \mu_{A_i^j}(e_i)}{\sum_{j=1}^{L}\left(\prod_{i=1}^{2} \mu_{A_i^j}(e_i)\right)} = \underline{\lambda}^T \underline{\zeta}(\underline{e}) \tag{10}$$

where $\mu_{A_i^j}(e_i)$ is the activation function of neuron A_i^j; L is the number of neurons in hidden layer; $\bar{\lambda}^j$ are unknown parameters of controller, which have been adjusted.
The equation (1) can be written as:

$$\begin{cases} \dot{x}_1 = f_1(\underline{x}) \\ \dot{x}_2 = f_2(\underline{x}) \\ \dots\dots\dots\dots\dots\dots\dots\dots\dots \\ \dot{x}_n = f_n(\underline{x}) + u + \hat{u} - \hat{u} \end{cases} \tag{11}$$

Substituting (5) and approximated control signal by RBF-network output $u = \underline{\lambda}^{*T} \underline{\zeta}(\underline{e})$ into (11), we can write last equation of (11) in the form:

$$\dot{x}_n = \left(\alpha \frac{\partial f_1(\underline{x})}{\partial x_n}\right)^{-1} \left[\Psi - \left(1 + \alpha \frac{\partial f_1(\underline{x})}{\partial x_1}\right) f_1(\underline{x}) - \dots - \alpha \frac{\partial f_1(\underline{x})}{\partial x_{n-1}} f_{n-1}(\underline{x})\right] + \left(\underline{\lambda}^T - \underline{\lambda}^{*T}\right)\underline{\zeta}(\underline{e}) \tag{12}$$

Next, we develop an adaptive law to adjust the parameters of vector $\underline{\lambda}$, which provides that the close-loop system is generally stable. Define the Lyapunov function candidate:

$$V = \frac{1}{2}\Psi(\underline{e})^2 + \frac{1}{2\gamma}\underline{\theta}^T \underline{\theta} \tag{13}$$

where $\underline{\theta} = \underline{\lambda} - \underline{\lambda}^*$; γ is positive constant. Using (11) and (12) we have a derivative of Lyapunov function as

$$\dot{V} = \Psi\dot{\Psi} + \frac{1}{\gamma}\underline{\theta}^T \underline{\dot{\theta}} = -\frac{1}{T}\Psi^2 + \underline{\theta}^T \left[\frac{1}{\gamma}\underline{\dot{\lambda}} - \alpha\Psi \frac{\partial f_1}{\partial x_n}\underline{\zeta}(\underline{e})\right] \tag{14}$$

If we choose the adaptive law :

$$\dot{\underline{\lambda}} = \gamma\Psi\frac{\partial f_1}{\partial x_n}\underline{\zeta}(\underline{e})$$

then (14) becomes:

$$\dot{V} = -\frac{1}{T}\Psi^2 \quad \le \quad 0 \tag{15}$$

which guarantees that the close-loop system is generally stable. The overall scheme of the developed adaptive control system is show in Fig.1 where C is an unit which inverts the error signal e into the error vector $\underline{e}$. Now, we make some few remarks.

Remark 1. The developed, in this paper, adaptive fuzzy control system using the error vector as inputs signal for the controller is suitable in situations, when the state vector is not measured variables.

Remark 2. The simple adaptive law makes easy to implement the adaptive NN-controller. The quality of the control system depends on form and parameters of function ψ.

Remark 3. In using the adaptive law (18), the choice of the constant γ is very important. At present, there has been no theoretical guidance about the choice of γ; trial-and-error seems to be the only practical option.

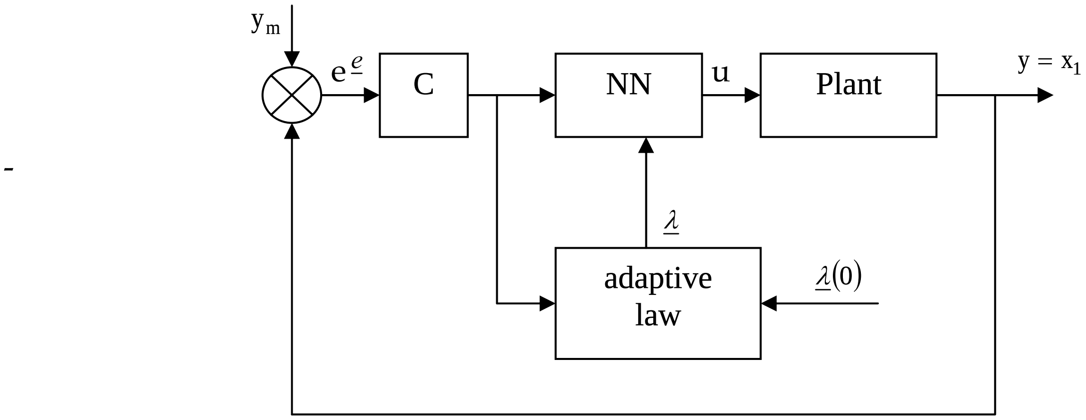


Figure 1. The overall scheme of adaptive control system

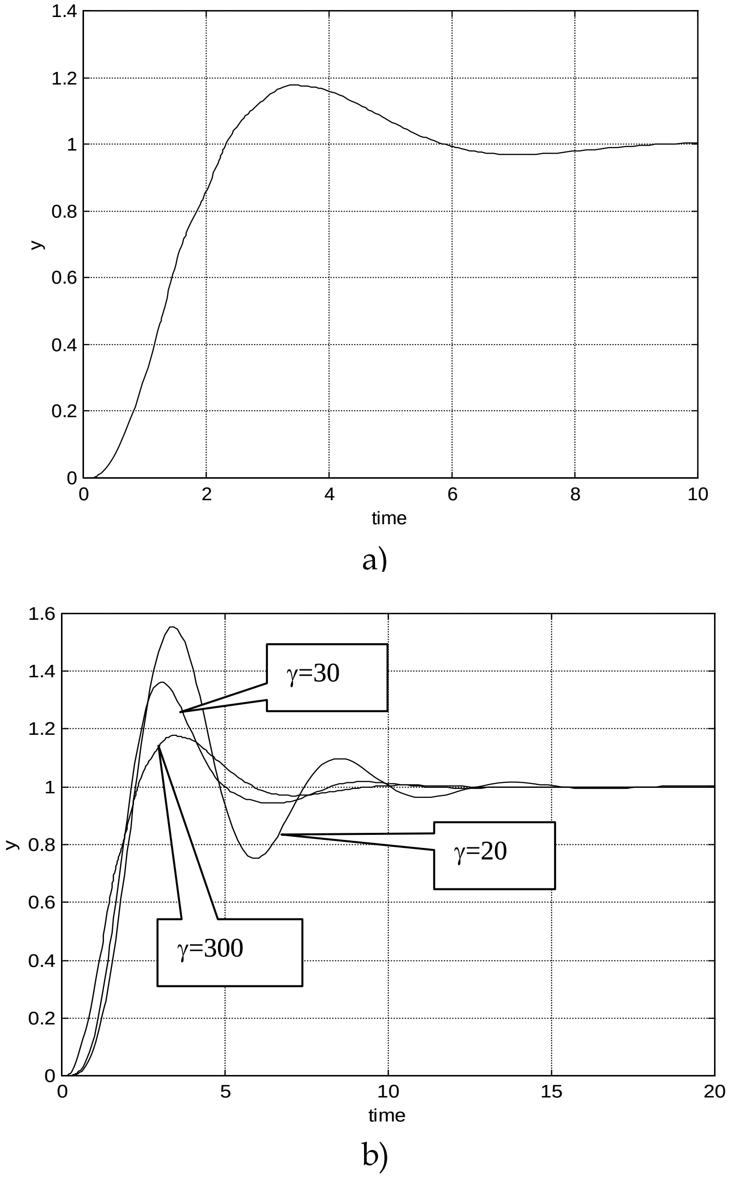
4. Applications

4.1 Adaptive Control of Nonlinear System Sat Function

In this section, we apply the adaptive RBF-network controller developed by proposed method to control the following nonlinear dynamic system:

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = -x_2 + \text{sat}(x_3) \\ \dot{x}_3 = -x_3 + u \\ y = x_1 \end{cases} \tag{16}$$

The step respond of close-loop control system is shown in Fig.2 in which we can see the controller could regulate the plant and the close-loop system is stable. The step respond of the control system in situation, where γ has different values, is shown in Fig.2.b. From this we can see that the quality of the transient process depends on γ.



a)

b)

Figure 2. The step respond of the control system

4.2 Adaptive Control of DC Motor Containing Nonlinear Friction

The high-quality servos are largely described by nonlinear models. Their performance is often limited by nonlinear phenomena such as friction and backlash. Therefore, we consider a problem of this type, namely, a servo with nonlinear friction, which causes difficulties and gives rise to poor performance in precision servos.

Friction compensation has been considered before. In order to address better the demands of high fidelity control, adaptive friction compensation algorithms have recently appeared in the literature. The use of a recursive least-squares algorithm to estimate the parameters in a nonlinear friction model was described by Canudas et al. (Canudas et al., 1987). Friedland and Park (Friedland B. et al., 1987) presented another adaptive friction compensation scheme which was based upon a Lyapunov-like argument involving the position error. Many other studies on friction compensation are reported in a survey paper (Amstrong-Helouvry et al., 1994). However, these methods are based on the characteristics of the nonlinearity and knowledge of some of the parameters, in contrast to the adaptive methods proposed here.

The overall scheme of adaptive fuzzy logic control system, which is considered in this paper, is shown in Fig.3. A DC motor with a permanent magnet was used in our control system. Such a motor is used in robots and precision servos. The motor with nonlinear friction can be described by the following model:

$$\begin{aligned} \dot{x}_1 &= -\frac{1}{T_m}x_1 + \frac{k_m}{T_m}\left[x_2 - M_d - F(x_1)\right] \\ \dot{x}_2 &= -\frac{1}{T_e}x_2 + \frac{k_e}{T_e}u \quad , \end{aligned} \tag{17}$$

where $x_1 = \omega$ is the velocity of the motor shaft, M_d is the load disturbance, k_e and k_m are the gains, T_e and T_m are the time coefficients. The friction model is:

$$M_f = F(\omega) = \alpha_0 \operatorname{sgn}(\omega) + \alpha_1 \exp\left(-\alpha_2|\omega|\right)\operatorname{sgn}(\omega) \tag{18}$$

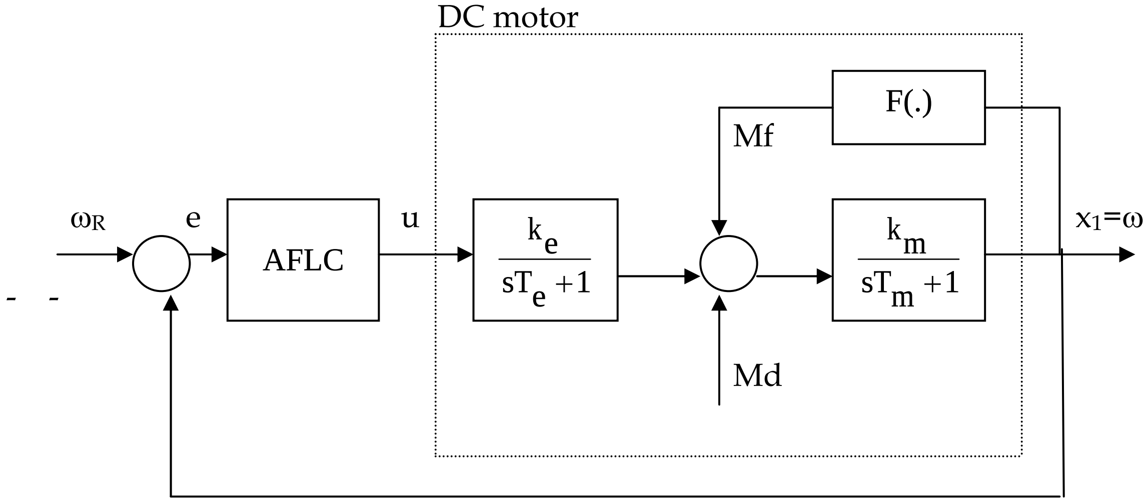
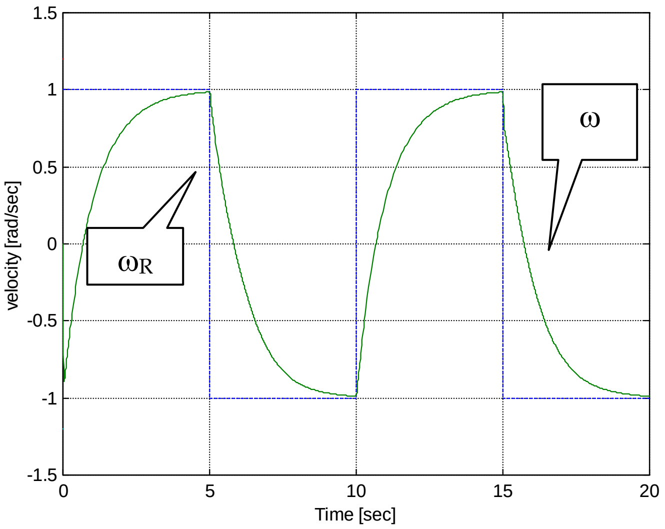


Figure 3. Adaptive control system of DC-motor



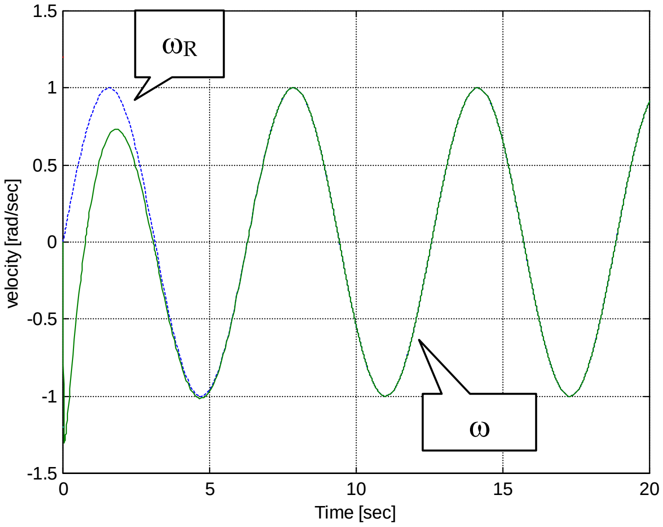


Figure 4. The references and real velocity of DC motor

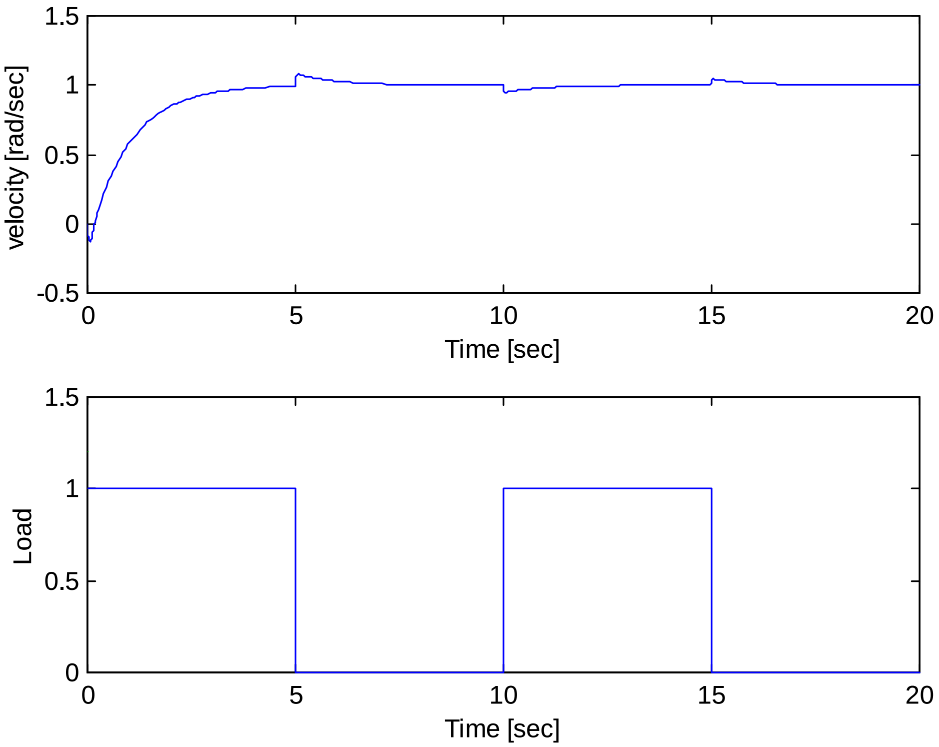


Figure 5. Velocity of DC motor with changing load

We apply the adaptive RBF-network controller developed in this chapter to control the DC motor with nonlinear friction. The computer simulation of control system is done by the language MATLAB.

The simulation result of the DC motor with adaptive RBF-network controller is shown in Fig.4, in which we can see that the developed controller could achieve the reference velocity of DC motor. We also investigate the motor velocity with the sinusoidal reference signal. In Fig.5 is showed the respond of velocity depending on load changing. From this we can see that the adaptive RBF-network controller achieves good trajectory. Figure 4. The velocity of DC motor and load.

4.3 Adaptive Control of Robot-Manipulator

Robot manipulators are complicated nonlinear dynamical systems with inherent unmodeled dynamics and unstructured uncertainties. These uncertainties make the controller design for robot manipulators a difficult task in the framework of classical adaptive and unadaptive control.

The traditional PID control with a simple structure and implementation (Koditchek D.E., 1984) has been the predominant method used for industrial manipulators controllers. CTM (Paul R.C., 1972) and ACM (Slotine J.E. et al., 1987) good performance, if manipulator dynamics are exactly known. However, they suffer from following difficulties: a) they require explicit a priory knowledge of individuals manipulators, which is very difficult to acquire in most practical applications; b) uncertainties existing in real manipulators seriously devalue the performance of both methods; c) the computational load of both methods is high.

In recent years, much attention has been paid to neural-network (NN) based controller. The nonlinear mapping and learning properties of NN a key factor for their use in the control field. These controllers take advantage of the capability of a NN for learning nonlinear functions and of the massive parallel computation, required in the implementation of advanced control algorithms.

This section deals with a neural-network based controller developed in this chapter for motion dynamic control of robot manipulator. The uncertainties on the robot dynamic parameters have motivated the design adaptive controller.

The general equation describing the dynamics of an n-degree of freedom rigid robot manipulator is given by:

$$M(q)\ddot{q}+C\left(q,\dot{q}\right)\dot{q}+G(q)+F\left(q,\dot{q}\right)=u \tag{19}$$

where $q,\dot{q}\in R^n$ are the vector of generalized coordinators and velocities; $M(q)\in R^{n\times n}$ is the positive inertia matrix; $C\left(q,\dot{q}\right)\in R^n$ are the coriolis and centrifugal torques; $G(q)\in R^n$ are the gravitational torques; $F\left(q,\dot{q}\right)$ is the unstructured uncertainty of the dynamics including friction and other disturbances.

We will consider robot-manipulator which dynamic model is described in state space (Anna Jadlovska, 2000) :

$$\begin{aligned}
\dot{x}_1 &= x_2\\
\dot{x}_2 &= \frac{m_b}{m_r}x_1x_4^2+\frac{K_1}{m_r}u_1\\
\dot{x}_3 &= x_4\\
\dot{x}_4 &= -\frac{2m_bx_1x_2x_4}{I_{23}+m_bx_1^2}+\frac{K_2}{I_{23}+m_bx_1^2}u_2
\end{aligned} \tag{20}$$

where $m_b = m_z + m_l$; $m_r = m_{rr} + m_b + m_2$; $m_g = 35$ kg is the mass of weight; $m_l = 52$ kg is the grasp head and part of the arm; $m_{rr} = 62,5$ kg is reduced mass of the gear; $m_2 = 78$ kg is the mass of servomotors of the arm; K1= 281.4 Nm, K2= 291 Nm are the constants of the operational values; $I_{23} = I_r + (m_2 + m_3 + m_4) \cdot r_0^2$ where $I_r = 82,5$ kgm² is reduced torque of inertial of the electric servomotor and gear-box; $m_3 = 90$ kg; $m_4 = 125$ kg; $r_0 = 250$ mm.

In this section, we apply the adaptive controller developed in this chapter to control the two-link robot-manipulator of (20). The computer simulation of control system is done by the language MATLAB. The response of close-loop control system is shown in Fig.6, where the given outputs are $x_1^r = 1$ and $x_3^r = 0.8$, $\gamma_1 = \gamma_2 = 500$, $\alpha_1 = \alpha_2 = 1$, in which we can see that the developed adaptive controller could regulate plant and the close-loop control system is stable.

In Fig.7 is shown the response of the control system, where γ_1, γ_2 have different values. From this we can see that the quality of the control system depends on γ_1, γ_2 ($\alpha_1 = \alpha_2 = 1$).

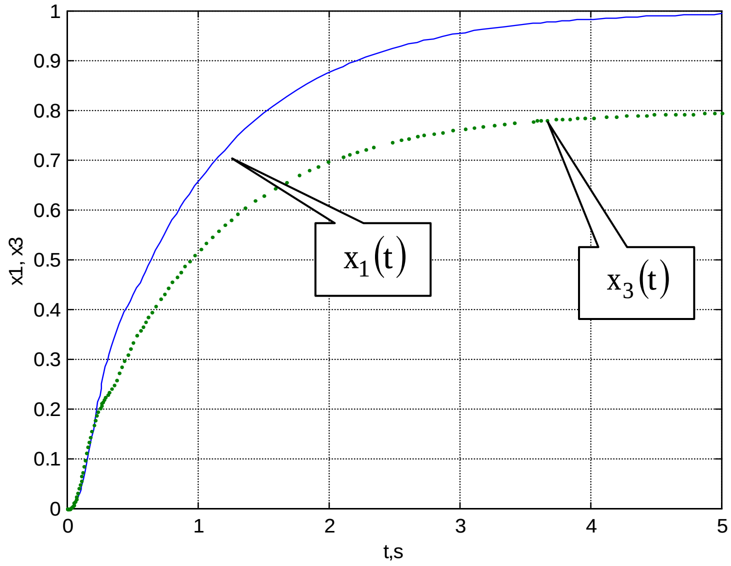


Figure 6. The response of the control system

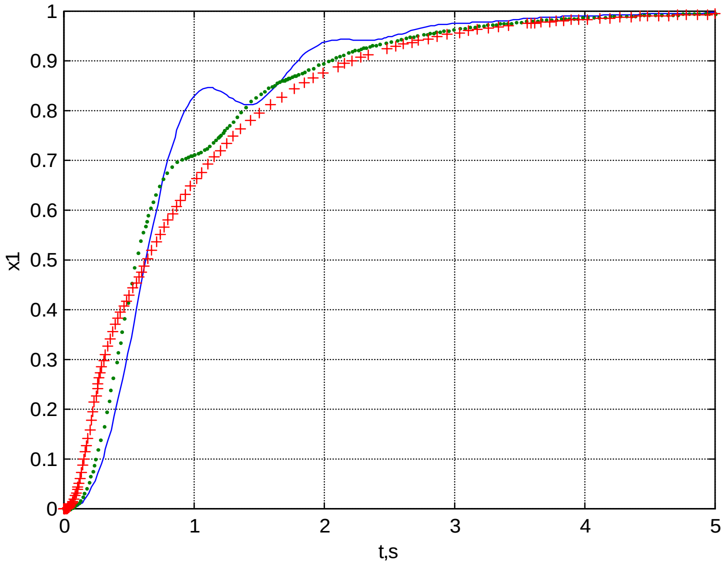


Figure 2. The state variable x1(t) γ1=γ2=50 ___, γ1=γ2=100..., γ1=γ2=500

5. Conclusion

In this work, we developed an adaptive fuzzy controller which : 1) does not require an accurate mathematical model of plant under control; 2) uses the error vector as controller's input, therefore it does not require all that components of state vector to be measurable; and 3) guarantees the global stability of the close-loop system. The simulation results show that the adaptive fuzzy controller could successful control the unknown nonlinear dynamic system.

6. References

K.S. Narenda, K. Parthasarathy (1990). Identification and control of dynamical systems using neural networks. IEEE Trans. Neural Networks, 1 (1), 4-27.

Ching-Teng Lin and C.S. George Lee (1996). A neuro-fuzzy synergism to intelligent systems. Prentice-Hall International, Inc.

Kosko, B (1994). Fuzzy systems as universal approximators, IEEE Transactions on computers, 43 (11),1329-1333.

Lewis, F.L., Zhu, S.Q. & Liu K. (1995). Function approximation by fuzzy systems. Proceedings of American control conference,.3760-3764.

Kolesnikov A.A. (1994). Synergism control theory. Energy-Automizdat.

C. Cadunas, K.J. Astrom, K. Braun (1987). Adaptive friction compensation in DC-motor drives. IEEE J. Robot. Automat RA, 3 (6), 681-685.

B. Friedland, Y.J. Park (1992). On adaptive friction compensation. IEEE Trans. Auto. contr. 37 (10), 1609-1612.

B. Armstrong-Helouvry, P. Dupont, C candudas De Wit (1994). A survey of models analysis tools and compensation method for the control of machines with friction. Automatica 30 (7), 1083-1138.

D.E. Koditchek (1984). Natural control of robot arms. Proc. Of IEEE Conference on Decision and Control,733-735.

R.C. Paul (1972). Modeling, trajectory calculation and servoing of a computer controlled arm. Stanford Artificial Intelligence Laboratory, Stanford University, A.I. Memo 177.

J.J.E. Slotine and W. Li (1987). On the adaptive control of robot-manipulators. The International Journal of Robotics Research, vol.6, No.3, 49-59.

Anna Jadlovska (2000). An optimal tracking neuro-controller for nonlinear dynamic system. In IFAC Conference Control system design.

Multi-Layered Learning System for Real Robot Behavior Acquisition

Yasutake Takahashi & Minoru Asada

1. Introduction

One of the main issues of autonomous robots is how to implement a system with learning capability to acquire both varieties of knowledge and behaviors through the interaction between the robot and the environment during its lifetime. There have been a lot of different works on learning approaches for robots to acquire behaviors based on the methods such as reinforcement learning, genetic algorithms, and so on. Especially, reinforcement learning has recently been receiving increased attention as a method for behavior learning with little or no a priori knowledge and higher capability of reactive and adaptive behaviors. However, a simple and straightforward application of reinforcement learning methods to real robot tasks is considerably difficult due to its almost endless exploration of which time easily scales up exponentially with the size of the state/action spaces, which seems almost impossible from a practical viewpoint.

One of the potential solutions might be application of so-called "mixture of experts" proposed by Jacobs and Jordan (Jacobs & Jordan, 1991), in which a set of expert modules learn and one gating system weights the output of the each expert module for the final system output. This idea is very general and has a wide range of applications. However, we have to consider the following two issues to apply it to the real robot tasks:

- **Task decomposition:** how to find a set of simple behaviors and assign each of them to a learning module or an expert in order to achieve the given initial task. Usually, human designer carefully decomposes the long time-scale task into a sequence of simple behaviors such that the one short time-scale subtask can be accomplished by one learning module.
- **Abstraction of state and/or action spaces for scaling up:** the original "mixture of experts" consists of experts and a gate for expert selection. Therefore, no more abstraction beyond the gating module. In order to cope with complicated real robot tasks, more abstraction of the state and/or action spaces is necessary.

Connell and Mahadevan (Connell & Mahadevan, 1993) decomposed the whole behavior into sub-behaviors each of which can be independently learned. Morimoto and Doya (Morimoto & Doya 1998) applied a hierarchical reinforcement learning method by which an appropriate sequence of subgoals for the task is learned in the upper level while behaviors to achieve the subgoals are acquired in the lower level. Hasegawa and Fukuda (Hasegawa & Fukuda, 1999, 2001) proposed a hierarchical behavior controller, which consists of three types of modules, behavior coordinator, behavior controller and feedback controller, and applied it to a brachiation robot. Kleiner et al. (Kleiner et al., 2002) proposed a hierarchical learning system in which the modules at lower layer acquires low level skills and the module at higher layer coordinates them. However, in these proposed methods, the designers have done the task decomposition very carefully in advance, or the constructions of the state/action spaces for higher layer modules are independent from the learned behaviors of lower modules. As a result, it seems difficult to abstract situations and behaviors based on the already acquired learning/control modules.

There are a number of works of automatic task decomposition. Digney (Digney, 1996, Digney, 1998) has proposed Nested Q-learning algorithm that generates hierarchical control structures in a learning system. The task decomposition has been done under two criteria; one criterion is based on the received reinforcement signals, and the other is on the frequency of visits to particular state space locations. However, this work has been applied in a simple grid maze world, therefore the state space is fixed and its size is relatively small so that the frequency heuristics can work. In the case of real robots, the size of state space is huge if the state space consists of all sensory information, and it is very rare to visit the same state frequently. Hengst (Hengst, 2000, Hengst 2002) has proposed a method of generating hierarchical structure from state variables based on a heuristics that the almost constant variables represent higher-level states while the frequently changing variables represent lower level states. However, the designer gives these hierarchized variables and usually we cannot expect that real robots have such abstracted variables beforehand.

A basic idea to cope with the above two issues is that any learning module has limited resource constraint, and this constraint of the learning capability leads us to introduce a multi-module and multi-layered learning system. That is, one learning module has a compact state-action space and acquires a simple map from the states to the actions, and a gating system enables the robot to select one of the behavior modules depending on the situation. More generally, the higher module controls the lower modules depending on the situation. The definition of this situation depends on the capability of the lower modules because the gating module selects one of the lower modules based on their acquired behaviors. From the other viewpoint, the lower modules provide not only the rational behaviors but also the abstracted situations for the higher module; how feasible the module is, how close to its subgoal, and so on. It is reasonable to utilize such information in order to construct state/action spaces of higher modules from already abstracted situations and behaviors of lower ones. Thus, the hierarchical structure can be constructed with not only experts and gating module but also more layers with multiple homogeneous learning modules.

In this paper, we show a series of studies towards the construction of such hierarchical learning structure developmentally. The first one (Takahashi & Asada, 2000) is automatic construction of hierarchical structure with purely homogeneous learning modules. Since the resource (and therefore the capability, too) of one learning module is limited, the initially given task is automatically decomposed into a set of small subtasks each of which corresponds to one of the small learning modules, and also the upper layer is recursively generated to cover the whole task. In this case, the all learning modules in the one layer share the same state and action spaces although some modules need the part of them. Then, the second work (Takahashi & Asada, 2001) and third one (Takahashi et al., 2003a) focused on the state and action space decomposition according to the subtasks to make the learning much more efficient. Further, the forth one (Takahashi et al, 2003b) realized unsupervised decomposition of a long time-scale task by finding the compact state spaces, which consequently leads the subtask decomposition. We have applied these methods to simple soccer situations in the context of RoboCup (Asada et al., 1998) with real robots, and show the experimental results.

2. Multi-Layered Learning System

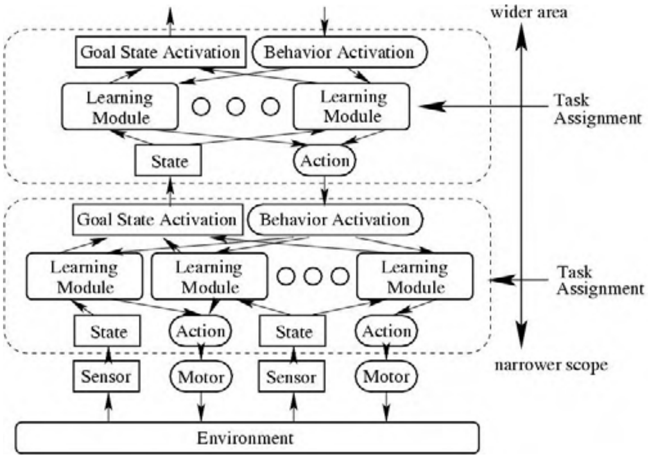


Figure 1. Hierarchical architecture in multi-layered learning system

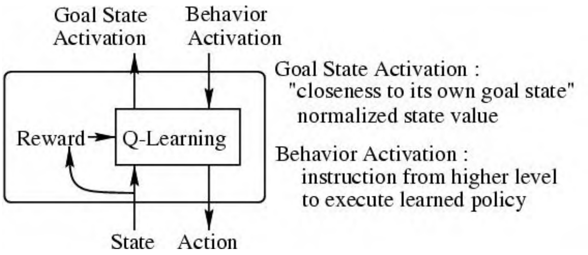


Figure 2. Behavior Learning Module

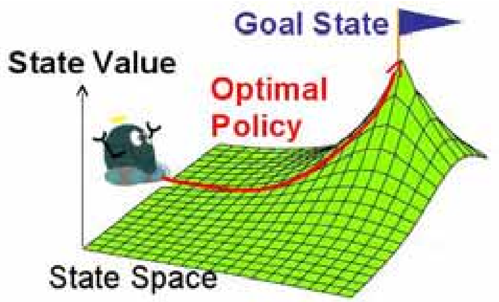


Figure 3. Sketch of a state value function

Figs. 1 and 2 show the architecture of the multi-layered reinforcement learning system, in which indicate a hierarchical architecture with two levels, and an individual learning module embedded in the layers are indicated. Each module has its own goal state in its state space, and it learns the behavior to reach the goal, or maximize the sum of the discounted reward received over time, using *Q*-learning method. The state and the action are constructed using sensory information and motor commands, respectively at the bottom level. The input and output to/from the higher level are the goal state activation and the behavior activation, respectively, as shown in Fig. 2. The goal state activation *g* is a normalized state value[1], and *g* = *1* when the situation is the goal state. When the module receives the behavior activation *b* from the higher modules, it calculates the optimal policy for its own goal, and sends action commands to the lower module. The action command at the bottom level is translated to an actual motor command, and then the robot takes the action in the environment.

One basic idea is to use the goal state activations *g* of the lower modules as the representation of the situation for the higher modules. Fig. 3 shows a sketch of a state value function where a robot receives a positive reward one when it reaches to a specified goal. The state value function can be regarded as closeness to the goal of the module. The states of the higher modules are constructed using the patterns of the goal state activations of the lower modules. In contrast, the actions of the higher-level modules are constructed using the behavior activations to the lower modules.

3. Behavior Acquisition on Multi-Layered System (Takahashi & Asada 2000)

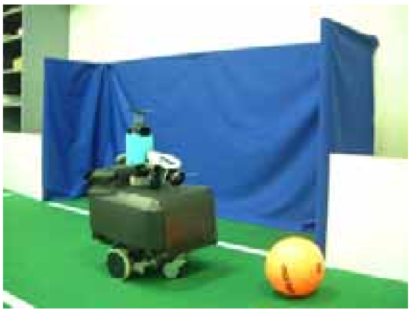


Figure 4. Experimental instruments

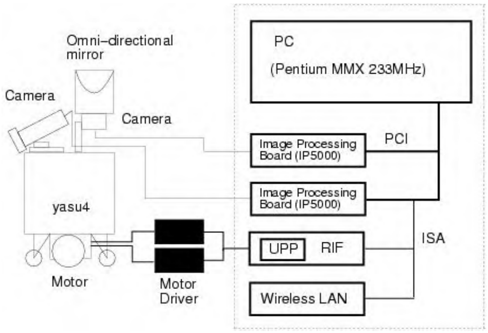


Figure 5. Overview of the robot system

Fig. 4 shows a picture of a mobile robot that we designed and built, a ball, and a goal, and Fig. shows an overview of the robot system. It has two TV cameras: one has a wide-angle lens, and the other an omni-directional mirror. The driving mechanism is PWS (Powered Wheels Steering) system, and the action space is constructed in terms of two torque values to be sent to two motors that drive two wheels. These parameters of the system are unknown to the robot, and it tries to estimate the mapping from the sensory information to the appropriate motor commands by the method. The environment consists of the ball, the goal, and the mobile robot.

[1] The state value function estimates the sum of the discounted reward received over time when the robot takes the optimal policy, and is obtained by Q learning.

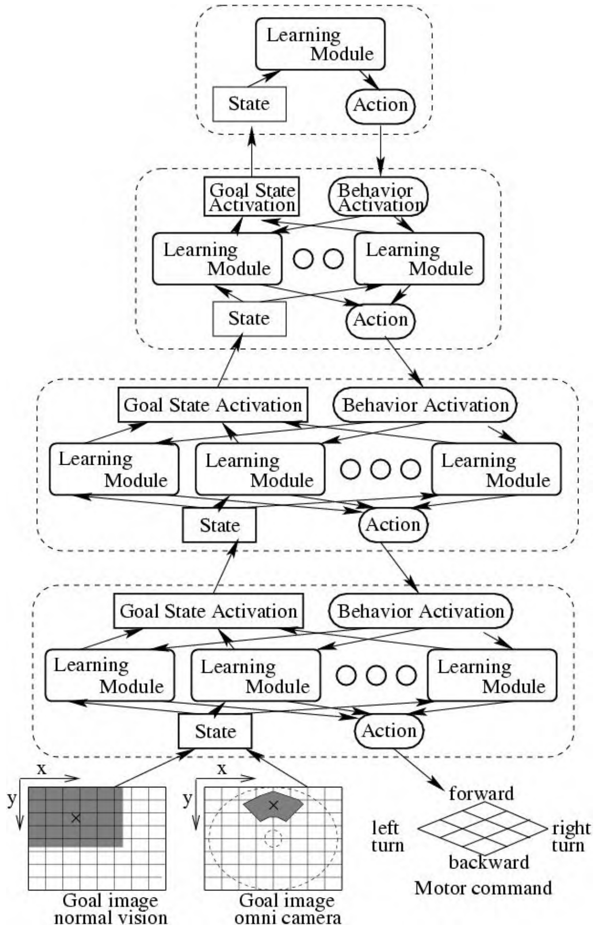


Figure 6. A hierarchical architecture on a monolithic state space

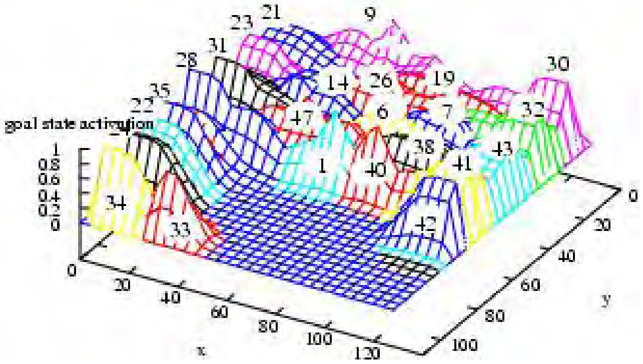


Figure 7. The distribution of learning modules at bottom layer on the normal camera image

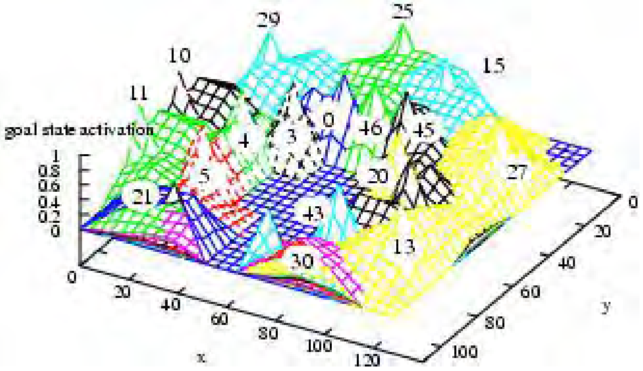


Figure 8. The distribution of learning modules at bottom layer on the omni-directional camera image

In this experiment, the robot receives the information of only one goal, for the simplicity. The bottom of Fig. 6. show a sketch of the state and action spaces of the bottom layer in the multi-module learning system. The state space is constructed in terms of the centroids of goal images of the two cameras and is tessellated both into 9 by 9 grids each. The action space is constructed in terms of two torque values to be sent to two motors corresponding to two wheels and is tessellated into 3 by 3 grids. Consequently, the numbers of states and actions are 162(9 x 9 x 2) and 9(3 x 3), respectively. The state and action at the upper layer is constructed by the learning modules at the lower layer which are automatically assigned.

The experiment is constructed with two stages: the learning stage and the task execution one. First of all, the robot moves at random in the environment for about two hours. The system learns and constructs the four layers and one learning module is assigned at the top layer (Fig. 6). We call each layer from the bottom, "bottom", "middle", "upper", and "top" layers. In this experiment, the system assigned 40 learning modules at the bottom layer, 15 modules at the middle layer, and 4 modules at the upper layer. Figs. 7 and 8 show the distributions of goal state activations of learning modules at the bottom layer in the state spaces of wide-angle camera image and omni-directional mirror image, respectively. The x and y axes indicate the centroid of goal region on the images. The numbers in the figures indicate the corresponding learning module numbers. The figures show that each learning module is automatically assigned on the state space uniformly.

Fig. 9 shows a rough sketch of the state transition and the commands to the lower layer on the multi-layer learning system during navigation task. The robot was initially located far from the goal, and faced the opposite direction to it. The target position was just in front of the goal. The circles in the figure indicate the learning modules and their numbers. The empty up arrows (broken lines) indicate that the upper learning module recognizes the state which corresponds to the lower module as the goal state. The small solid arrows indicate the state transition while the robot accomplished the task. The large down arrows indicate that the upper learning module sends the behavior activation to the lower learning module.

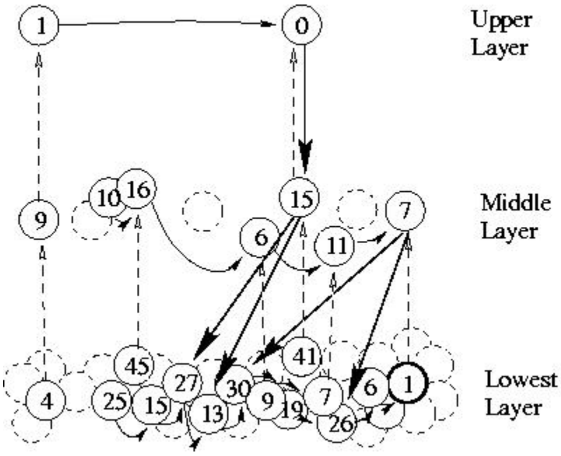


Figure 9. A rough sketch of the state transition on the multi-layer learning system

4. State Space Decomposition and Integration (Takahashi & Asada, 2001)

The system mentioned in the previous section dealt with a whole state space from the lower layer to the higher one. Therefore, it cannot handle the change of the state variables because the system suppose that all tasks can be defined on the state space at the bottom level.

Further, it is easily caught by a curse of dimension if number of the state variables becomes large. Here, we introduce an idea that the system constructs a whole state space with several decomposed state spaces. At the bottom level, there are several decomposed state spaces in which modules are assigned to acquire the low level behaviors in the small state spaces. The modules at the higher level manage the lower modules assigned to different state spaces. In this paper, we define the term “layer” as a group of modules sharing the same state space, and the term “level” as a class in the hierarchical structure. There might be several layers at one level (see Fig. 10).

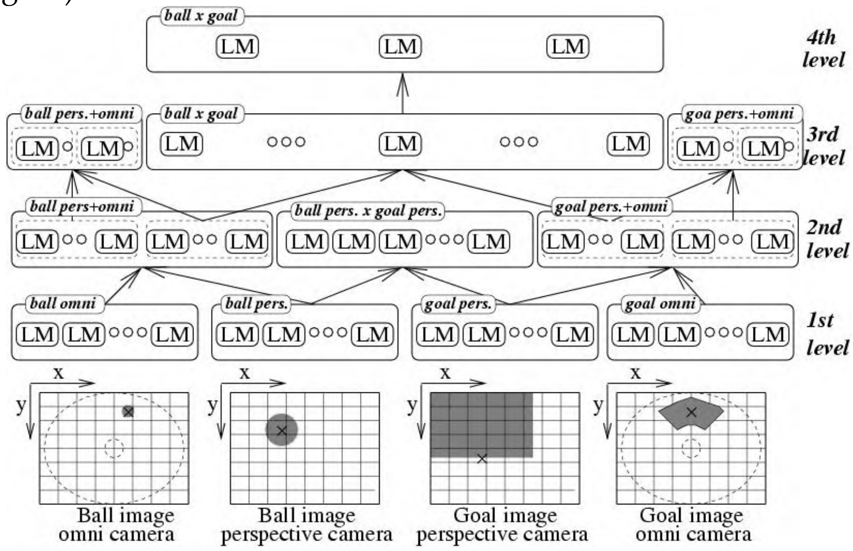


Figure 10. A hierarchical structure of learning modules

When the higher layer constructs its state-action space based on situations and behaviors acquired by the modules of several lower layers, it should consider that the layers are independent from each other, or there is dependence between them. The layer might be basically independent from each other when the each layer's modules recognize different objects and learn behaviors for them. For example, in the case of robot in the RoboCup field, one layer's modules could be the experts of ball handling and the other layer's modules the one of navigation on the field. In such a case, the state space is constructed as direct product of module's activations of lower layers. We call this way of state space construction “a multiplicative approach”.

On the other hand, there might be dependence between the layers when modules on both layers recognize the same object in the environment with different logical sensor outputs. For example, our robot recognizes an object with both perspective vision system and omni-directional one. In such a case, the system can recognize the situation complementary using plural layers' outputs even if one layer loses the object on its own state spaces. We call this way of state space construction “a complementary approach”.

Fig. 10 shows an example hierarchical structure. At the lowest level, there are four learning layers, and each of them deals with its own logical sensory space (ball positions on the perspective camera image and omni one, and goal position on both images). At the second level, there are three learning layers in which one adopts the multiplicative approach and the two others adopt the complementary approach. The multiplicative approach of the “*ball pers. x goal pers*” layer deals with lower modules of “*ball pers.*” and “*goal pers.*” layers. The arrows in the figure indicate the flows from the goal state activations to the state vectors. The arrows from the action vectors to behavior activations are eliminated. At the third level, the system has three learning layers in which one adopts the multiplicative approach and the others adopt the complementary approach, again. At the levels higher than third layer, the learning layer is constructed as described in the previous section.

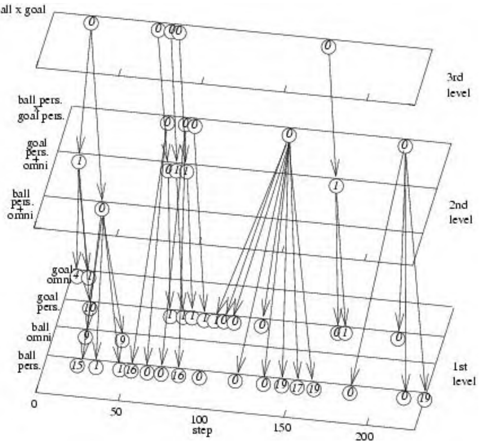


Figure 11. A sequence of the behavior activation of learning modules and the commands to the lower layer modules

After the learning stage, we let our robot do a couple of tasks. One of them is shooting a ball into the goal using this multi-layer learning structure. The target situation is given by reading the sensor information when the robot pushes the ball into the goal; the robot captures the ball and goal at center bottom in the perspective camera image. As an initial position, the robot is located far from the goal, faced opposite direction to it. The ball was located between the robot and the goal. Fig. 11 shows the sequence of the behavior activation of learning modules and the commands to the lower layer modules. The down arrows indicate that the higher learning modules fire the behavior activations of the lower learning modules.

5. Behavior Segmentation and Coordination

Fig. 12 shows a picture of a soccer robot for middle size league of RoboCup we designed and built, recently. The driving mechanism is PWS, and it equips a pinball like kicking device in front of the body (see Fig. 13). These days, many robots have number of actuators such as navigation devices and object manipulators, and have a capability of execution of many kinds of tasks by coordinating these actuators. If one learning module has to manipulate all actuators simultaneously, the exploration space of action scales up exponentially with the number of the actuators, and it is impractical to apply a reinforcement learning system.

Fortunately, a complicated behavior which needs many kinds of actuators might be often generally decomposed into some simple behaviors each of which needs small number of actuators. The basic idea of this decomposition is that we can classify them based on aspects of the actuators. For example, we may classify the actuators into navigation devices and manipulators, then the some of behaviors depend on the navigation devices tightly, not on the manipulators, while the others depend on manipulators, not on the navigation. The action space based on only navigation devices seems to be enough for acquisition of the former behaviors, while the action space based on manipulator would be sufficient for the manipulation tasks. If we can assign learning modules to both action spaces and integrate them at higher layer, much smaller computational resources is needed and the learning time can be reduced significantly.

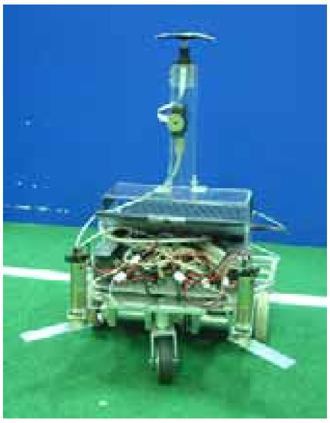


Figure 12. Robot with kicking devices

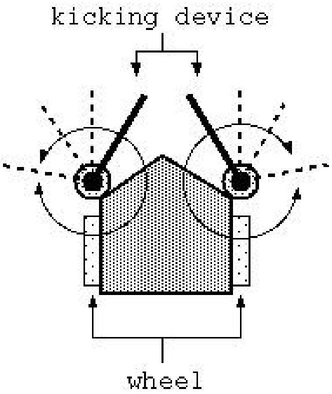


Figure 13. Configuration of kicking device and wheels

We have implemented two kind of hierarchical system to check the basic idea. Each system has been assigned a task. One is placing the ball in the center circle (task 1), and the other is shooting the ball into the goal (task2).

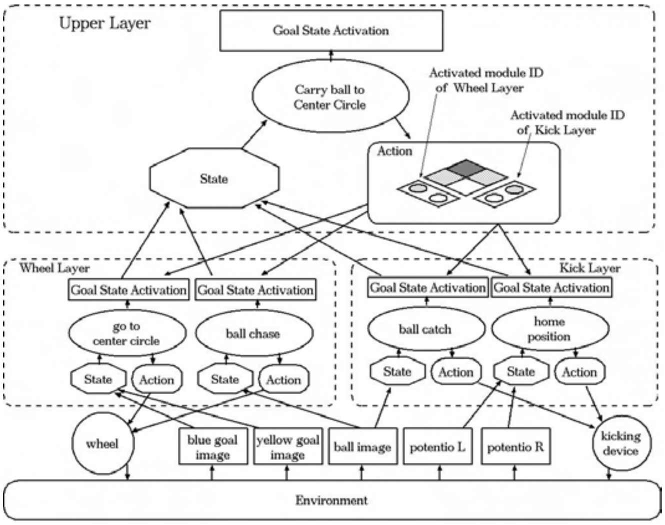


Figure 14. Hierarchical learning system for task 1

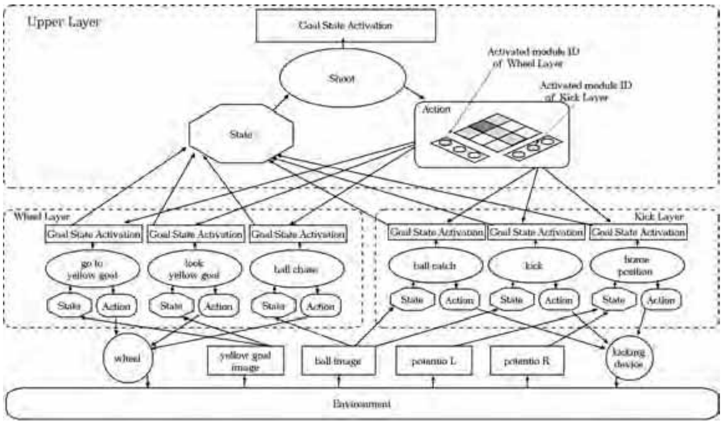


Figure 15. Hierarchical learning system for task 2

We have prepared the following subtasks for the vehicle: ``*Chasing a ball*'', ``*Looking the goal in front of the body*'', ``*Reaching the center circle*'', and ``*Reaching the goal*''. We have also prepared the following subtasks for the kicking device: ``*Catching the ball*'', ``*Kicking the ball*'', and ``*Setting the kicking device to the home position*''. Then, the upper layer modules integrates these lower ones.

After the learner acquired low level behaviors, it puts new learning modules at higher layer as shown in Figs. 16 and 17, and learn two kinds of behaviors.

Fig. 16 shows the sequence of the goal state activations of lower modules and behavior commands to the lower ones. At the start of this behavior, the robot activates the module of setting home position behavior for the kicking device and ball chasing module for the vehicle at lower layer. The robot reaches the ball, then it activates the module of catching the ball for kicking device and the module of reaching the center circle. Then, it achieves the task of placing a ball to the center circle.

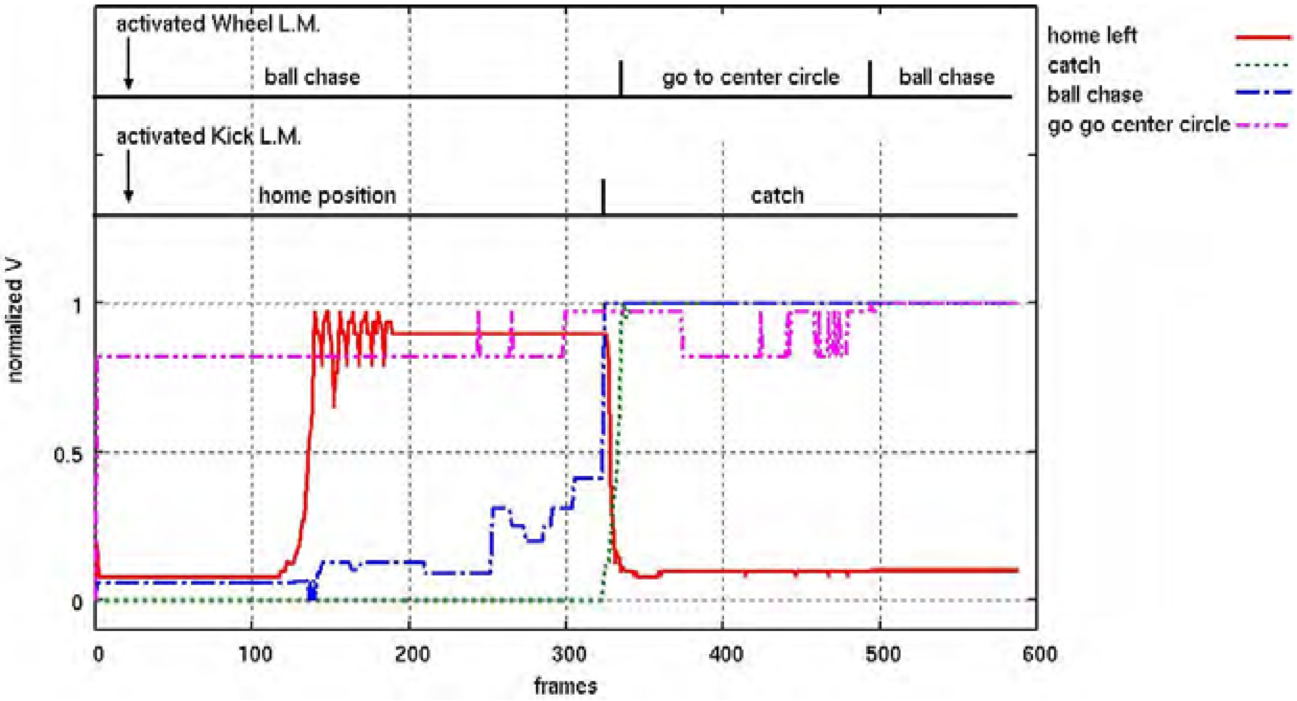


Figure 16. A sequence of the goal state activations and behavior commands (Task 1)

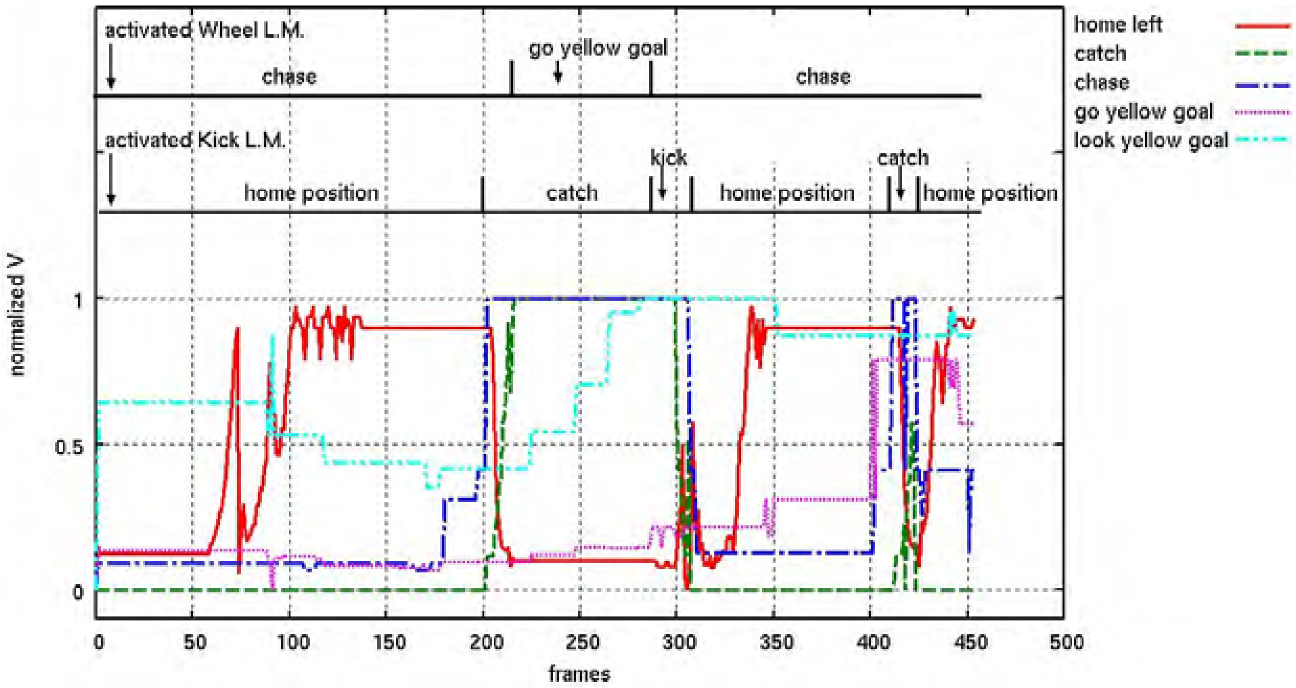


Figure 17. A sequence of the goal state activation and behavior activation (Task 2)

Figs. 17 and 18 shows the sequence of the goal state activations of lower modules and behavior commands to the lower ones and the scene sequence of a real robot experiment while the robot shoots a ball into a goal. At the start of this behavior (Fig.18-1), the robot activates the module of setting home position behavior for the kicking device and ball chasing module for the vehicle at lower layer (Fig.18-2,3). The robot reaches the ball (Fig.18-4,5), then it activates the module of catching the ball for kicking device and the module of reaching the goal for the vehicle (Fig.18-6). When the robot captures the goal in front of the body and gets near to the goal (Fig.18-7), it activates the module of kicking the ball, then successfully shoots the ball into the goal (Fig.18-7).

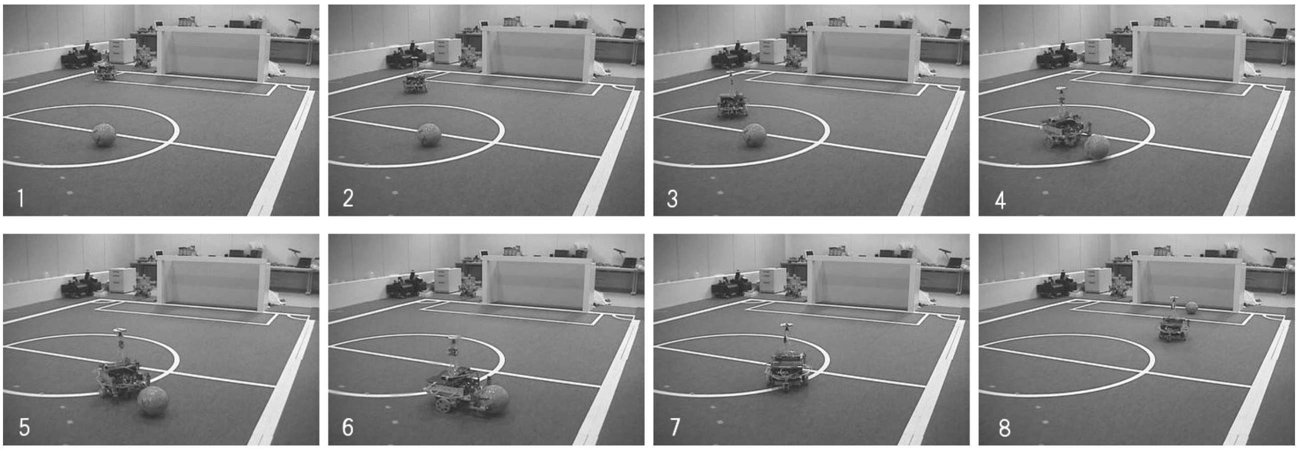


Figure 18. A sequence of an acquired behavior (Shooting)

6. Task Decomposition Based on Self-interpretation of Instruction by Coach (Takahashi & Asada 2003)

When we develop a real robot which learns various behaviors in its life, it seems reasonable that a human instructs or shows some example behaviors to the robot in order to accelerate the learning before it starts to learn. We proposed a behavior acquisition method based on hierarchical multi-module leaning system with self-interpretation of coach instructions. The proposed method enables a robot to

1. decompose a long term task into a set of short term subtasks,
2. select sensory information needed to accomplish the current subtask,
3. acquire a basic behavior to each subtask,
4. and integrate the learned behaviors to a sequence of the behaviors to accomplish the given long term task.

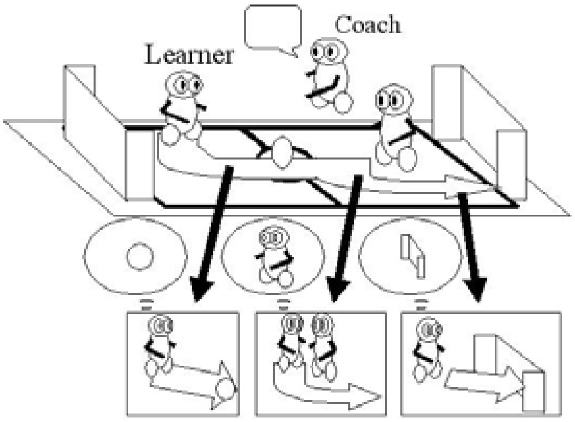


Figure 19. Basic concept: A coach gives instructions to a learner. The learner follows the instruction and finds basic behaviors by itself

Fig.19 shows a rough sketch of the basic idea. There are a learner, an opponent, and a coach in a simple soccer situation. The coach has *a priori* knowledge of tasks to be played by the learner. The learner does not have any knowledge on tasks but just follows the instructions. In Fig. 19, the coach shows a instruction of shooting a ball into a goal without collision to an opponent. After some instructions, the learner segments the whole task into a sequence of subtasks, acquires a behavior for each subtask, finds the purpose of the instructed task, and acquire a sequence of the behaviors to accomplish the task by itself. When the coach gives new instructions, the learner reuses the learning modules for familiar subtasks, generates new learning modules for unfamiliar subtasks at lower level. The system generates a new module for a sequence of behaviors of the whole instructed task at the upper level.

Fig. 20 shows a rough sketch of the idea of the task decomposition procedure. The top of the Fig. 20 shows a monolithic state space that consists of all state variables (*x*1, *x*2, …, *xn*). The red lines indicate sequences of state value during the given instructions. As we assume beforehand, the system cannot have such a huge state space, then, decomposes the state space into subspaces that consist of a few state variables. The system regards that the ends of the instructions represent goal states of the given task. It checks all subspaces and selects one in which the most ends of the instruction reach a certain area (*Gtask* in Fig. 20). The system regards this area as the subgoal state of a subtask which is a part of the given long-term task. The steps of the procedure are as follows:

1) find module unavailable areas in the instructions and regard them as unknown subtask.
2) assign a new learning module.
   a) list up subgoal candidates for the unknown subtasks on the whole state space.
   b) decompose the state space into subspaces that consist of a few state variables.
   c) check all subspaces and select one in which the subgoal candidates reach a certain area best (*Gsub* in Fig. 3).
   d) generate another learning module with the selected subspace as a state space and the certain area as the goal state.
3) check the areas where the assigned modules are available.
4) exit if the generated modules cover all segments of instructed behaviors. Else goto 1.

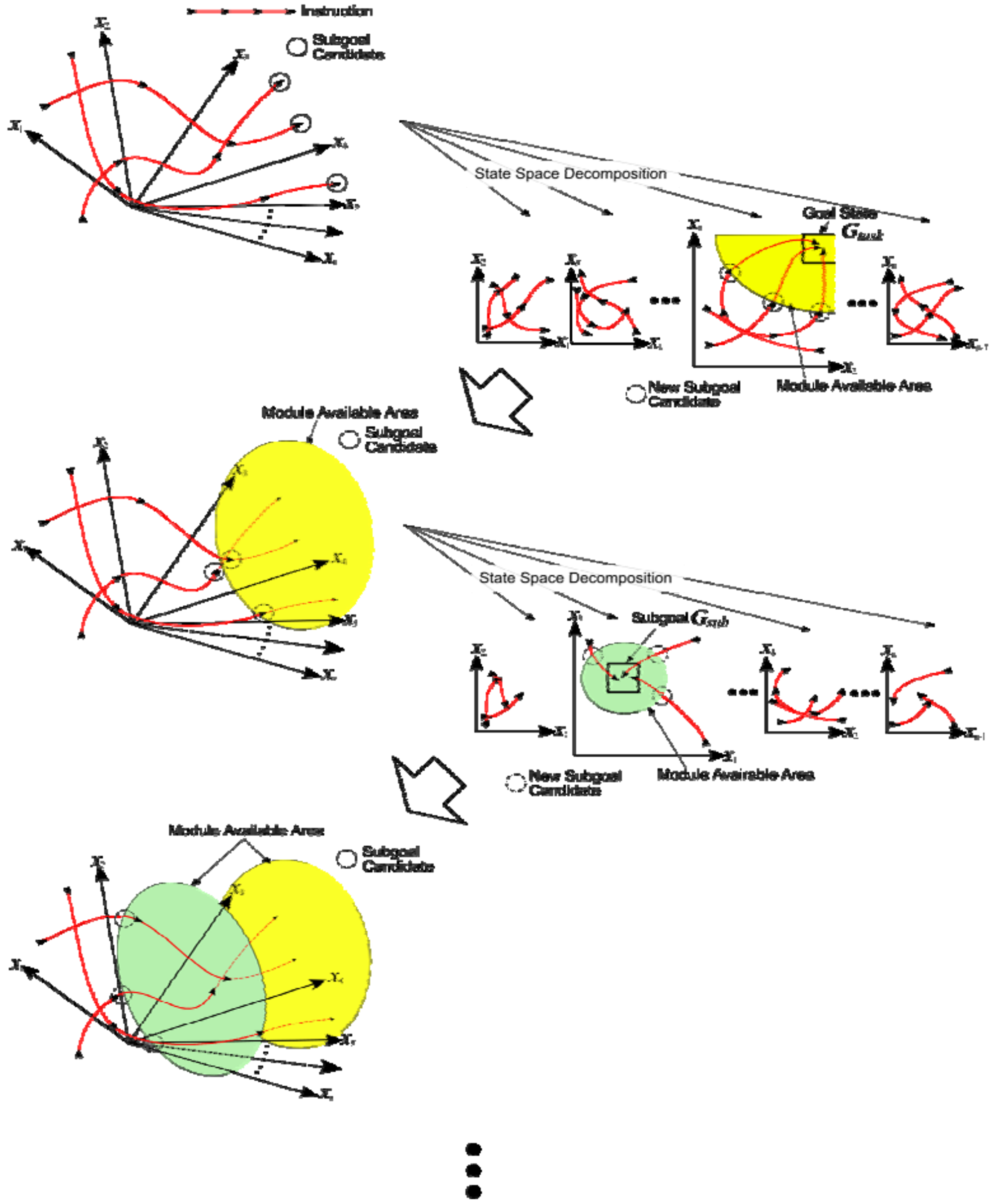
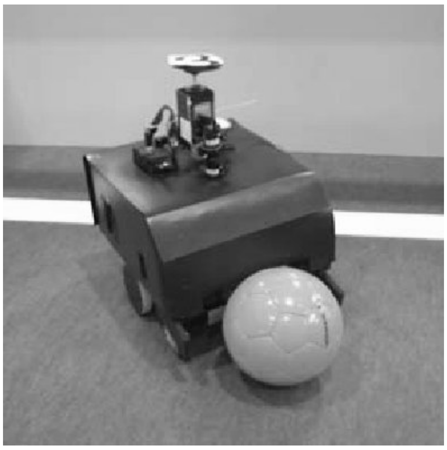


Figure 20. Rough sketch of the idea of task decomposition procedure

The details are described in (Takahashi & Asada, 2003).Fig. 21 shows the mobile robot and a situation with which the learning agent can encounter. The robot has an omni-directional camera system. A simple color image processing is applied to detect the ball area and an opponent one in the image in real-time (every 33ms).



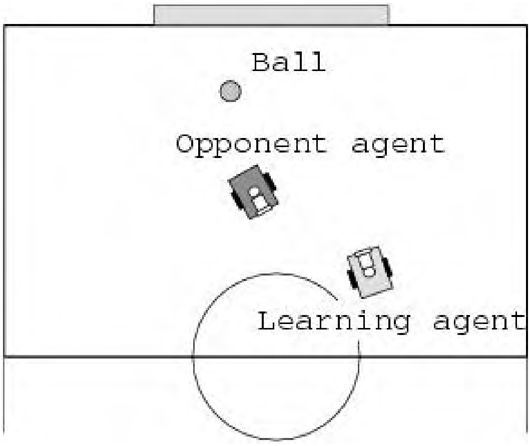


Figure 21. A real robot and a ball (left), and a top view of the simulated environment (right)

The robot receives instructions for the tasks in the order as follows:

Task 1: chasing a ball

Task 2: shooting a ball into a goal without obstacles

Task 3: shooting a ball into a goal with an obstacle

Figs. 22, 23, and 24 show the ones of the example behaviors for task 1, 2, and 3, respectively. Figs. 25, 26, and 27 show the constructed systems after the learning of the tasks. First of all, the coach gives some instructions for the ball chasing task (task 1). The system produce one module which acquired the behavior of ball chasing (Fig.25). At the second stage, the coach gives some instructions for the shooting task (task 2). The learner produces another module which has a policy of going around the ball until the directions to the ball and the goal become same (Fig.26). At the last stage, the coach gives some instructions for the shooting task with obstacle avoidance (task 3). The learner produces another module which acquired the behavior of going to the intersection between the opponent and the goal avoiding the collision (Fig.27). Fig.28 shows a sequence of a acquired behavior of the real robot for task 3.

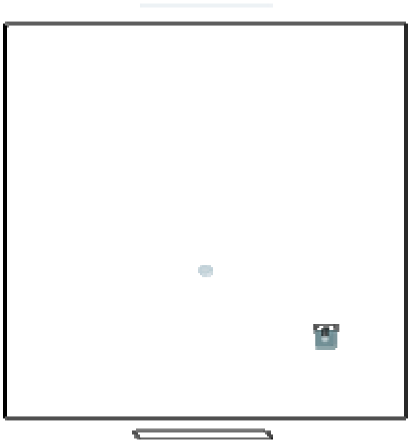


Figure 22. One of the example behaviors for task 1

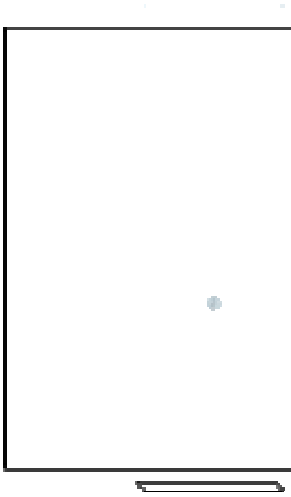


Figure 23. One of the example behaviors for task 2

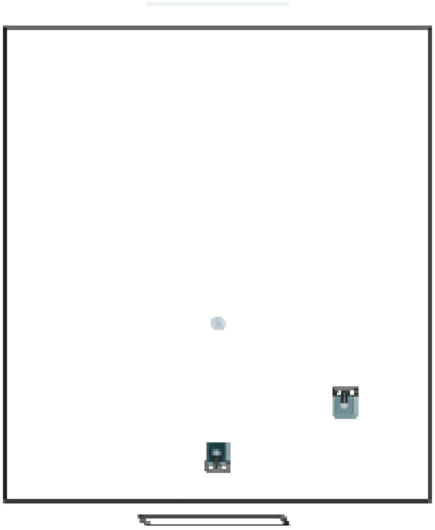


Figure 24. One of the example behaviors for task 3

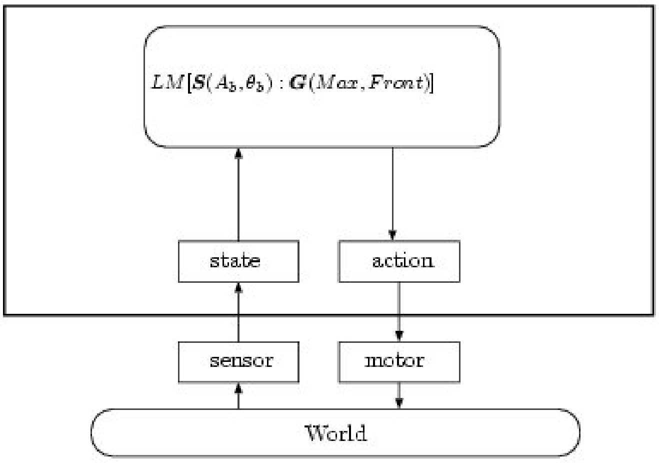

$LM[\boldsymbol{S}(A_b,\theta_b):\boldsymbol{G}(Max,Front)]$
state
action
sensor
motor
World

Figure 25. Acquired learning module for task 1

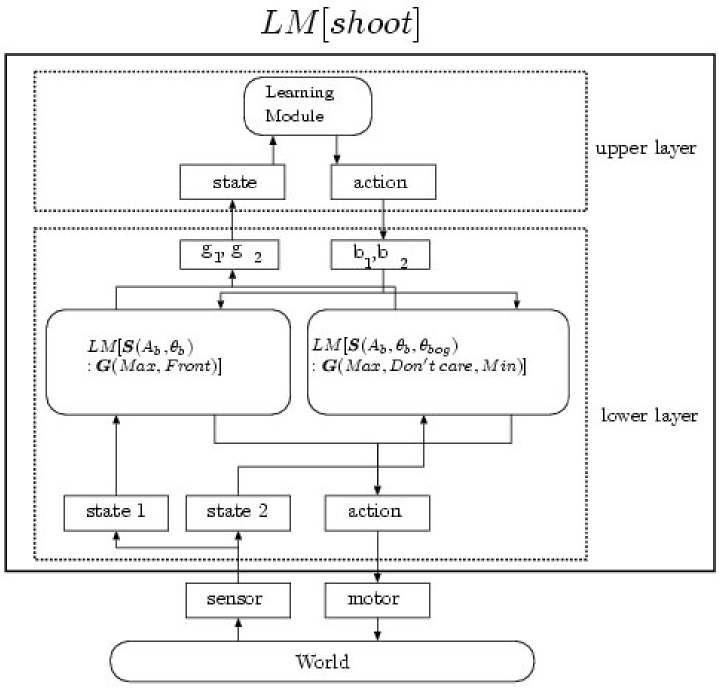


Figure 26. Acquired hierarchical structure for task 2

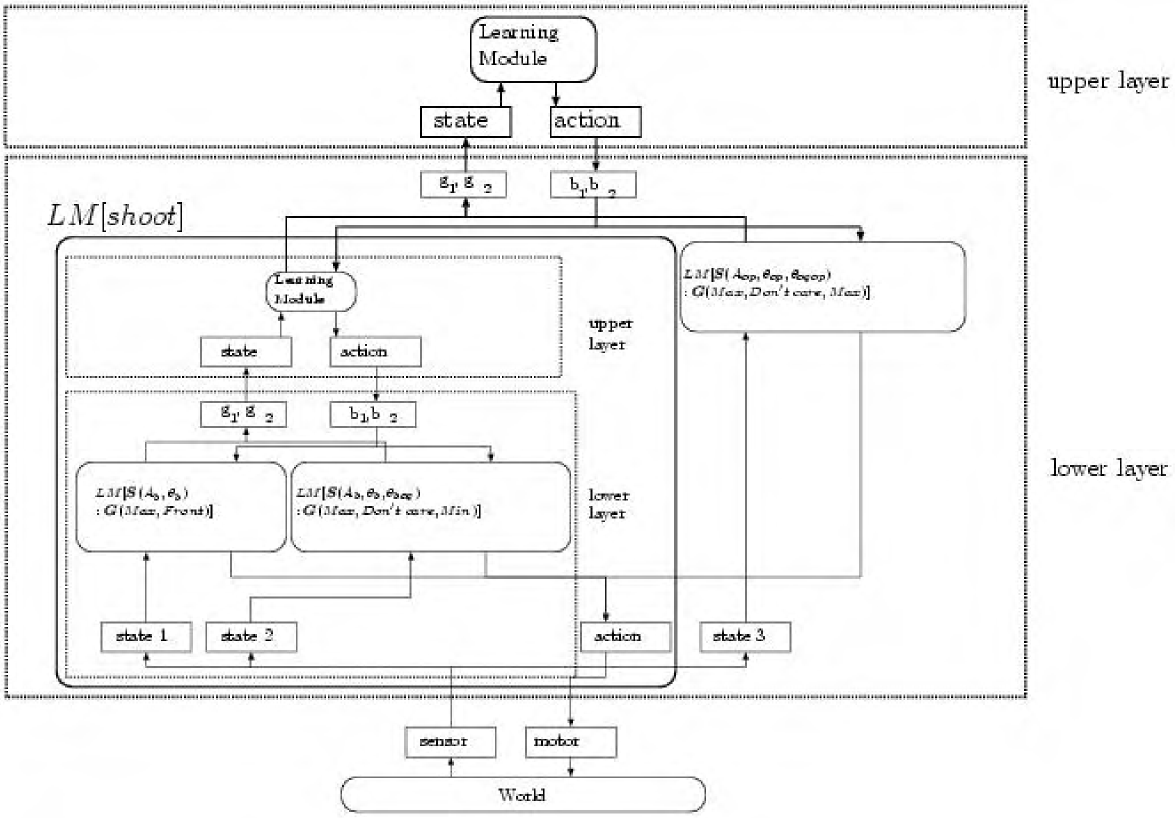


Figure 27. Acquried heirarchical structure for task 3

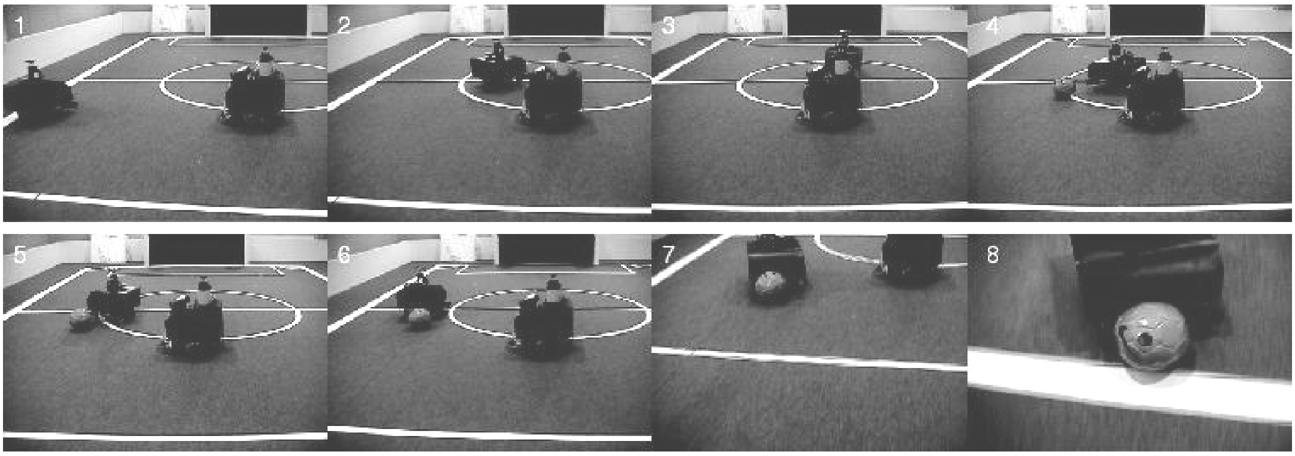


Figure. 28. A sequence of real robot behavior : shooting a ball into a goal with an obstacle (task3)

7. Discussion

We showed a series of approaches to the problem of decomposing the large state action space at the bottom level into several subspaces and merging those subspaces at the higher level. As future works, there are a number of issues to extend our current methods.

Interference between modules

One module behavior might have inference to another one which has different actuators. For example, the action of a navigation module will disturb the state transition from the view point of the kicking device module; the catching behavior will be success if the vehicle stays while it will fail if the vehicle moves.

Self-assignment of modules

It is still an important issue to find a purposive behavior for each learning module automatically. In the paper (Takahashi & Asada, 2000), the system distributes modules on the state space uniformly, however, it is not so efficient. In the paper (Takahashi & Asada, 2003), the system decomposes the task by itself, however, the method uses many heuristics and needs instruction from a coach. In many cases, the designers have to define the goal of each module by hand based on their own experiences and insights.

Self-construction of hierarchy

Another missing point in the current method is that it does not have the mechanism that constructs the learning layer by itself.

8. References

Asada, M.; Kitano, H.; Noda, I. & Veloso, M. (1999). RoboCup: Today and tomorrow – what we have learned. *Artificial Intelligence,* 193–214.

Connell, J. H. & Mahadevan, S. 1993. *ROBOT LEARNING.* Kluwer Academic Publishers. chapter “RAPID TASK LEARNING FOR REAL ROBOTS”.

Digney, B. L., “Emergent hierarchical control structures: Learning reactive/hierarchical relationships in reinforcement environments, In *From animals to animats 4: Proceedings of The fourth conference on the Simulation of Adaptive Behavior: SAB 96* (P. Maes, M. J. Mataric, J.-A. Meyer, J. Pollack, and S. W. Wilson, eds.), pp. 363–372, The MIT Press, 1996.

Digney, B. L., "Learning hierarchical control structures for multiple tasks and changing environments," in *From animals to animats 5: Proceedings of The fifth conference on the Simulation of Adaptive Behavior: SAB 98* (R. Pfeifer, B. Blumberg, J.-A. Meyer, and S. W. Wilson, eds.), pp. 321–330, The MIT Press, 1998.

Hasegawa, Y. & Fukuda, T. (1999). Learning method for hierarchical behavior controller. In *Proceedings of the 1999 IEEE International Conference on Robotics and Automation*, 2799–2804.

Hasegawa, Y.; Tanahashi, H. & Fukuda, T. (2001). Behavior coordination of brachation robot based on behavior phase shift. In *Proceedings of the 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume CD-ROM, 526–531.

Hengst, B, "Generating hierarchical structure in reinforcement learning from state variables," in *6th Pacific Rim International Conference on Artificial Intelligence (PRICAI 2000)* (R. Mizoguchi and J. K. Slaney, eds.), vol. 1886 of *Lecture Notes in Computer Science*, Springer, 2000.

Hengst, B., "Discovering hierarchy in reinforcement learning with HEXQ," in *Proceedings of the Nineteenth International Conference on Machine Learning (ICML02)*, pp. 243–250, 2002.

Jacobs, R.; Jordan, M.; S, N. & Hinton, G. (1991). Adaptive mixture of local experts. *Neural Computation* 3:79–87.

Kleiner, A.; Dietl, M. & Nebel, B. (2002). Towards a life-long learning soccer agent. In Kaminka, G. A.; Lima, P. U.; and Rojas, R., eds., *The 2002 International RoboCup Symposium Pre-Proceedings*, CD–ROM.

Morimoto, J., and Doya, K. (1998). Hierarchical reinforcement learning of low-dimensional subgoals and highdimensional trajectories. In *The 5th International Conference on Neural Information Processing*, volume 2, 850–853.

Takahashi, Y. & Asada, M. (2000). Vision-guided behavior acquisition of a mobile robot by multi-layered reinforcement learning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 1, 395–402.

Takahashi, Y. & Asada, M. (2001). Multi-controller fusion in multi-layered reinforcement learning. In *International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI2001)*, 7–12.

Takahashi, Y. & Asada, M. 2003. Multi-layered learning systems for vision-based behavior acquisition of a real mobile robot. In *Proceedings of SICE Annual Conference 2003 in Fukui*, volume CD-ROM, 2937–2942.

Takahashi, Y.; Hikita, K. & Asada, M. 2003. Incremental purposive behavior acquisition based on self-interpretation of instructions by coach. In *Proceedings of 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems*, CD–ROM.

-V-

Multi-Robot Systems

The Design of a Pair of Identical Mobile Robots to Investigate Co - Operative Behaviours

Dale Carnegie, Andrew Payne & Praneel Chand

1. Introduction

When moving large or heavy items, the traditional tendency with machinery is to build a large mechanism capable of handling the load. This leads to continuous scaling of the mechanical size and weight of devices being built, with a proportional increase in expense. An alternative to this conventional trend is the use of cooperative behaviour. Humans are limited by their physical size and strength – and yet are capable of moving heavy furniture and other large items outside their individual strength ability. This is accomplished with the help of another person to share the load, or a large number of people for a substantial object.

Requirements of coordination, communication and precision control have restricted this approach being considered for mobile robots in the past. However, more powerful processors and software are removing these barriers, permitting cooperative robots to be seriously considered for a number of applications.

By working in parallel, cooperative behaviour can increase efficiency and reduce the time required to complete a task. Reliability is increased by introducing redundancy when using a team of robots, while cost is reduced due to the use of smaller simplistic machine designs. Application specific design and manufacturing costs can be removed by fabricating the robots semi-generically. Reduced weight means less preparation and upkeep of the working environment, and new complex tasks can be introduced which are too difficult for a single entity to achieve.

We demonstrate a methodology for the construction of a low-cost, versatile, highly manoeuvrable and computationally powerful robot capable of autonomous operation. We demonstrate that these robots can be fabricated at a small fraction of the price of equivalent commercial systems. As our interest is in the development of cooperative robotic behaviour, the software control system, machine interface and an introduction to cooperative robotic systems are also discussed.

1.1 Commercial Mobile Platforms

Most robotic arms are designed to sit on a factory floor, are large, heavy and use a substantial amount of power. A small number of suitably designed mobile platforms with a mounted manipulator arm are made by manufacturers around the world. They are mainly used for research or remote bomb disposal and are sparse and expensive.

The MR-5, shown in Figure 1, is a bomb disposal robot built and sold by ESI (Engineering Service Inc., http://www.esit.com). It is a remote controlled platform, with a manipulator arm consisting of up to eight joints that can reach 1.7 m supporting a payload of 20 kg. It has dimensions 70 × 127 × 79 cm (w × l × h), weighs 250 kg, a maximum speed of 2.5 km/h and can operate for 3 – 5 hours. The system retails for US$80 K to $140 K depending on the configuration.

The MR-5 is controlled by a human operator via a remote station. The communication is either wireless RF (Radio Frequency) with a 500 m range (line-of-sight), or cable with a range of 200 m. Tracks can be fitted over the wheels for a rougher environment. Optional sensors include a pan-and-tilt zoom camera providing video and sound feedback to the user through the control station, an infrared camera, and an x-ray mounting assembly.

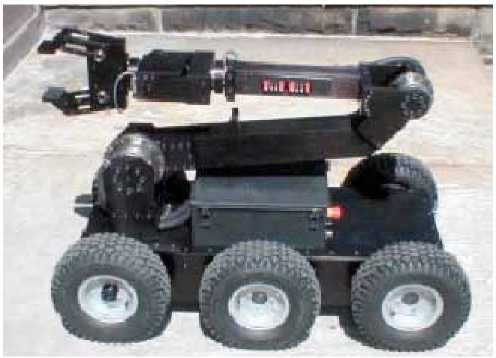


Figure 1. ESI's MR-5

RoboProbe Technologies Inc. (http://www.roboprobe.com) produce a remote controlled bomb disposal robot shown in Figure 2. The robot is similar to the MR-5, but is smaller (43.5 × 91.5 × 40.5 cm), lighter (35 - 45 kg) and with an operating time of 2 – 3 hours. Permanent tracks are used rather than wheels, giving it a similar maximum speed to the MR-5. The arm has a reach of 96.5 cm and can support a 4 kg payload. Incorporated sensors are 3 cameras, an IR (Infrared) camera with pan-and-tilt capability and IR light source (on a high mounting), colour drive camera with halogen lights (at the base of the arm), and a colour camera mounted on the claw.

The MURV-100 is similar to the MR-5, but very lightweight (only 23 kg). Shown in Figure 3 fitted with ten wheels, these can be changed for either six large wheels or tracks. It has dimensions: 43 × 58 × 30 cm, an operating time of 2 – 4 hrs and a modest top speed of 0.39 km/hr. The arm can extend 66 cm and support a 4.5 kg payload. Approximate cost is US$35 K.

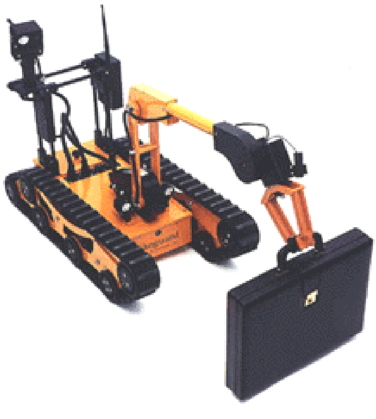


Figure 2. RoboProbe Technologies Inc. bomb disposal robot

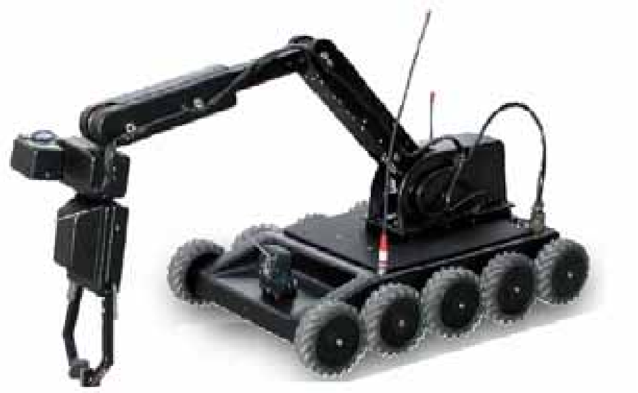


Figure 3. The lightweight MURV-100

Manufactured by Defenders Network Inc. (http://www.defend-net.com), the MURV-100 offers communication by either a cable or by a standard wireless system, up to 300 m in each case (line of sight required for wireless operation). Optional sensors include a tilt-and-pan camera, a claw camera, and sensors capable of detecting toxic gases, radioactive materials and explosives.

The PowerBot (ActivMedia Robotics, http://www.activrobots.com) fitted with a PowerArm, (Figure 4) is designed for research use. A number of sensors are incorporated into the design to complement this, allowing control to be provided by the robot itself rather than from a remote user. It is designed for indoor use, with a small ground clearance and limited traction on rough terrain. It has dimensions 62.5 × 85 × 43 cm and a maximum speed of 6 km/hr. The arm can reach 80 cm, and support a 2 kg payload

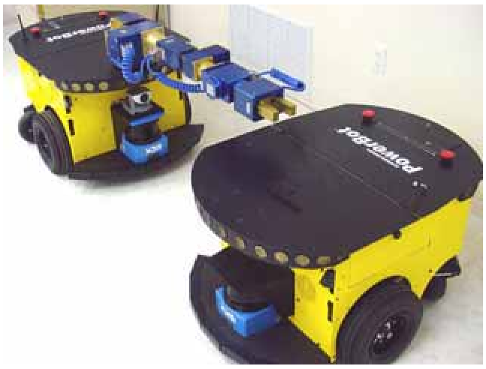


Figure 4. PowerBot with PowerArm from Activemedia Robotics

The PowerBot has a HitachiH8S-based microcontroller onboard, along with shaft encoders and sonar sensors to provide control. An onboard PC, laser scanner, camera and other optional extras are available. Drive is provided to two wheels, with casters added for stability. The robot comes with a price tag of US$25 K to $85 K depending on the options chosen.

Mobile robotic arm kits are available from Lynxmotion (http://www.lynxmotion.com). These are only miniature robots (the base is approximately 20 cm × 20 cm), and are not particularly suited to the applications envisioned for our cooperating robots.

In summary, although mobile platforms supporting an extendable manipulator arm are commercially available, they are generally prohibitively expensive. The affordable options are more hobbyist devices unable to support the sensors and processing power our application requires. The solution then, is to design and construct our own mobile platform and manipulator arm, at a low cost but not at the expense of functionality.

The remainder of this paper details the construction of a pair of mobile robots (christened "Itchy" and "Scratchy") that will be used to investigate autonomous cooperative behaviour. The robots will eventually be required to work together cooperatively to perform a single task, for example carrying a long piece of wood, although the actual cooperative architecture is outside the scope of this paper. A manipulator arm is currently being constructed and will be attached to the front of the robots above the two drive wheels.

2. Specifications

The design must consider the following attributes:

- **Environment** - The robots are required to operate in an outdoor terrain with a relatively smooth and level surface, such as concrete or mown grass.
- **Payload** – Each robot must be able to support all onboard components and an additional external payload of at least 40 kg
- **Manoeuvrability** – Each unit must be sufficiently agile to allow control in limited space. Precise control of the robot's path will be required
- **Self sufficient** – All required power and computation should be onboard the robot to facilitate independent, autonomous operation
- **Operating time** – Each robot must be capable of operating continuously for one hour
- **Communication** – The robots must have a communication link to allow them to transfer data, instructions or intentions
- **Scope** – The design must support future development, providing the computational capabilities, power supply and space to include additional sensors and actuators (particularly the manipulator arm)

3. Locomotion

A number of different drive systems are applicable to this project, and are reviewed in Carnegie et al., (2004). It is anticipated that the units will primarily be used outside on mostly flat surfaces. This avoids the complexity of legged, or self-laying track systems, and permits the implementation of a wheeled bicycle, tricycle or quadcycle locomotion.

A tricycle design was selected as it meets the requirements of:

1) Providing a simple design. Suspension or coordinated steering systems are not required when constructing a simple tricycle.
2) Minimising cost by reducing components required.
3) Being capable of adequately traversing a smooth outdoor terrain.
4) Offering manoeuvrability. It cannot perform point turns; however driving with the steering wheel at the rear will increase manoeuvrability.

5) Good stability. By locating heavy components near the bottom of the design and keeping the vehicle's height to a minimum, the centre-of-mass is lowered, reducing the chance of tipping. This is especially important when the manipulator arm is attached.

The methodology used to design these robots is also detailed in Carnegie et al. (2004).

4. Drive System and Motor Selection

With the chosen tricycle design, drive will be applied to the fixed pair of wheels to reduce the amount of slippage that would occur if only the steering wheel was driven. When driving both wheels, an allowance must be made for turning tight corners, since the inner wheel will travel a shorter distance than the outer wheel. Two solutions are available to this problem:

1) Drive each wheel independently using two separate motors.
2) Drive both wheels from a single motor through a differential gear, as used in automobiles.

The first solution requires an additional motor, gearbox and control circuitry. An advantage offered is that by increasing the power to one drive wheel, the robot will experience a rotational force which can assist steering during tight turns in a similar way that a wheelchair configuration steers.

The second solution only uses one motor and one reduction gearbox. This increases efficiency while reducing the required control sophistication and circuitry, though mechanical complexity is slightly increased by adding a differential gearbox. This was the method chosen.

Rather than purchase precision motors and gearboxes for propulsion, two motors (one for each mechatron), differential gears and axles were acquired from mobility scooters. The motors are 24 V dc, 400 W, with an electro-mechanical brake, and can power a 100 kg payload. Experimentally, it was found that these motors draw 3 A unloaded and 10 A when heavily loaded.

A steering motor is required to change the orientation of the single wheel over an expected angle range of ±60°. This motor must have sufficient torque to be able to turn the wheel and hold it in position, estimated to be approximately 10 Nm. A 24 V dc motor is preferred as this could be powered from the same voltage rail as the drive motors. Rather than use stepper motors or servomotors (which for the required torque and voltage values tend to be expensive), the decision was made to use a permanent magnet dc motor and gearing from a truck windscreen wiper. The motor shaft comprises a helical worm gear, which in turn drives a spur gear to give a 99:1 gearing ratio. To control the orientation of the steering wheel, a position sensor (in this case a potentiometer) is attached to the motor to provide feedback to a proportional-integral-derivative (PID) controller. The large reduction ratio provides extremely high torque, and makes position movement easier to control as the output shaft moves rather slowly. The peak current (loaded) drawn by this motor is approximately 6 A, whilst unloaded the current demand is 2 A.

5. Electronics

As these robots will be used to investigate cooperative robotic interaction, substantial processing power needs to be incorporated. Distributed embedded controllers can generally not run high level software packages such as MATLAB or LabVIEW, and a similar argument rules out Handheld PCs or Palm devices. The robots are reasonably large and can accommodate a full sized PC board. This option was chosen over the incorporation of a Laptop PC due to the cost saving advantages. Additionally, the full-sized PC board allows hardware interface through serial, parallel, USB (universal serial bus), and PCI (peripheral component interconnect) connections.

The specifications of the selected system (chosen as a compromise between power and cost) are:

- 2.6 GHz Celeron
- 512 MB RAM
- 100 GB Hard drive
- Windows XP Professional

Communication between the robots and also to a base station is achieved using Netgear 401 2.4 GHz wireless network cards, using the 802.11b protocol. Transfer speeds are up to 11 Mbit/s with a strong signal.

A National Instruments PCI-6025E card provides data I/O to the CPU from the robot's sensors and actuators. The DAQ card provides 32 digital input/output lines, 16 analogue input lines, 2 analogue output lines and 2 counters/timers.

Global positioning system (GPS) positioning is achieved using Motorola M12 Oncore receivers. The GPS receiver is a 12 parallel channel receiver, capable of tracking 12 satellites at once. A position is reported once per second by a serial data transmission.

Shaft encoders provide position information from the main drive wheels. The encoders are HEDS-5701 panel mount optical encoders, providing a 500 count per revolution quadrature output, and are driven by a pulley and belt arrangement from the drive axle. Noting that the inflated tyre diameter of the robots is 250 mm, each odometer count corresponds to a linear distance travelled of 1.247 mm.

Six Sharp GP2Y0A02YK infrared (IR) object detectors provide a distance measurement to objects within a 20 - 150 cm range, with a stated accuracy of ±15 cm, arranged as indicated in Figure 5. By characterising and filtering each individual detector this accuracy was increased to ±5 cm (for indoor use). The detectors provide an analogue 0.25 – 2.85 V dc output corresponding to the distance measured, updating every 32 ms.

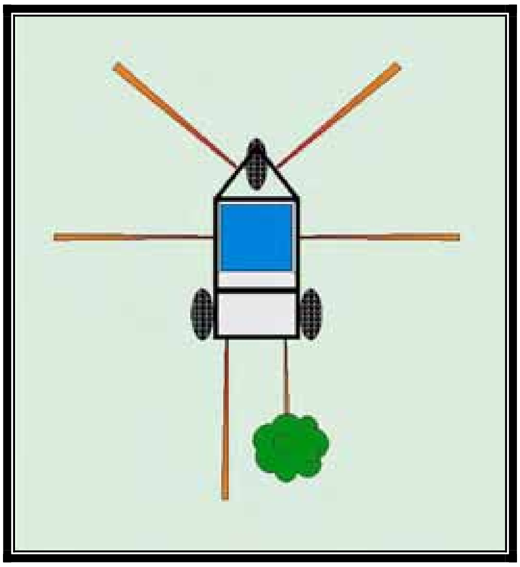


Figure 5. Infrared obstacle detection configuration

The detectors use a triangulation measuring method (rather than reflected intensity) that produces minimal variation in detected distance for objects of different colours or textures. The "forklift" style steering implies that the rear of the robot will swing outwards during a turn while travelling in forward direction. The two rear detectors are angled to monitor the area that the wheel will pass over during steering, and to also monitor behind the robot while reversing. This configuration however, does have a blind spot directly behind the mechatron – large objects will be detected by the angled sensors but smaller items may not be seen.

In most cases the robot will have travelled in a forward direction across the area of concern before any reversing manoeuvre is performed, allowing prior knowledge of the landscape to determine if the intended path intersects with obstacles.

To prevent the infrared light from adjacent emitters beating (due to cross talk resulting from off-axis emission), the two front emitters have each been mounted on a 1° angle away from the centre, making the intensity in any intersecting area negligible.

6. Power Requirements

A suitable power supply is required to run the PC. The power supply must offer the following rails (typical current rating shown for each rail):

- +12 V @ 7 A
- +5 V @ 25 A
- +3.3 V @ 8 A
- -5 V @ 0.5 A
- -12 V @ 1 A
- +5 V Standby @ 0.75 A

Previously the Mechatronics Group have used an uninterruptible power supply (UPS) (Cordes & Carnegie, 2002) to convert the 24 V battery supply into 230 V 50 Hz ac. This directly feeds into a standard ATX power supply of a desktop PC. This is a quick, easy solution but has the following major disadvantages:

- Efficiency is low converting 24 V dc to 230 V ac. and then back to 12 V dc and the other required voltages.
- The size of the UPS is large.
- The UPS weighs 10.9 kg.
- To turn the UPS on it must be connected to 230 V mains supply. For outdoor use, this means that the unit must be started in vicinity of a 230 V supply and remain in a standby condition until power up is required.

A better solution is to use an industrial power supply designed for the input voltage available, however these are only available from a small number of manufacturers and suppliers. An ACE-828C industrial ATX power supply was purchased, offering an 18 – 32 V dc input range. It has the same dimensions as a standard 230 V ATX supply.

A comparison of efficiency between using a UPS/230 V ac supply and the 24 V dc supply is shown in Table 1. The current shown is the amount drawn from the 24 V batteries by each power supply option while the CPU performs a defined task. The maximum values listed are the highest peak currents observed, and the "average working" values are the average currents drawn while performing a scandisk operation on the hard drive. The idle setting is the state where the power is on, but no processing or disk access is operating, and the standby setting corresponds to the state where the CPU is turned off, with only the +5 V standby rail active on the motherboard.

Using this ATX supply rather than a UPS unit results in a 30% improvement in efficiency for the robots under their normal operating conditions (not taking into account the weight savings from utilising the lighter supply).

Current usage (as drawn from the 24 V batteries) is estimated as 2 A continuous, 2.5 A peak for the PC board, 10 A continuous, 17 A peak for the drive motor, 2 A continuous, 6 A peak for the steering motor and an additional 1 A continuous, 2 A peak for the other electronics. The power supply then, has to deliver 15 A continually for one hour, with the ability to supply peak currents up to 27.5 A.

Considering the range of batteries available (Carnegie et al., 2004), two E360C Champion flooded lead acid (FLA) automotive batteries were selected for each robot. These batteries have a cold cranking amps (CCA) rating of 360 A, and a reserve capacity (RC) of 60 minutes. As the RC of a battery is defined as its ability to supply a constant load of 25 amperes at 25 °C without the voltage falling below 10.5 volts (for a 12 volt battery), over 90 minutes of continuous operation should be expected from this source. However, these are not deep cycle batteries, and their life will be greatly extended if run-times are restricted to a maximum of one hour.

	UPS/230 V ac	**24 V dc**	**Increase in Efficiency**
Maximum	3.17A	2.20A	30%
Average Working	2.89A	2.06A	29%
Idle	2.21A	1.41A	36%
Standby	0.79A	0.18A	77%

Table1. Comparison of UPS versus dc power supply current consumption

7. Chassis

The chassis design is constructed from 16 mm box section, 20 mm and 25.4 mm right angle steel to provide a strong frame on which to mount the robot's components.

Designed with the driving wheels at the front, the motor is positioned above the drive axle. This increases tyre traction by applying the weight of the motor onto the two drive wheels. The differential gearbox, which is part of the axle, limits the overall ground clearance of the mechatron to 65 mm. The remainder of the chassis has a ground clearance of 100 mm to prevent the mechatron from bottoming out on uneven terrain.

The batteries provide a significant proportion of the overall weight of the mechatron. Maintaining the centre of gravity within the area defined by the wheels requires that the battery placement be in the centre of the robot. However, the batteries will be used to counterpoise the weight of the manipulator arm (to be installed at a future date) and any load it may carry, and therefore is placed near the rear of the design. This placement, combined with the motor above the front axle, distributes the weight over the entire mechatron, maximising stability.

The PC case has been placed in the centre of the design due to the considerable area it requires. Minimal access to the PC is required as all work can be performed remotely using the wireless network.

The steering wheel is at the rear of the mechatron, with the motor mounted above it. The output from the motor gearbox is used to directly change the steering angle of the wheel.

The electronic printed circuit boards (PCBs) are supported by an acrylic (non-conductive) tray mounted at the top of the mechatron. This permits easy access to the circuits during development, and also allows the addition of new circuits as further sensors and actuators are added. A rectangular steel frame is mounted above the PCB area to protect the circuits from being knocked during transportation. Figure 6 shows the completed chassis frame.

The battery supports are made from 25.4 mm right-angle steel which hold the base of each battery along the front and rear edges. With an area of 125 mm × 370 mm, the batteries (each 125 mm × 180 mm), fit in the supports with little movement.

The PC is mounted in the same manner as the batteries, using right-angle steel across the front and rear of the base. Both the batteries and PC can be removed through the side of the chassis rather than out through the top to allow easier access.

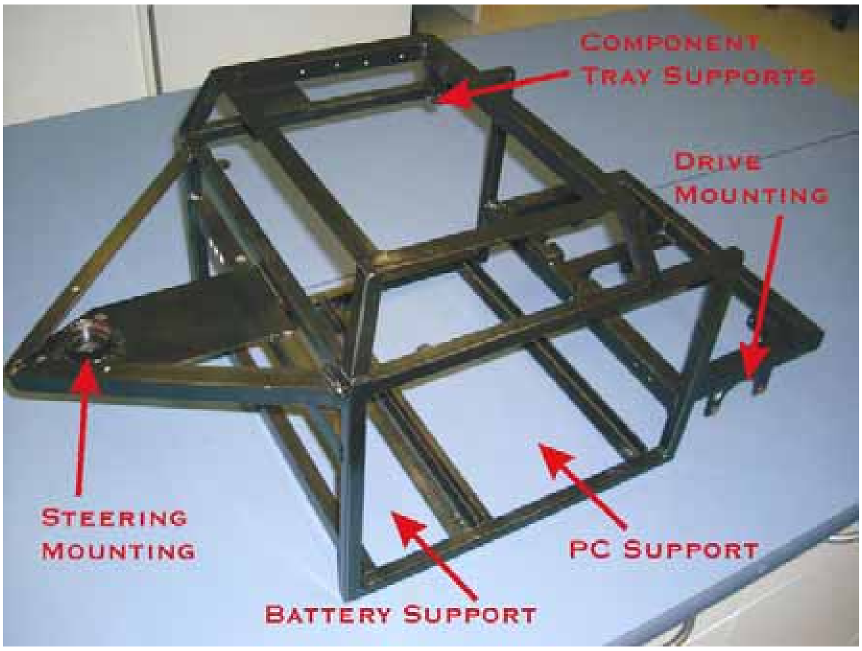


Figure 6. Chassis layout

8. Drive and Steering System

The drive system consists of a motor that drives an internal reduction gearbox (20:1 ratio) to a chain. The chain transfers the power to a differential gearbox (at a 1:1 ratio), which distributes the motion to the two ends of the axle. The assembled drive system is shown in Figure 7.

A 50 mm × 60 mm flat steel plate is welded 20 mm from the top of the mechatron, inside the PCB area to support the GPS antenna. This provides an unobstructed view of the sky for the antenna which uses a magnetic backing to secure it into place. The receiver can be mounted directly underneath the antenna in this configuration, reducing the area required for the complete GPS unit.The completed robots' dimensions are 865 mm × 600 mm × 430 mm (l × w × h), and they each weigh 58 kg (with onboard PC and batteries). The rear steering wheel supports 24 kg and the driving wheels combined support 34 kg, with the location of the centre-of-gravity shown in Figure 8. This is within the triangle generated between the wheels, and offers good stability. The complete assembled chassis with the drive systems constructed is shown in Figure 9.

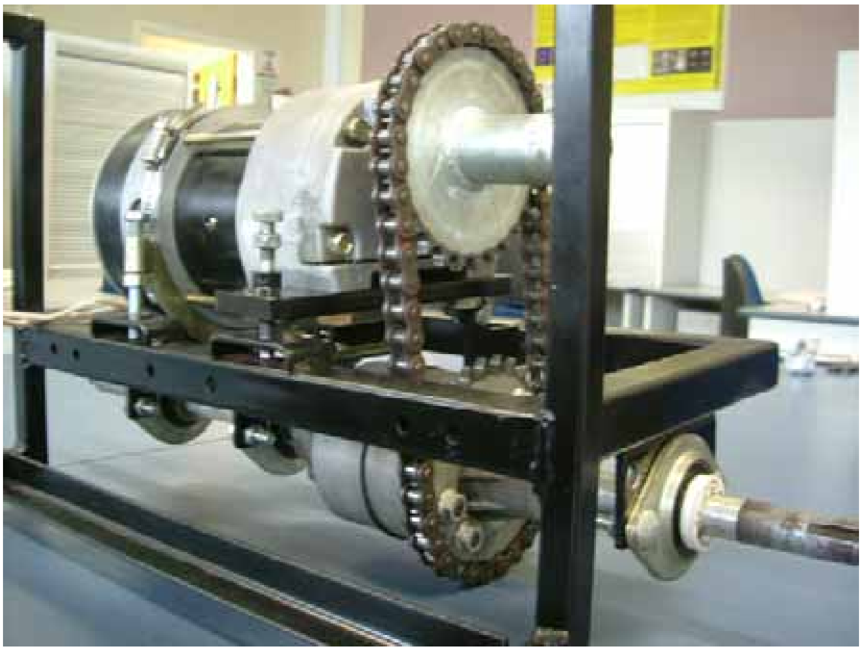


Figure 7. Mounted drive system showing chain tensioning method

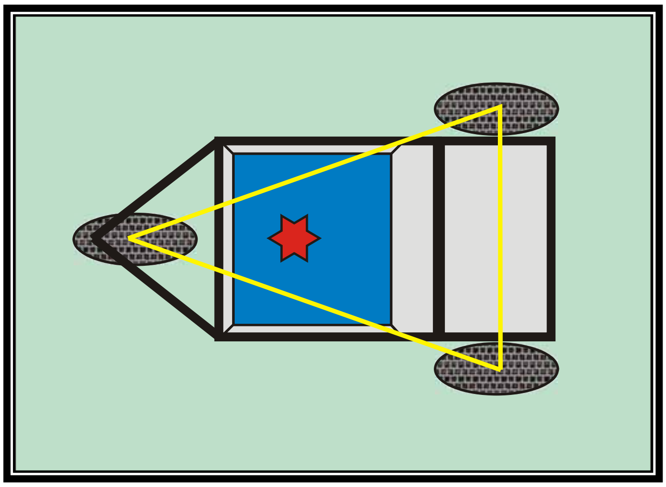


Figure 8. Centre of gravity of the loaded mechatron indicated by the star

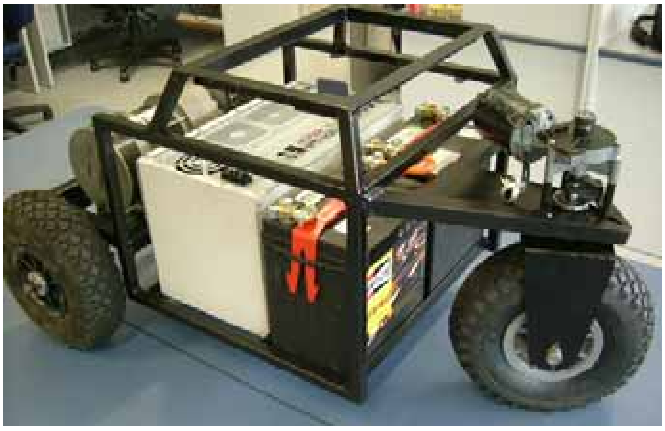


Figure 9. Complete chassis and drive components

9. Motor Control

The robots' drive motors are powered via a full H-bridge arrangement using a pulse width modulation (PWM) signal generated by an embedded Philips P89C51RC2 microcontroller in response to a command output from the DAQ card (Payne & Carnegie, 2003).

The modified wiper motor used to orientate the steering wheel, requires both a control system, and a power drive system. The desired steering wheel position (derived from a software PID algorithm) is input to a UC3637 IC using an analogue signal between 0 – 10 V (matching the Lab-PC+ analogue output range). This voltage is compared to the voltage from the potentiometer mounted on the motor drive shaft. The error is amplified, and converted to two 0 – 24 V 30 kHz PWM signals used to control the motor's speed and direction.

The L298 H-Bridge motor driver IC powers the steering motor. The input drive signals require 0 - 5 V logic, so the output from the UC3637 is converted from 24 V to 4.9 V using zener diodes. The completed units (without manipulator arm) are illustrated in Figure 10.

10. Manual Operation

Eventually these robots will be programmed to undertake assigned tasks in a cooperative manner, independent of human intervention. For testing purposes, the devices are controlled from a remote laptop computer, via the wireless LAN connection, using a Joystick. Two different user interfaces have been designed: one for use by the base that is controlling the mechatrons, the other on the mechatron itself.

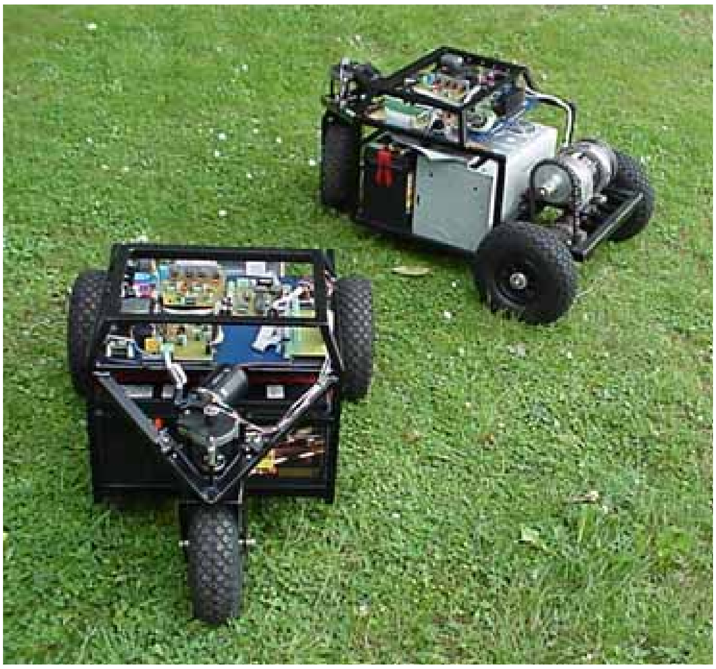
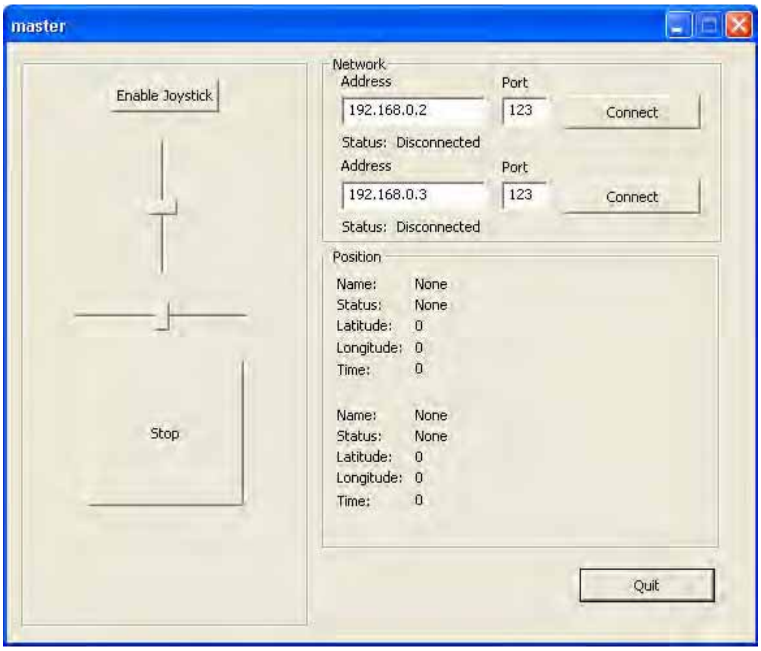


Figure 10. Completed mobile platforms, "itchy" and "scratchy"

The base provides a client connection to each mechatron. It allows control of the mechatron's speed and direction through the use of two sliders, and an emergency stop button is provided. The mechatron reports its current position, which is displayed on screen. Either a mouse or joystick (USB or analogue) can be used to control the sliders and hence the mechatron's movements. A screenshot of the base station is shown in Figure 11.

To assist development, all data received by the sensors are displayed locally on the mechatron. Any settings or variables can be viewed and altered with minimal effort. The left-hand half of the dialog box of Figure 12 provides control to the motors. If commands are being received through the network, the sliders will respond as the base sliders are moved. Otherwise the sliders can be moved by the user, providing a debugging interface. The progress bar next to each slider shows the mechatron's state, the current steering wheel angle, and the speed of the drive wheels. The stop button provides an emergency halt to both of the motors and can be used at any time.

The right-hand side of the dialog box is broken into a number of tabs, providing information on different parts of the system.



Network
Address: 192.168.0.2 Port: 123
Status: Disconnected
Address: 192.168.0.3 Port: 123
Status: Disconnected

Position

Name:	None
Status:	None
Latitude:	0
Longitude:	0
Time:	0
Name:	None
Status:	None
Latitude:	0
Longitude:	0
Time:	0

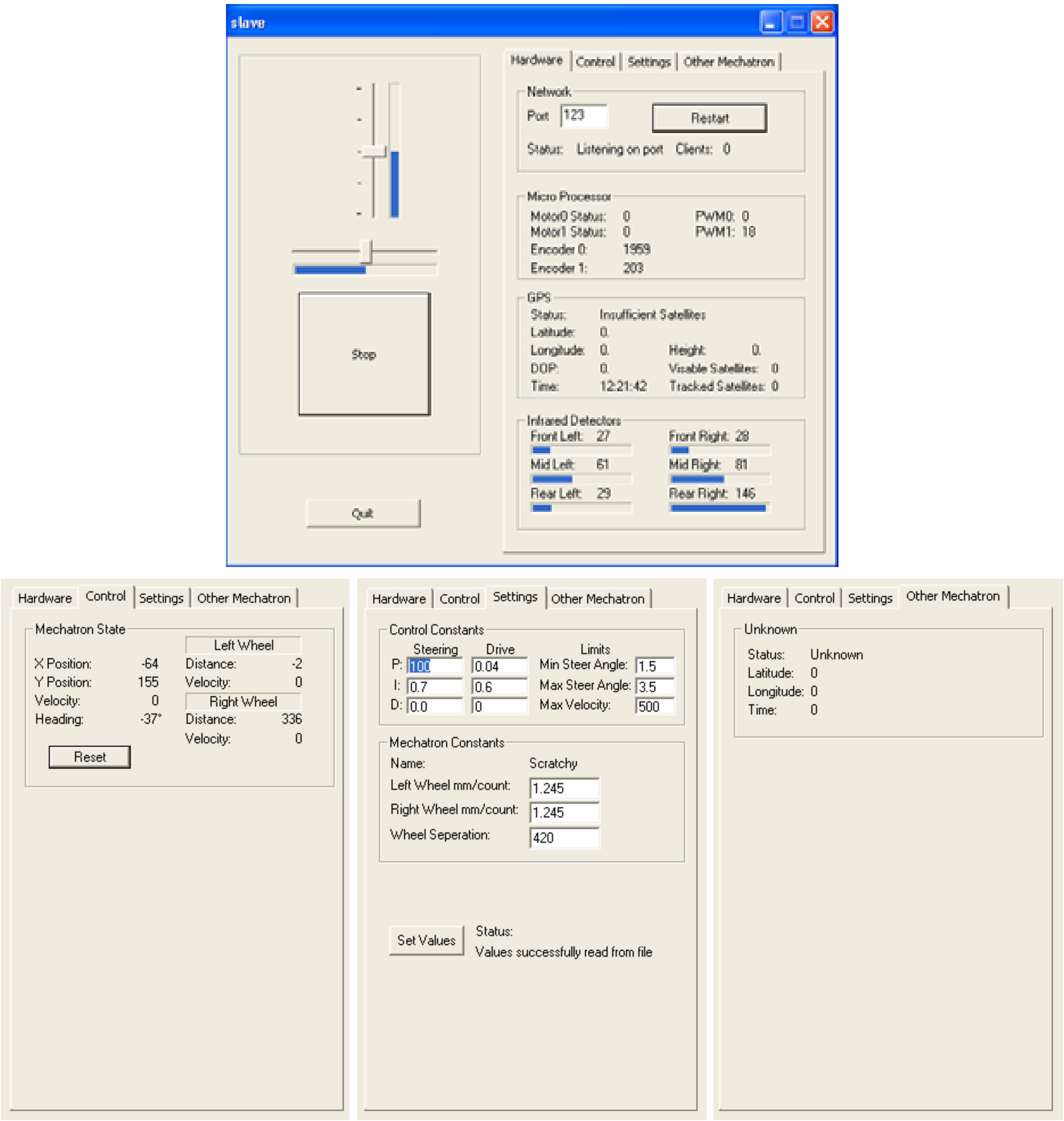
Figure 11. Base station user interface

The *hardware* tab provides network, microprocessor, GPS and IR object detector data. This tab is primarily used to display low-level data from the sensors and to indicate any problems. The status of the network port is shown, along with the number of clients connected. The microprocessor section shows the current PWM signals being sent from the PC, and the response (indicating any errors). The raw encoder values received are shown, again indicating any errors. The GPS shows the status of the receiver, the 3D position, the number of satellites available and the time. The time has been included to allow for logging of data on separate robots for later comparison. Using progress bars, the IR obstacle detectors show a graphical representation of the distance measured to any surrounding obstacles. A numerical representation is also shown with centimetre units.

The *control* tab is used to provide processed information to the user. The distance and speed of each drive wheel is shown, along with a calculated position and heading determined from this data. A large area of this tab has been provided for future development.

The *settings* tab permits on-the-fly alterations to the control system constants. This includes the PID constants for both the drive and steering systems, and physical constants

including the values used to convert encoder counts into a distance. These are provided to allow the user to "tweak" the values for each individual mechatron.
Each mechatron provides GPS data back to the base. This data is relayed so that each unit knows its own position as well as the position of others around it.



Network
Port: 123
Status: Listening on port Clients: 0

Micro Processor
Motor0 Status: 0 PWM0: 0
Motor1 Status: 0 PWM1: 18
Encoder 0: 1959
Encoder 1: 203

GPS
Status: Insufficient Satellites
Latitude: 0.
Longitude: 0. Height: 0.
DOP: 0. Visable Satellites: 0
Time: 12:21:42 Tracked Satellites: 0

Infrared Detectors
Front Left: 27 Front Right: 28
Mid Left: 61 Mid Right: 81
Rear Left: 29 Rear Right: 146

Mechatron State

		Left Wheel	
X Position:	-64	Distance:	-2
Y Position:	155	Velocity:	0
Velocity:	0	Right Wheel	
Heading:	-37°	Distance:	336
		Velocity:	0

Control Constants

	Steering	Drive	Limits	
P:	100	0.04	Min Steer Angle:	1.5
I:	0.7	0.6	Max Steer Angle:	3.5
D:	0.0	0	Max Velocity:	500

Mechatron Constants
Name: Scratchy
Left Wheel mm/count: 1.245
Right Wheel mm/count: 1.245
Wheel Seperation: 420

Status:
Values successfully read from file

Unknown
Status: Unknown
Latitude: 0
Longitude: 0
Time: 0

Figure 12. Rover user interface

11. Experimental Results

With the PWM control signal limited in software to a 75% duty cycle, the maximum velocity of the mechatron is 2 m/s. This limit is implemented to prevent damage to the mechatron and motor drive circuits if part of the system fails.
To determine the accuracy of both the GPS and dead-reckoning navigation techniques implemented, a mechatron was driven over a rectangular sports field of dimensions 55 m × 35 m four times.

The mechatron path started and ended at the same location so that any recorded data should form a closed loop when plotted.

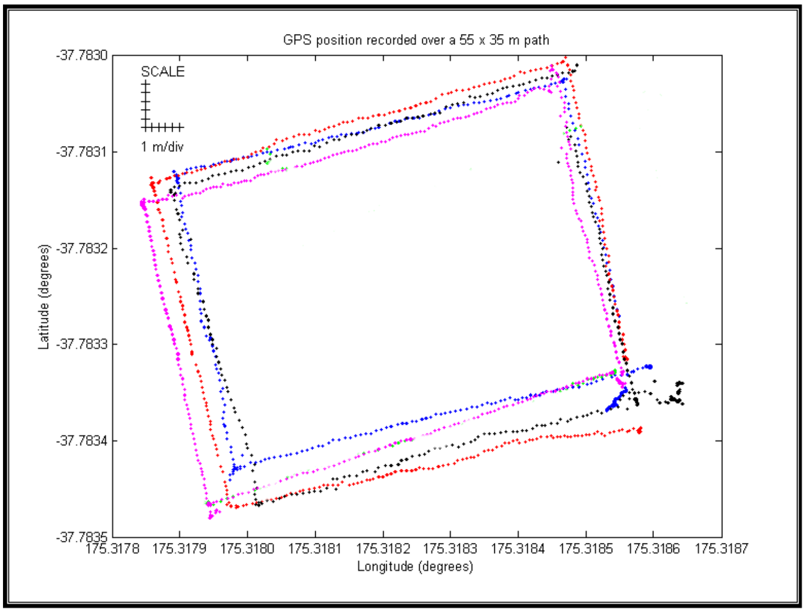


Figure 13. GPS reported position over a rectangular path

The reported position was logged to file every second, and the results of multiple runs illustrated in Figure 13. A scale is shown in the top left corner of the first plot, allowing a degree to metre conversion to be performed. The latitude and longitude axes are scaled independently.

Although each individual run plots a rectangular shape, the rectangle is not always closed as the start and end positions are not aligned (bottom right hand corner of each plot in Figure 13). The start-to-end measured distance (the same physical position) for the four runs was 7.81 m, 8.85 m, 6.33 m, and 0.94 m. This error is too large for the robots to operate in close vicinity of each other (<10 m) using a standard GPS position. A Vector V2X electronic compass has been used in another mobile robot constructed by the Mechatronics Group (Payne & Carnegie, 2003). This sensor has a stated operating resolution of 2°, which would produce a maximum lateral offset of 35 mm over a one metre distance. The use of this sensor approximately halves the start-to-end measurement error obtained from the GPS only trials.

To determine any increase in position accuracy when using differential GPS (DGPS) techniques, two M12 GPS receivers were used to report one position. Achieved by keeping the receivers stationary for a six hour period, the reported position was recorded over time with any variation due to GPS error. The test was repeated using pseudo-range corrections and also a block shift method (Payne & Carnegie, 2003).

Without corrections, standard deviations of 2.38 m and 4.64 m were recorded in the latitude and longitude directions respectively. Using a block-shift correction, the resulting standard deviation results were worse, 3.88 m and 6.90 m. However, the pseudo range correction improved the deviations to 1.68 m and 2.86 m. Whilst an improvement on the stand-alone GPS results, they are still not sufficient to permit accurate interaction between cooperating robots.

It is well known that odometry is subject to accumulating errors (Borenstein & Feng, 1996, Victorino & Borrelly, 2000). Using only odometry information for dead reckoning, the same test yielded final displacement errors of up to 50 m. There was a noticeable pull to the left due to uneven tyre inflation, and whilst this could be partially offset by the use of an on-board compass, it is obvious that the accumulating errors for this technique render it unsuitable for distances beyond 10 m, especially if the operating surface is uneven.

For reliable cooperative operation where relative distance accuracy needs to be of the order of 10 cm, further improvements must be made to the GPS system. The purchasing of carrier phase capable receivers, and/or dual frequency receivers could provide the accuracy required, but the cost is prohibitive for this system, being in excess of US$10,000. Regardless, some fusing of the GPS information, the odometers (potentially using landmarks to reduce the accumulating errors), the compass and additional sensors such as inertial sensors, or a laser ranger (identifying landmarks), will be necessary to advance the goal of cooperative robotic interaction.

The infrared object detectors did function reliably in the outdoor environment, although their operational range was affected by changes in the ambient sunlight (between 1.0 and 1.5 m).

12. Software Control

With the exception of the poorer than anticipated GPS resolution, the manual testing of the robots was successful. Whilst not the subject of this paper (which intends to concentrate on the mechanical and electronic design of the mobile robot), it is appropriate to provide some details on the anticipated software control of these mechatrons. A hierarchical approach to the design is considered and illustrated in Figure 14, having been successfully employed on three other mobile robots constructed by the Mechatronics Group (Carnegie, 2002).

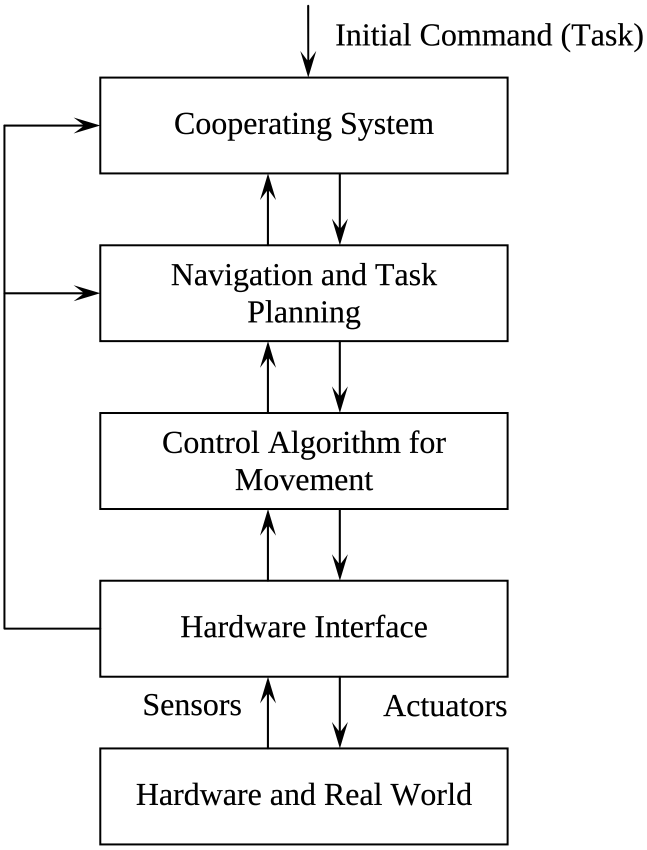


Figure 14. Software hierarchy

The addition to this hierarchy for Itchy and Scratchy is at the top level, the cooperating system which consists of a communications layer and a coordination layer. The communication layer will provide a means of awareness and knowledge sharing amongst the robots and will utilize the wireless LAN. The coordination layer will utilize the shared knowledge so that each robot takes into account the actions of the other robot in order to form a coherent group. This layer also processes the initial command (task) given to the robotic system by the human operator.

The navigation and task planning level involves path planning, i.e. moving in known or unknown environments as well as decomposing the task into subtasks and scheduling them. Below this is the control algorithm for movement which is responsible for controlling the robot's velocity and heading so that the task can be achieved (Lee-Johnson & Carnegie, 2003). The hardware interface provides a link between the hardware and software components and consists of device drivers such as drivers for the steering and drive motors. The lowest level of the architecture consists of the physical robot sensors and actuators, which we have interfaced to the upper level software through the data acquisition card described in section 5.

12.1 Co-Operative Robotic Control

Co-operative robotic behaviour independent of human intervention is a very active area of mobile robotic research. Ideally, a human should only provide the initial command to the robots that then decide for themselves how to execute the task. For the purposes of completeness, a brief review of cooperative robotic research is included here.

Three major applications of co-operative robotic behaviour are transportation, sensing and foraging. Cooperative transportation involves multiple robots transporting objects from one location to another and has been exhibited in robot soccer teams (Shim et al., 1997, and de la Rosa et al., 1997) as well as in box pushing (Mataric et al., 1995) and object lifting and carrying (Huntsberger et al., 2003). Cooperative sensing develops a group robotic system for map building and localization for navigation and exploration (Sossai et al., 1999 and Yamauchi, 1999). In foraging, groups of robots must pick up objects scattered in the environment (Parker, 1998).

The control or group architecture provides the infrastructure upon which collective behaviours are implemented and determines the capabilities and limitations of the system (Cao et al., 1997). Research in this area addresses issues such as action selection, delegation of authority and control, the communication structure, and heterogeneity versus homogeneity of robots (Arai et al., 2002). One of the key architectural features of a group architecture for mobile robots is whether the system is centralized or decentralized. Centralized architectures are characterized by a single control agent whereas decentralized architectures allow multiple control agents. The decentralized architecture has been the dominant group architecture since it has several inherent advantages over centralized structures. Two types of decentralized architectures include hierarchical architectures and distributed architectures (Cao et al., 1997).

Hierarchical architectures are locally centralized. The agents are independent in carrying out tasks to achieve certain goals but they communicate with a master or host that has a global view of operations and assigns goals to the agents. CEBOT, a hierarchical architecture which consists of a group robotic system that is dynamically reconfigurable, has been simulated (Fukuda & Iritani, 1995). The GOFER architecture which uses a central task planning and scheduling system was used to study distributed problem solving by multiple robots in an indoor environment using traditional AI techniques and was successfully used with three physical robots (Cao et al., 1997). Cooperative behaviour based on fuzzy logic optimized by micro genetic algorithms for fixed obstacle and multiple robot avoidance in a centrally managed robot system has been simulated in Glorennec (1997). Hierarchical architectures have also been implemented in co-operative robot soccer teams where soccer robots are linked to a host computer system but have distributed control. The host computer may be used as a coach for the team (de la Rosa et al., 1997) or in a more difficult implementation, as a sensor for processing vision data only (Shim et al., 1997). The use of two cooperative robots operating in a master/slave configuration to facilitate localization and mapping has also been studied (Sossai et al., 1999). Hierarchical architecture implementations, in general, require an additional computer system to act as the master or host. This means additional costs and also leads to the implementation not being robust in the event of this primary device failing.

Distributed architecture implementations remove the need for a master or host. In distributed architectures, all agents have equal control and hence are completely autonomous in the decisional process with respect to each other. A practical application based on a distributed architecture is map building for exploration in an unknown environment using two co-operative robots (Yamauchi, 1999). In this application the robots share perceptual information but maintain separate maps and make independent decisions which leads to the system being robust to the loss of communication between them as well as to the loss of a robot. A distributed system carrying out a box pushing task using explicit communication for coordination has been shown to perform more effectively than a single robot or two non-communicating robots (Mataric et al., 1995). A cooperative box pushing mission by two heterogeneous robots has been achieved using a fully distributed system at both the individual robot and team levels based on the ALLIANCE architecture. This architecture has also been implemented on a physical robot team performing a laboratory version of a hazardous waste cleanup (Parker, 1998). The ALLIANCE architecture has the advantage of using adaptive actions to achieve fault tolerant control within small to medium sized teams of heterogeneous robots. The ABBA architecture, which is designed for distributed cooperative planning, has been utilized in the implementation of a cooperative cleaning task with two autonomous mobile robots (Jung & Zelinsky, 1999). This implementation has also shown the advantage of robustness in the face of failures. CAMPOUT, a distributed control architecture for tightly coupled coordination of multiple robot systems is being developed at the Jet Propulsion Laboratory (Huntsberger et al., 2003). It has been applied to ongoing physical experiments involving the exploration of cliff faces and the deployment of extended payloads.

An approach to distributed coordination in a heterogeneous multiple robot system based on dynamic role assignment is described in Iocchi et al., (2003). The robots are heterogeneous in individual robot control architectures as well as in physical nature. A programming environment called ETHNOS is used to implement the different architectures as well as the communication layer. The distributed coordination protocol has shown robustness to communication failures.

Itchy and Scratchy will implement a decentralized distributed group architecture due to the advantages outlined above. Their initial task will be one of cooperative mapping of an unknown environment. At present the basic control of the individual robot movements are based on PID control laws. A behaviour-based architecture (reactive system) (Brooks, 1986) for the individual robots is being constructed for the coordination layer using neural networks and fuzzy logic. (Mataric et al., 1995, Huntsberger et al., 2003, Parker, 1998 and Jung & Zelinsky, 1999) have also used behaviour-based approaches.

13. Future Work

The reduction in the range of the infrared sensors in bright sunlight indicates that it is advisable to include additional obstacle avoidance sensors. Laser range finders are an attractive solution; however they tend to be expensive, requiring either high speed electronics or a dedicated digital camera system (often with an associated frame grabber card). A simple alternative is the incorporation of ultrasonic sensors. These can reliably project out to a distance of several meters, and detect obstacles that the robot is likely to encounter. By narrowing the transmitted beam, multipath reflections can be reduced, and simple time-of-flight calculations can easily yield the robot-to-obstacle distance.

Accurate localisation of the robots will need further development. The several meter accuracy of the GPS units is not adequate for fine positioning of the robots. Whilst infrareds and/or ultrasonics can provide accurate *relative* positioning once the robots are closer than 5 meters apart, it would be desirable to have sub-meter absolute positioning. This could be achieved with the purchasing of more expensive GPS modules (US$10K), this would negate the low-cost emphasis of this construction. Odometry and inertial sensing are accurate over short distances, so it is anticipated that use will be made of landmarks to reset the accumulated odometry error, and provide for more accurate localisation. Unfortunately, this constrains the robots to operating in a known environment, and is not an optimal solution.

Finally, the manipulator arm obviously needs to be designed and mounted on the robots. This is a significant task, and is not the focus of this article.

14. Conclusion

Two identical robots, "Itchy" and "Scratchy" have been constructed at low cost (below US$2000). They are mechanically and electronically complete, though awaiting the future implementation of a manipulator arm. Weighing 58 kg, they are capable of supporting an external payload of 80 kg, and can operate on their contained power supply for times in excess of 1 hour. The devices are very manoeuvrable, capable of turning within a 1.5 m enclosure.

The data from the shaft encoders provides a resolution of 1.245 mm, and accuracy within 1% on outdoor terrain. The heading calculated from the shaft encoder data alone is insufficient for localisation, but combined with the recently installed compass provides a more accurate dead reckoning navigation system. The GPS receiver is capable of providing an absolute position with standard deviations of 1.68 m and 2.86 m in the longitude and latitude directions when using pseudo-range corrections. For applications where the mechatrons are distant from one another, this can be used for initial localisation. The motors are controlled by a microcontroller interfaced to the PC.

The PC permits the use of advanced software tools, and communication amongst themselves or with a base station via a wireless LAN card. The operator can directly control the mechatrons from the onboard computer through a virtual desktop using Windows XP terminal server, or using the remote base software written. The remote software allows a joystick to be used on the local machine to ease control of moving the mechatron.

The hierarchical software system used successfully by the authors for other mobile robots is being extended to facilitate human independent co-operative interaction between Itchy and Scratchy.

Acknowledgements

The authors would like to express their gratitude to the Automation Systems Division of Industrial Research Ltd. (Parnell), and to Bruce Rhodes and Scott Forbes of the Department of Physics and Electronic Engineering at the University of Waikato.

15. References

Arai, T., Pagello, E., and Parker, L.E., "Editorial: Advances in multi-robot systems", *IEEE Transactions on Robotics and Automation*, vol. 18, no. 5, pp. 665-661, 2002.

Borenstein, J., and Feng, L., "Measurement and correction of systematic odometry errors in mobile Robots", *IEEE Transactions on Robotics and Automation*, 1996, **12**(6).

Brooks, R., "A robust layered control system for a mobile robot", *IEEE Journal of Robotics and Automation*, vol. 2, no. 1, pp. 14-23, 1986.

Cao, Y.U., Fukunaga, A.S., and Kahng, A.B., "Cooperative mobile robotics: antecedents and directions", *Autonomous Robots*, vol. 4, no. 1, pp. 7-27, 1997.

Carnegie, D.A., Towards a fleet of autonomous mobile mechatrons. In *Proceedings of the International Federation of Automatic Control Second Conference on Mechatronics Systems*, Berkeley, 9-11 December 2002, pp 673-678.

Carnegie, D., Loughnane, D.L., and Hurd S.A., "The design of a mobile autonomous robot for indoor security applications", *Proc. Instn Mech. Engrs Part B: J. Engineering Manufacture*, 2004, **218**(B5), 533 – 543.

Cordes, J.C., and Carnegie, D.A., The Creation of an autonomous multi-terrain mechatron. In *Proceedings of the Ninth New Zealand Electronics Conference*, Dunedin, New Zealand, 14 – 15 November 2002, pp 49 – 54

de la Rosa, J.L., Oller, A., Vehi, J., and Puyol, J., "Soccer team based on agent-oriented programming", *Robotics and Autonomous Systems*, vol. 21, no. 2, pp. 167-176, 1997.

Fukuda, T., and Iritani, G., "Construction mechanism of group behaviour with cooperation", in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Pittsburgh, USA, 1995, pp. 535-542.

Glorennec, P., "Coordination between autonomous robots", *International Journal of Approximate Reasoning*, vol. 17, no. 4, pp. 433-446, 1997.

Huntsberger, T., Pirjanian, P., Trebi-Ollennu, A., Nayar, H.D., Aghazarian, H., Ganino, A.J., Garrett, M., Joshi, S.S., and Schenker, P., "CAMPOUT: A control architecture for rightly coupled coordination of multirobot systems for planetary surface exploration", *IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans*, vol. 33, no. 5, pp. 550-559, 2003.

Iocchi, L., Nardi, D., Piaggio, M., and Sgorbissa, A., "Distributed coordination in heterogeneous multi-robot systems", *Autonomous Robots*, vol. 15, no. 2, pp. 155-168, 2003.

Jung, D., and Zelinsky, A., "An architecture for distributed cooperative planning in a behaviour-based multi-robot system", *Robotics and Autonomous Systems*, vol. 26, no. 2-3, pp. 149-174, 1999.

Lee-Johnson, C., and Carnegie, D., "The development of a control system for an autonomous mobile robot". In *Proceedings of the Tenth New Zealand Electronics Conference*, Hamilton, New Zealand, 1 -2 September, 2003, pp 77 – 82.

Payne, A.D., and Carnegie, D.A., "Design and construction of a pair of tricycle based robots to investigate cooperative robotic interaction", In *Proceedings of the Tenth*

Electronics, New Zealand Conference, Hamilton, New Zealand, 1 – 2 September 2003, pp 53 – 58

Mataric, M.J., Nilsson, M., and Simsarian, K.T., “Cooperative multi-robot box-pushing”, in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Pittsburgh, USA, 1995, pp. 556-561.

Parker, L.E., “ALLIANCE: An architecture for fault tolerant multirobot cooperation”, *IEEE Transactions on Robotics and Automation*, vol. 14, no. 2, pp. 220-240, 1998.

Shim, H.S., Kim, H.S., Jung, M.J., Choi, I.H., Kim, J.H., and Kim, J.O., “Designing distributed control architecture for cooperative multi-agent system and its real-time application to soccer robot”, *Robotics and Autonomous Systems*, vol. 21, no. 2, pp. 149-165, 1997.

Sossai, C., Bison, P., Chemello, G., and Trainito, G., “Sensor fusion for localization using possibility theory”, *Control Engineering Practice*, vol. 7, no. 6, pp. 773-782, 1999.

Victorino, A.C., Rivers, P., and Borrelly, J.J., Localization and map building using a sensor-based control strategy. In *Proceedings of the 2000 IEEE International Conference on Intelligent Robots and Systems*, Takamatsu, Japan, October 2000.

Yamauchi, B., “Decentralized coordination for multirobot exploration”, *Robotics and Autonomous Systems*, vol. 25, no. 2-3, pp. 111-118, 1999.

Cooperative Hunting by Multiple Mobile Robots Based on Local Interaction

Zhi-Qiang Cao, Min Tan, Saeid Nahavandi & Nong Gu

1. Introduction

The problem of multi-robot coordination and cooperation has drawn great interest in recent years [1]. Generally speaking, for a given task, utilizing more than one robot may enhance the quality of the solution. Furthermore, many inherently distributed tasks must require a distributed solution. However, if the robots are not properly organized, the interference among them will block the task. Many challenging issues should be considered carefully. In this paper, we conduct our research in the context of multi-robot hunting.

The hunting task concerning mobile robots and its target to be hunted is a particular challenge due to the nature of unknown and irregular motion of the target. In order to coordinate the motion of multiple mobile robots to capture or enclose a target, a novel feedback-control law [2], linear autonomous system [3] and Multiple Objective Behavior Coordination [4] have been used. Other related works including pursuit game [5]- [7], whose environments are usually modeled in grid. In this chapter, we choose non-grid environments where each robot with a limited visual field moves in any collision-free direction.

This chapter considers a typical hunting task where multiple mobile robots (pursuers) cooperatively hunt an invader with certain intelligence in unknown environments. After the invader is found, its position information is broadcasted. If each robot knows where other robots are, it may take action from the group’s perspective. However, this will generate a communication burden that is too heavy to be practical. An approach where no positions of the robots are exchanged must be pursued. With less information, what action should the individual robot take? In this paper, an approach called *Cooperative Local Interaction* (CLI) is proposed to achieve cooperative hunting.

This chapter is organized as follows. Section 2 describes CLI approach where the hunting task is divided into four states and the detailed design in each state is given. In section III, the motion strategies for the invader are designed. Simulations are conducted in section IV and section V concludes the chapter.

2. CLI Approach

Assume that a multi-robot team is used to perform a hunting task. We label each robot $R_i (i = 1,2,..., N) \in \aleph$. In order to avoid possible collisions, a range sensor model S_{range} is used to perceive the environment. We assume that the robots and the invader can see each other

with the same range as that of s_{range}. Each robot may recognize other teammates and also can recognize the invader.
The task is divided into four states, which are as follows:

search state — each robot will search the environment until the invader is found.
pursue state — when a robot knows the position of the invader either by perception or by communication and it thinks that the occasion to catch the invader hasn't come yet, it keeps pursuing the invader.
catch state — when a robot thinks that the condition to catch the invader has been met, it will catch the invader within a period of time.
predict state — after a robot loses track of the invader, it will predict the invader within a period of time.

The central idea of CLI approach can be stated as follows: each robot determines its current state, makes decisions based on the information of the invader and other robots near it. The task is executed through local interaction among the robots. In the following, the detailed design in each state is introduced.

2.1 Search State

The robot endeavours to find the invader in the search state. The individual robot keeps away from other robots to utilize the resources better when it sees them (Boolean variable b1=1), and in other cases (b1=0) it moves randomly. Regardless of initial distribution of robots and the environment's shape, an effective search can be accomplished. The robot's moving direction $(x_r, y_r)^T$ is as follows.

$$\begin{bmatrix} x_r \\ y_r \end{bmatrix} = \begin{cases} \begin{bmatrix} p_{cx} - p_{nex} \\ p_{cy} - p_{ney} \end{bmatrix} / \sqrt{(p_{cx} - p_{nex})^2 + (p_{cy} - p_{ney})^2} & b_1 = 1 \\ \begin{bmatrix} \cos(-\tau \cdot \text{sig}\rho) & -\sin(-\tau \cdot \text{sig}\rho) \\ \sin(-\tau \cdot \text{sig}\rho) & \cos(-\tau \cdot \text{sig}\rho) \end{bmatrix} \cdot \begin{bmatrix} x_d \\ y_d \end{bmatrix} & b_1 = 0 \end{cases} \tag{1}$$

where $(p_{cx}, p_{cy})^T$, $(p_{nex}, p_{ney})^T$ are coordinates of the robot and another one nearest to it, respectively; $(x_d, y_d)^T$ refers to the robot's heading; τ is an angle randomly rotated; $\rho(\rho \in [0,1])$ is a random number and $\text{sig}\rho = \begin{cases} -1 & 0.5 > \rho \geq 0 \\ 1 & 1 \geq \rho \geq 0.5 \end{cases}$.

2.2 Pursue State

Let P_T, C_T denote the current position of the invader and its center, respectively. $\wp(\chi_1, \chi_2] = \left\{ P \middle| \chi_1 < \left| \overrightarrow{PP_T} \right| \leq \chi_2 \right\}$ is defined $(\chi_1, \chi_2]$ zone around the invader, that is, the distance from position $P(P \in \wp(\chi_1, \chi_2])$ to the invader is within $(\chi_1, \chi_2]$. For robot $R_q \in \aleph$, when its position $P_r(q) \in \wp(\chi_1, \chi_2]$, the robot is called a $(\chi_1, \chi_2]$ robot.
Once a robot acquires the information of the invader, according to the distance to the invader, three zones can be generated and they are $\wp(0, x_{near}]$, $\wp(x_{near}, x_{far}]$ and $\wp(\chi_{far}, +\infty)$, where χ_{near} and χ_{far} ($x_{far} > x_{near}$) both are parameters determined by the robot's maximum sensing range and its radius. Only when a robot is a $(0, x_{near}]$ robot is it meaningful to judge whether the condition to catch the invader is satisfied.

For a $(0, x_{near}]$ robot R_i whose position is expressed by $P_r(i)$, let $P_r^i(m_{nb})(m_{nb} = 1,2,...,N_i)(N_i \geq 1)$ denote the positions of all $(0, x_{near}]$ robots near it and itself. The robots involved consist of a set $\aleph_i$. The process judging whether the condition to catch the invader is satisfied is described in the following.

Step_1: if $N_i \geq 3$, the robot chooses a m_{nb}.

Step_2: when N_i is an even number, the robot obtains $P_r^i(\min_{m_{nb}})$, which is $P_r^i(m)$ that makes the angle between $\overrightarrow{P_T p_r^i(m_{nb})}$ and $\overrightarrow{P_T P_r^i(m)}(m = 1,2,...,N_i, m \neq m_{nb})$ minimal and establishes a pole coordinate system $\Sigma_{eveni}^{m_{nb}}$ whose pole, polar axis direction are C_T and $\frac{\overrightarrow{P_T p_r^i(m_{nb})} + \overrightarrow{P_T P_r^i(\min_{m_{nb}})}}{2}$, respectively. If N_i is an odd number, the robot establishes a polar coordinate system $\Sigma_{oddi}^{m_{nb}}$ whose pole is C_T with the polar axis direction of $\overrightarrow{P_T P_r^i(m_{nb})}$.

Step_3: calculate the coordinates $P_s^{m_{nb}i}(\lambda_s, \gamma_s)(s = 1,2,.., N_i)$ in $\Sigma_{eveni}^{m_{nb}}$ (or $\Sigma_{oddi}^{m_{nb}}$) for all robots in $\aleph_i$, where $\gamma_s \in [0, 2\pi)$.

Step_4: When $\exists m_{nb}$ such that $\exists \gamma_k \in \left[\frac{\pi}{2}, \pi\right) \cap \exists \gamma_j \in \left[\pi, \frac{3\pi}{2}\right)(k, j = 1,2,..., N_i)$, the condition to catch the invader is considered to be satisfied.

When a robot thinks that the condition to catch the invader is not satisfied, it is in the pursue state and has to pursue the invader first. Considering any robot R_i, the local decision-making process is as follows.

Step_1: acquire the positions of all other robots near R_i $P_{rn}^i(m_n)(m_n = 1,2,...N_n)$, where N_n may be null and it means there is not a robot near R_i (go to Step_6). If $N_n > 0$, it establishes a polar coordinate system Σ_{ri} whose pole, polar axis direction are C_T and $\overrightarrow{P_T P_r(i)}$, respectively.

Step_2: calculate the coordinates $P_l^{neari}(\mu_l, \varphi_l)(l = 1,..., N_n)$ for all neighboring robots in Σ_{ri}, where $\varphi_l \in (-\pi, \pi]$.

Step_3: get the angle φ_{min} that is the minimum of $|\varphi_l|$ and the corresponding robot's position $P_{rn}^i(\min)$.

Step_4: if φ_{min} is less than ς, R_i should endeavour to reduce or eliminate the interference with other robots (go to Step_5), otherwise, there is no need to consider other robots (go to Step_6), where $\varsigma = \frac{2\pi}{3}(N_n \leq 2)$ or $\varsigma = \frac{2\pi}{N_n + 1}(N_n > 2)$.

Step_5: acquire the coordinate $P_{min}^i(\varpi_i, \phi_i)$ for R_i in a polar coordinate system Σ_{min}^i whose pole and polar axis direction are C_T and $\overrightarrow{P_T P_{rn}^i(\min)}$, respectively. Naturally, we have $|\phi_i| = \varphi_{min}$. If $\phi_i > 0$, the coordinate of the robot's ideal position in Σ_{min}^i should be $P_d^i(r_{dis}, \varsigma)$, otherwise, it should be $P_d^i(r_{dis}, -\varsigma)$, where r_{dis} is χ_{far} (when R_i is a $(\chi_{far}, +\infty)$ robot) or $\chi_{near} - 2\Delta$ (in other cases). Δ is a margin.

Step_6: the ideal motion position $P_r^d(i)$ for R_i should locate in $\overrightarrow{P_T P_r(i)}$ and $\left|\overrightarrow{P_T P_r^d(i)}\right| = r_{dis}$.

2.3 Catch State

For the robot R_i, once the condition to catch the invader is satisfied, it is in the catch state and will catch the invader by moving to the ideal position $P_r^{cd}(i)$, which locate in $\overrightarrow{P_T P_r(i)}$ and $\left|\overrightarrow{P_T P_r^{cd}(i)}\right| = r_c$ $(r_c < \chi_{near} - 2\Delta)$, where r_c is so small that the invader cannot move any more when most of the robots involved arrive at their ideal positions. The decision-making process is adopted within certain steps $\mathrm{step}_{\mathrm{catch}}$ without considering the condition to catch the invader. If the step number the robot moves reaches $\mathrm{step}_{\mathrm{catch}}$, it means that the invader maybe breaks away from being caught. In this case, the relevant robots need to re-analyze the situation.

2.4 Predict State

Because of the complexity of the task and its environment, the robots possibly lose track of the invader. For R_i, when it loses track of the invader, it is in the predict state and has to predict the invader within certain steps $\mathrm{step}_{\mathrm{predict}}$ based on previous invader position P_{pre}^{T} and its motion information to find the invader again. We denote with P_{pre}^{i} the previous position of R_i. If the invader escapes along $\overrightarrow{P_{\mathrm{pre}}^{i}P_{\mathrm{pre}}^{T}}$ and moves in the maximum step size of R_i, the suppositional escaping position of the invader P_{sup}^{T} can be calculated. Thus the robot hopes to move along $\overrightarrow{P_r(i)P_{\mathrm{sup}}^{T}}$. If the step number R_i in the predict state moves reaches $\mathrm{step}_{\mathrm{predict}}$, it shows that it is difficult to find the invader by prediction. This requires R_i to re-search the environment.

2.5 Motion Strategy

The above decision-making of each state generates an output (a moving direction or a motion position) without considering the obstacles. Therefore, an effective motion strategy is indispensable to make each robot move effectively and safely. It combines the corresponding output with readings from sensors to control the robot.

The strategy proposed here may enable each robot to obtain its safe moving direction that has the least angle with its ideal direction on the premise of predetermined step size.

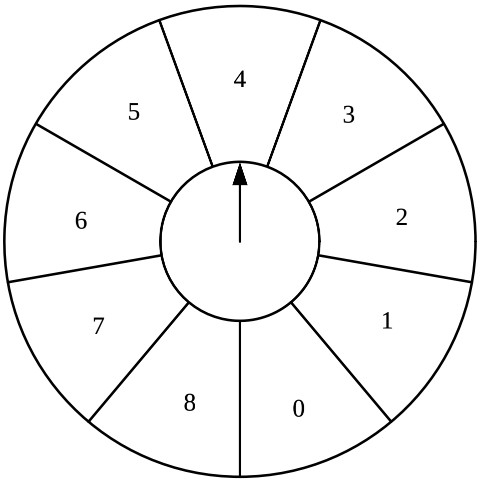


Figure 1. The layout of sensors

The robot adopts a range sensor model S_{range} to perceive the environment. The detecting zone of each sensor is a sector. Fig. 1 shows the layout of sensors whose numbers are assigned from 0 to 8 as starting from the reverse direction of the robot's heading, which is shown in arrow. The robot can know the presence or absence of other objects in each sector as well as the distance to them. For any robot R_i, it establishes a polar coordinate system Σ_{Ri} whose pole and polar axis direction are its center and current moving direction, respectively. Denote $P_{rr}(\rho_r,\theta_r)$ as the coordinates of the ideal motion position of R_i in Σ_{Ri}.

The coordinates of the detecting border of sensors $\mathrm{S_t}(\mathrm{t}=0,1,\ldots,8)$ in Σ_{Ri} are $\mathrm{P_s^t}(\rho_\mathrm{t},\theta_\mathrm{t})$, where ρ_t is the maximum sensing range when no obstacle is detected, otherwise, reading from $\mathrm{S_t}$ after the invader is considered, and $\theta_\mathrm{t}\in\left[-\pi+\frac{2\pi}{9}\mathrm{t},-\pi+\frac{2\pi}{9}(\mathrm{t}+1)\right]$. We denote with $\mathrm{P_a}(\rho_\mathrm{a},\theta)$ the coordinates of next position of $\mathrm{R_i}$ in Σ_{Ri}, where ρ_a is the step size determined by the robot's current position, ideal position and maximum step size; θ is the angle it rotated. The goal is to seek θ within $[-\zeta_{\mathrm{r\,max}},\zeta_{\mathrm{r\,max}}]$ of current moving direction on the premise of predetermined ρ_a such that the robot moves along the collision-free direction that has the least angle with the ideal direction. Base on sensory information, the distances $\mathrm{P_aP_s^t}(\mathrm{t}=0,1,\ldots,8)$ from $\mathrm{P_a}$ to the detecting border of each sensor should be greater than or equal to a safety distance $\mathrm{D_{safe}}$ determined by the robot's velocity and its radius, namely,

$$\mathrm{P_aP_s}^{\mathrm{t}}\geq \mathrm{D_{safe}}(\mathrm{t}=0,1,\ldots,8) \tag{2}$$

The final value of θ should satisfy eq. (2) and make $|\theta-\theta_\mathrm{r}|$ a minimum. Considering the t^{th} sensor, we have

$$\sqrt{(\rho_\mathrm{a}\cos\theta(\mathrm{t})-\rho_\mathrm{t}\cos\theta_\mathrm{t})^2+(\rho_\mathrm{a}\sin\theta(\mathrm{t})-\rho_\mathrm{t}\sin\theta_\mathrm{t})^2}\geq \mathrm{D_{safe}} \tag{3}$$

where $\theta(\mathrm{t})$ are the values of θ satisfying the condition of the t^{th} sensor in eq. (2). From eq. (3), it can be obtained that

$$\cos(\theta(\mathrm{t})-\theta_\mathrm{t})\leq\frac{\rho_\mathrm{a}{}^2+\rho_\mathrm{t}{}^2-\mathrm{D_{safe}}{}^2}{2\rho_\mathrm{a}\rho_\mathrm{t}}=\mathrm{D} \tag{4}$$

When $|\rho_\mathrm{a}-\rho_\mathrm{t}|\geq \mathrm{D_{safe}}$ is satisfied, $\theta(\mathrm{t})\in[-\zeta_{\mathrm{r\,max}},\zeta_{\mathrm{r\,max}}]$.
When $\rho_\mathrm{a}+\rho_\mathrm{t}<\mathrm{D_{safe}}$ is satisfied, $\theta(\mathrm{t})\in\Phi$, the empty set.
When $\rho_\mathrm{a}+\rho_\mathrm{t}\geq\mathrm{D_{safe}}\cap|\rho_\mathrm{a}-\rho_\mathrm{t}|<\mathrm{D_{safe}}$ is satisfied, we have

$$\theta(\mathrm{t})-\theta_\mathrm{t}\in[-2\pi+\arccos\mathrm{D},-\arccos\mathrm{D}]\cup[\arccos\mathrm{D},2\pi-\arccos\mathrm{D}] \tag{5}$$

Any value within the range of θ_t should be suitable for the above equation, therefore,

$$\theta(\mathrm{t})\in\left\{\left[\arccos\mathrm{D}-\frac{25}{9}\pi+\frac{2\pi}{9}\mathrm{t},-\arccos\mathrm{D}-\pi+\frac{2\pi}{9}\mathrm{t}\right]\cup\left[\arccos\mathrm{D}-\frac{7}{9}\pi+\frac{2\pi}{9}\mathrm{t},-\arccos\mathrm{D}+\pi+\frac{2\pi}{9}\mathrm{t}\right]\right\}\cap[-\zeta_{\mathrm{r\,max}},\zeta_{\mathrm{r\,max}}] \tag{6}$$

when $\arccos\mathrm{D}\leq\frac{8\pi}{9}$ is satisfied, and $\theta(\mathrm{t})\in\Phi$, when $\arccos\mathrm{D}>\frac{8\pi}{9}$ is satisfied.
The set of the values of θ satisfying the conditions of all sensors is defined as Ω, which is the intersection of $\theta(\mathrm{t})(\mathrm{t}=0,1,\ldots,8)$. When Ω is not empty, the most preferred value of θ can be obtained to make $|\theta-\theta_\mathrm{r}|$ a minimum, or else, the proper θ cannot be found. In this case, the robot will turn right angle $\zeta_{\mathrm{r\,max}}$ without any change in its position.

3. Strategies for the Invader

Assume that the invader adopts the same model as that of the individual robot. While the invader does not see any robot or static obstacle, it moves randomly; otherwise, it will move along a safety direction determined by the safety-motion strategy. The invader establishes a polar coordinate system Σ_e whose pole is its center and the polar axis direction is its heading. Denote $P_e^i(\rho_i,\theta_i)$ as the coordinates of the detecting border of sensors $S_e^i(i=0,1,\ldots,8)$ in Σ_e, where ρ_i is the reading from S_e^i when it senses any object, or else, the sensor is ignored and for convenience ρ_i is far greater than the maximum sensing range of the invader; $\theta_i \in \left[-\pi+\frac{2\pi}{9}i,-\pi+\frac{2\pi}{9}(i+1)\right]$. Based on the invader's current direction, Q (a multiple of 4) directions are generated and their set $\Im$ is depicted as follows.

$$\Im=\left\{\zeta_q \middle| \zeta_q=-\pi+\frac{2q\pi}{Q}(q=0,1,\ldots,Q-1)\right\} \tag{7}$$

The invader may move to the position $P_n^q(v_e,\zeta_q)$ on the premise of the predetermined step size v_e without any collisions when the distance from the position to the detecting border of each sensor should be greater than or equal to a safety distance L_{safe} influenced by the invader's velocity and its radius, that is,

$$d_i(\zeta_q)=\min\left(P_n{}^qP_e{}^i\right)\geq L_{safe}(i=0,1,\ldots,8) \tag{8}$$

When $\exists\zeta_q\in\Im$ satisfying eq. (8), the invader is still capable of moving, otherwise, no feasible moving direction is available and the invader is captured.

We label Ψ as the set of all directions within $\left[-\frac{\pi}{2},\frac{\pi}{2}\right)$ of the current direction and $\Psi=\left\{\zeta_q \middle| \zeta_q=-\pi+\frac{2q\pi}{Q}\left(q=\frac{Q}{4},\frac{Q}{4}+1,\ldots,\frac{3Q}{4}-1\right)\right\}$. When the invader can still move, the safety-motion strategy is used to select the best one ζ from all ζ_q satisfying eq. (8) in the set Ψ, and the best value should make $dis(\zeta_q)$ maximum, thus,

$$dis(\zeta)=\max_{\zeta_q} dis(\zeta_q)=\max_{\zeta_q}\min\left(d_0(\zeta_q),d_1(\zeta_q),\ldots,d_8(\zeta_q)\right) \tag{9}$$

If ζ is found, the invader will rotate ζ with the step size v_e, or else, it only turns right $\frac{\pi}{2}$.

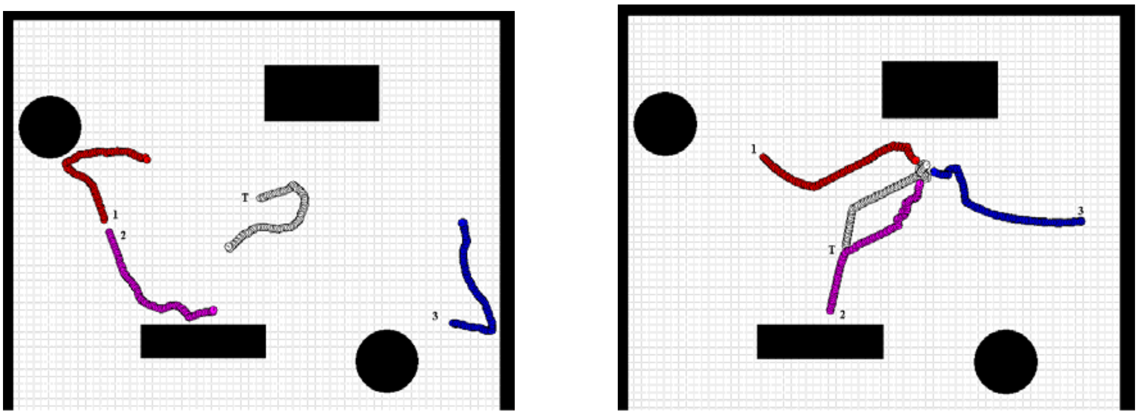
4. Simulations

In the following simulations, a random noise D_d with a mean μ and a variance σ^2 is introduced into the individual robot's sensing system in the form of $D_m=D_a(1+0.1\cdot D_d)$, where D_m and D_a are the simulated measured value and accurate value, respectively. In addition, considering the communication transfer among the robots may be failed occasionally, we assume that the robot cannot acquire the necessary information by communication with a probability of $prob_c$.

A team of robots of ID 1,2, … is required to hunt a tricky invader T. The invader is regarded as one special case of a round robot. They have the same parameters: the radius, maximum step size and maximum sensing range are 0.2, 0.1 and 3.0, respectively. μ, σ^2 in the random noise are 0 and 0.33, respectively. $prob_c$ is 0.02. The parameters in CLI approach and safety-motion strategy are shown in Table I.

Parameter	Value	Parameter	Value
τ	$\frac{\pi}{18}$	χ_{near}	1.3
χ_{far}	1.7	Δ	0.15
r_c	0.5	$step_{catch}$	15
$step_{predict}$	15	D_{safe}	0.3
$\zeta_{r\,max}$	$\frac{\pi}{2}$	L_{safe}	0.39
Q	72	——	

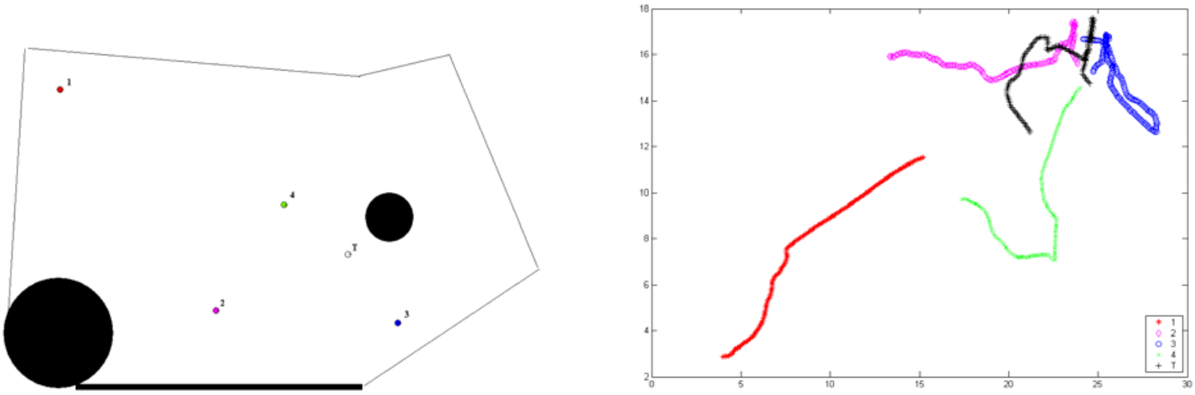
Table 1. The values of parameters concerned



(a) Before the invader is found (b) After the robotic system detects the invader

Figure 2. The trajectories of the robots and the invader in simulation 1

Fig. 2(a)~(b) show the trajectories of the robots and the invader before/after the invader is found in simulation 1, respectively. Three robots are chosen to execute the task. From Fig. 2, it is seen that at the initial stage of motion, the robot of ID 1 and 2 will keep away from each other to enlarge their visual fields. When the robot of ID 2 detects the invader T, it informs other robots and the pursuit begins. Each robot decides its own movement. By local interaction among the robots, finally the invader is captured.

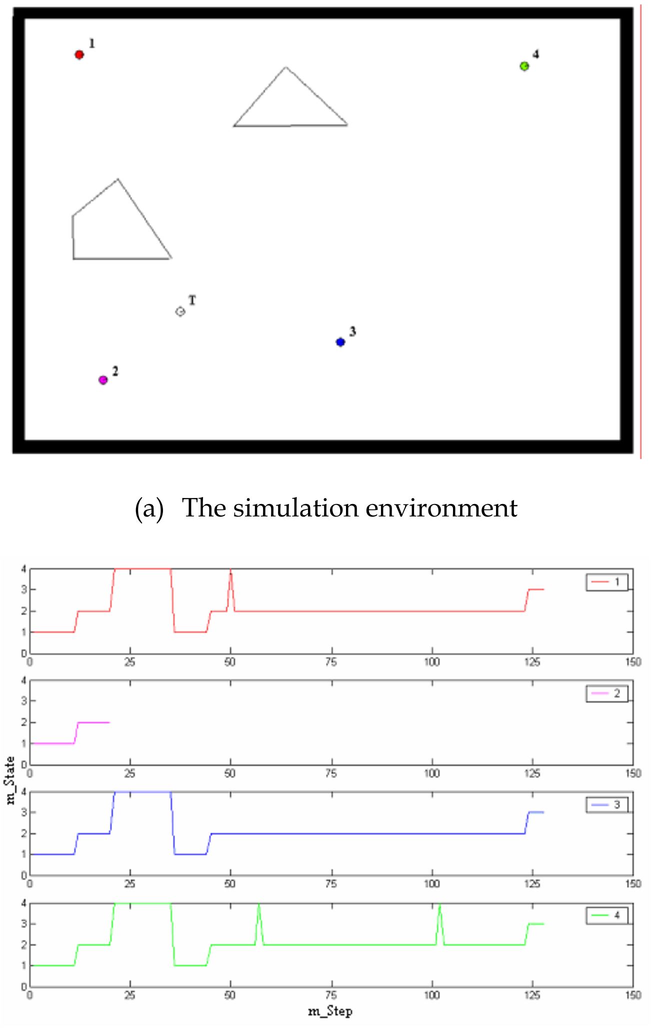


(a) The initial environment (b) The coordinates of robots and the invader

Figure 3. The simulation 2

In simulation 2, four robots are adopted and the environment is depicted in Fig. 3(a). Fig. 3(b) shows the coordinates of robots and the invader. We cans see that when the robot of

ID 1 is trying to move closer to the invader, the other three robots have already captured it. Although perhaps there are many robots (4 in this simulation) executing the task, it may be completed by local interaction among three robots. In general, more robots' participation within certain range may shorten the task time.



(a) The simulation environment

(b) The variations of the robots' states

Figure 4. The simulation 3

Simulation 3 considers the case where a robot is suddenly abnormal. The initial environment is shown in Fig. 4(a). We denote with m_State the robot's state. When m_State is chosen 1, 2, 3, 4, the robot is in the search, pursue, catch, and predict states, respectively. The variations of m_State for each robot as the task progresses are plotted in Fig. 4(b).

The process may be described as follows: The robot of ID 2 sees the invader after a short period of searching, and it broadcasts the invader's information to others. The other robots begin to move intentionally. However, several steps later, the robot of ID 2 suddenly becomes dysfunctional. The only source to provide the invader's information is cut off. Thereupon other robots try to predict the invader, and it does not work. They have to re-search the environment. Later, the robotic system finds the invader again and ultimately captures it.

To further confirm the validity of CLI approach, it is also compared with individual action (IA), an approach where each robot takes individual action regardless of other robots. Not any local interaction among the robots adopting IA approach exists. A series of simulations (simulation 4) are conducted with the distance *d* increasing (see Fig. 5). For each *d*, 20 runs were performed and the results are shown in Fig. 6, which describes the relationship of average step numbers of the robots adopting different approaches versus *d*. It can be seen that although the robots adopting IA approach sometimes may shorten the completion time than those adopting CLI approach, in many cases, the selfish behaviour of the robots adopting IA approach will lead to a delay of completion time. From all simulations conducted, CLI approach is considered an effective one.

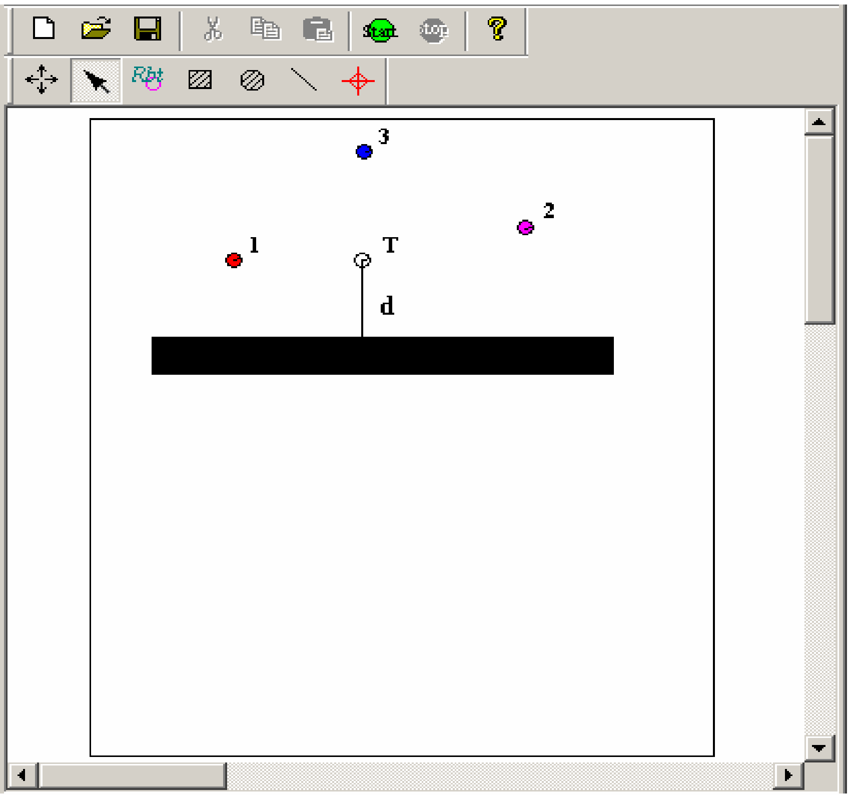


Figure 5. The test environment of simulation 4

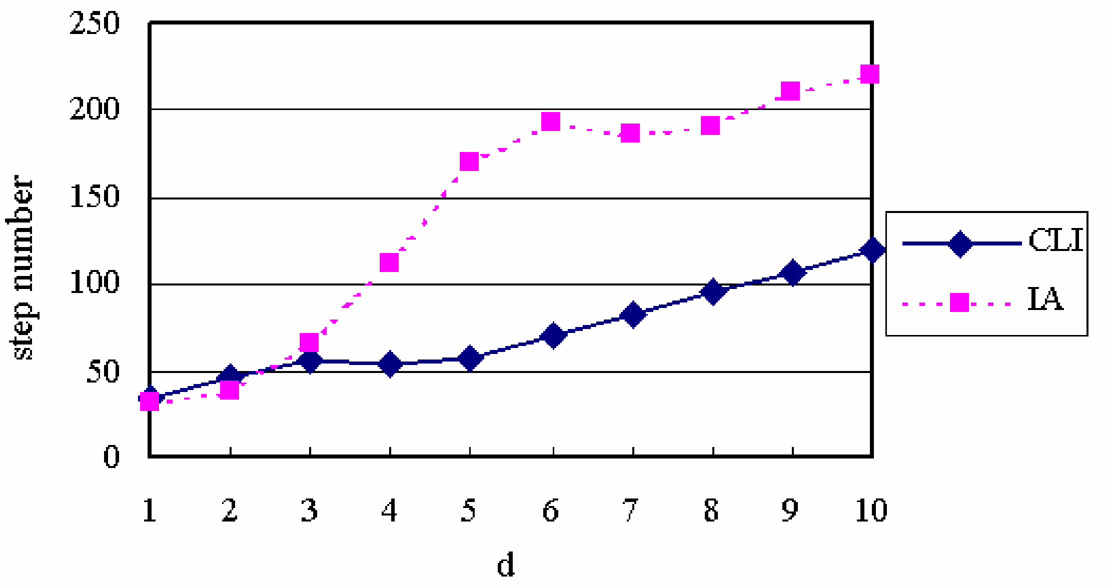


Figure 6. The comparison of different approaches for simulation 4

5. Conclusions

This chapter has mainly focused on the problem of cooperative hunting by multiple mobile robots in unknown environments. Because the positions of the robots are not exchanged among them in order to reduce the communication burden, it is hard for each robot to make a global decision. A better idea is to complete the task by local interaction among the robots.

In this chapter, an effective approach called Cooperative Local Interaction (CLI) has been proposed. The approach is robust and independent of the environments. As the invader actively tries to escape by adopting the safety-motion strategy, the difficulty of hunting is increased. The validity of CLI approach is supported by simulations.

6. References

[1] Y. U. Cao, A. S. Fukunaga, A. B. Kahng, "Cooperative mobile robotics: antecedents and directions," Autonomous Robots, 1997, 4(1), pp.7-27

[2] H. Yamaguchi, "A cooperative hunting behavior by mobile-robot troops," International Journal of Robotics Research, 1999, 18(8), pp.931-940

[3] H. Yamaguchi, T. Arai, "Distributed and autonomous control method for generating shape of multiple mobile robot group," in Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS'94, Munich, Germany, 800-807

[4] Paolo Pirjanian, M. J. Mataric, "Multi-robot target acquisition using multiple objective behavior coordination," in Proceedings of the 2000 IEEE International Conference on Robotics and Automation, San Francisco, CA, 2000: 2696-2702

[5] J. Denzinger, M. Fuchs, "Experiments in learning prototypical situations for variants of the pursuit game," in Proceedings 2nd International Conference on Multi-agent Systems, 1996, pp.48-55

[6] Ono, N., Fukumoto, K. and Ikeda, O, "Collective Behavior by Modular Reinforcement-Learning Animats," in Proceedings of the 4th International Conference on the Simulation of Adaptive Behavior, 1997: 618-624

[7] R. Vidal, S. Rashid, C. Sharp, O. Shakernia, J. Kim, S. Sastry, "Pursuit-evasion games with unmanned ground and aerial vehicles," in Proceedings IEEE International Conference on Robotics and Automation, 2001, pp.2948-2955

Market-Driven Multi-Agent Collaboration in Robot Soccer Domain

Hatice Kose, Kemal Kaplan, Cetin Mericli, Utku Tatlidede & Levent Akin

1. Introduction

In recent robotic studies, in many key areas decoupled multi-agent systems have become more popular than complex single agent systems, where the former is more robust, fast and cheap to implement. The most important reason behind this preference is to eliminate the possibility of single point of failure, which is a vital concern for single complex agents. Usage of decoupled multi-agent systems may also reduce the total cost of the entire system when it is possible to use a team of single cheap robots for performing complex tasks, instead of building up a single complex and expensive robot to satisfy all the system needs. As a result, typically when a team of robots is used, the system throughput increases while the total cost decreases. Since the robots usually have simpler physical structures, generally less complicated controller programs are necessary to manipulate the agents. The decoupled behaviors of agents can cause communication and coordination problems, however. The studies in (Dudek *et al*, 1996; Cao *et al*, 1997; Arkin and Balch, 1998; Svestka and Overmars, 1998; Stone and Veloso, 2000; Song and Kumar, 2002), refer to many different approaches to the usage of multi-agent teams in key application areas. Before dealing with coordination problem among agents in a multi-agent system, the individual capabilities and limits of the agents should be determined. This task is usually trivial for homogeneous teams. However, in heterogeneous systems, the decoupled system should provide a feasible utilization for each agent. The most important individual action for a simple robot in a homogeneous system is usually related with motion because the path planning or trajectory planning routines depend on the capabilities of the actuators of the robot. This task becomes a challenging one even for the omnidrive robots without non-holonomic constraints.

The key problem in coordinating the team of robots is to decompose the complex task in to several simple low-level actions, and assignment of these actions among the team in an optimum way such that the tasks should be formed by combining low-level actions of the robots, while avoiding collisions and allowing all the low-level actions to be implemented synchronously, and successfully. There are numerous theoretical and practical studies about decomposing a goal into subtasks, which can then be easily performed by robots with their basic actions. However, in multi-agent systems, assigning the tasks to the agents is not an easy task because the complexity of the problem increases exponentially with the number of agents and the dimensionality of the configuration space. In robot soccer, which is a challenging test bed for multi-agent systems, two teams of robots compete with each other to win the match. For the benefit of the team, the robots should work collaboratively, whenever possible. Designing a team, which can beat every opponent available, is certainly a hard mission. The market-driven approach applies the basic properties of free market economy to a team of robots for increasing the profit of the team as much as possible. It enables implementation of high level skills by using the "team spirit" of a group of simple robots, which is quite challenging and hard in case of classical planning and task allocation mechanisms, and while avoiding the collisions and enabling collaboration, allows gathering maximum profit from the implemented tasks.

Recently the market-driven approach was introduced as an alternative method for robot coordination in (Dias and Stenz, 2001). It is highly robust and avoids the single point failure problem, while increasing the team performance considerably. There are several applications of market-driven approach. The work in (Zlot *et al*, 2002) introduces the approach to multi-robot exploration. In (Gerkey and Mataric, 2002) a work on auction based multi-robot coordination is presented. These implementations seem to work well but are limited due to the static nature of the environment. Domains like agricultural areas are simple, static and do not require fast task allocation, planning and coordination as in robot soccer (Kose *et al*, 2004).

In order to provide a satisfactory solution to the task assignment and collaboration problem in robot soccer, several approaches have been implemented including static assignment (Kaplan, 2003), market based assignment (Kose *et al*, 2003) and reinforcement learning based extension to market based approach (Kose *et al*, 2004; Tatlidede *et al*, 2005). In this chapter, these approaches are compared and studied in detail.

In the next section, the robot soccer domain and in section 3, market-driven methodology is introduced. In the section 4, the previously developed approaches proposed in this work are described briefly. The results of the tests for analysis and comparison of the approaches are given in Section 5. In the section 6 there is a brief conclusion related to these approaches.

2. Robot Soccer Domain

Robot soccer domain is a well-defined environment for developing multi-agent strategies. The initial world model, constraints and goals are known. The nature of the game enables the implementation of different levels of team coordination, besides allows the development of challenging complex behaviors from simple low level tasks implemented by simple agents.

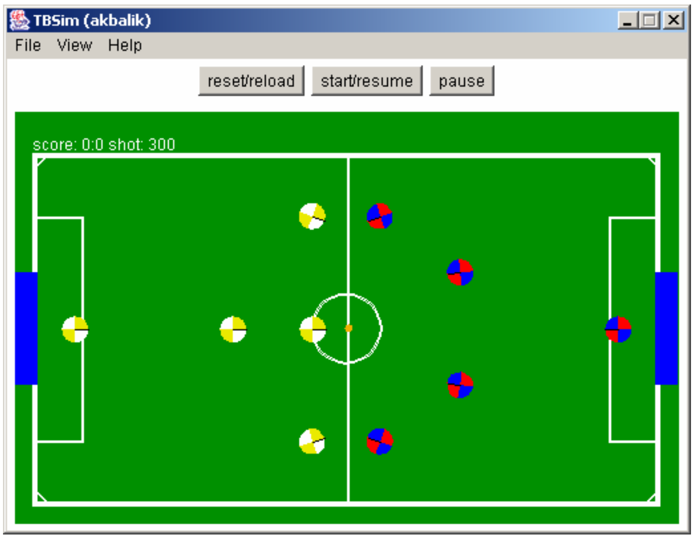


Figure 1. *Teambots* Simulator

It is also possible to test a new strategy against the existing ones in the international robot soccer competitions (FIRA, 2003; ROBOCUP, 2003). In this work, a modified version of *Teambots* simulator (Balch, 2000) is used to develop and train the proposed controllers (See Figure 1). Although *Teambots* is not used in any international robot soccer competition, it is a well-known multi-purpose simulator. In addition, it is an open source Java project, and enables easier development of different kinds of learning strategies.

The modification in the simulator is the implementation of free-ball for deadlock situations. The ball is moved to the center of the quarter of the field in which a deadlock situation occurs. In *Teambots* each team has five omnidrive robots. Localization information is available for each robot. The robots can communicate with any other robot via broadcasting specific types of information messages. The relative position of the ball and the other robots are sensed by the robot with a Gaussian noise.

3. Market Methodology

The main goal in free-markets is the maximization of the overall system profit. If each participant in the market tries to maximize its profit, as a result of this, the overall profit for the system is expected to increase. The idea of the market-driven method for multi-robot teams is based on the interaction of the robots among themselves in a distributed fashion for trading work, power and information. In general, there is a main goal of the team (i.e., building the map of an unknown planet, harvesting an agricultural area, sweeping buried landmines in a particular area, etc.). Some entity outside of the team is assumed to offer a payoff for that goal. The main goal of the system is decomposed into smaller tasks and an auction is performed for each of these tasks. In each auction, the participant robots (who are able to communicate among themselves) calculate their estimated cost for accomplishing that task and offer a price to the auctioneer. At the end of the auction, the bidder with the lowest offered price will be given the right of execution of the task and receives its revenue on behalf of the auctioneer. There are many possible actions that can be taken. A robot may open another auction for selling a task that it won from another auction, two or more robots may cooperatively work and get a task which is hard to accomplish by a single robot, or for a heterogeneous system, robots with different sensors/actuators may cooperate by resource sharing (for example, a small robot with a camera may guide a large robot without a vision system for carrying a heavy load).

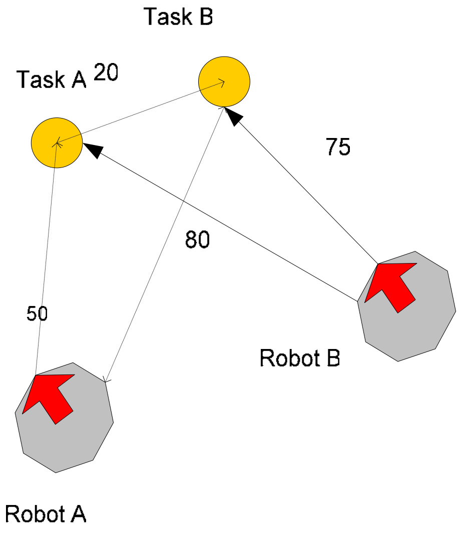


Figure 2. Market-driven scenario

In order to implement the strategy, a cost function is defined for mapping a set of resources (required energy, required time, etc.) to a real number and the net profit is calculated by subtracting the estimated cost for accomplishing the task from the revenue of the task. For example, in Figure 2. the estimated cost values are given for each task. The robots calculate their own cost values for each task. Although it seems cheaper to assign task B to robot B, when the overall profit of the team is considered, it is more profitable to assign both tasks to robot A.

4. Role Assignment Approaches in the Robot Soccer Domain

Although the robot soccer domain is well defined, it is not a trivial task to manage a robot soccer team. The first challenge is designing the low-level actions. We use a potential fields based motion strategy. The potential fields are used both for local actions like obstacle avoidance and global actions like positioning near the ball (see Figure 3). Each object on the field has a potential effect on the player. The weights of these fields are fine tuned by using genetic algorithms (Kaplan, 2003).

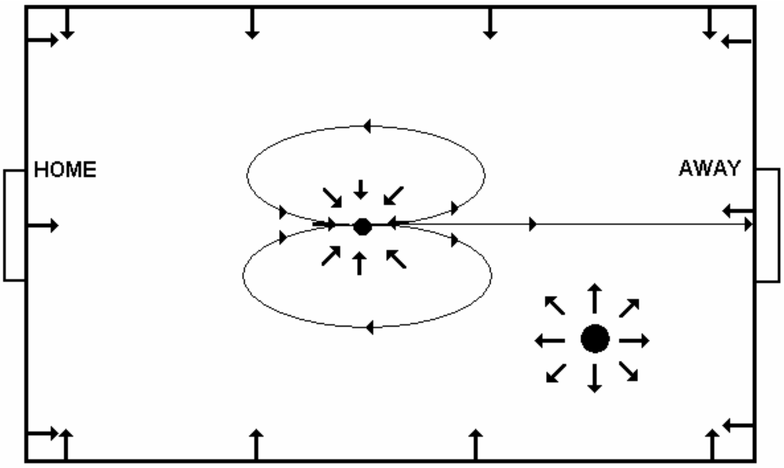


Figure 3. Potential forces.

Although with the potential fields, specific tasks, like shooting or defending can be managed, this method is not adequate for team coordination. In other words, the roles can be performed by using potential fields; however, it is not feasible to assign roles only by using potential fields.

4.1 Static Role Assignment

The first remedy for the role assignment problem is the static role assignment, which is certainly not a good practice, since it causes system failure in case of a single agent failure. The static role assignment is used only for the goalie, since this is the most optimum choice for the team success, and as the role switching time for the goalie increases, the team performance decreases.

After assigning the goalie, there are four more agents left to manage. The roles should be assigned to the agents according to some metrics, like the distance between the agent and the ball. The "*RIYTeam*", which is the first team developed with dynamic role assignment, first chooses the agent closest to the ball as the attacker. Next, the agent, which is closest to its own goal among the unassigned agents, becomes the defender. Finally, the remaining two agents help the attacker by holding strategic positions behind the attacker (Kaplan, 2003). This method has also some drawbacks. The metrics we use for selecting the attacker or defender are rather primitive. In addition, the agent, which controls the ball, should have different options other than shooting. All these requirements introduce new metrics, which are quite hard to calculate and communicate in a multi-agent system.

4.2 Market-Driven Approach

In order to address the problems mentioned in the previous subsection a new team, "*MarketTeam*" is developed where market-based strategy is used to simplify the problem by only communicating the cost values of each agent for every action instead of communicating all metrics. As a result, every robot calculates its own bid for each action according to the following equations and broadcasts only these values (Kose *et al*, 2003; Frias-Martinez *et al*, 2004).

$$C_{ES} = \mu_1.t_{dist} + \mu_2.t_{align} + \mu_3.clear_{goal} \tag{1}$$

$$C_{bidder} = \mu_4.t_{dist} + \mu_5.t_{align} + \mu_6.clear_{teammate\,(i)} + C_{ES\,(i)}, i \neq robotid \tag{2}$$

$$C_{auctionerr} = C_{ES\,(robotid\,)} \tag{3}$$

$$C_{defender} = \mu_7.t_{dist} + \mu_8.t_{align} + \mu_9.clear_{defense} \tag{4}$$

where *robotid* is the id of the robot, t_{dist} is the time required to move for specified distance, t_{align} is the time required to align for specified amount, μ_i are the weights of several parameters to emphasize their relative importance in the total cost function, $clear_{goal}$ is the clearance from the robot to goal area, $clear_{ball}$ is the clearance from the robot to ball, $clear_{defence}$ is the clearance from the robot to the middle point on the line between the own goal and the ball, and similarly $clear_{teammate(i)}$ is the clearance from the robot to the position of a teammate. Each robot should know its teammates score and defense costs. In our study each agent broadcasts its score and defense costs to its teammates.

This approach increases overall performance; however, there are still problems with the role assignment strategy. The first one is the restriction of one-to-one assignment. Previous strategies assign only one agent for each role simultaneously. However, if it is not restricted in the rules of the game, more than one agent may perform the same role to increase the performance. Another problem is the training of the system. The coefficients of the cost functions, which are used in the previous strategies, are fine tuned by using genetic algorithms. However, these cost functions are manually introduced to the system by human experts. For example, while calculating the cost of defense role for each robot, we use a specific formulation. This formulation may not be the optimum one for selecting the agent for defense action. This means, that the learning phase is designed to optimize the coefficients of the cost function instead of finding the optimum cost functions.

4.3 Reinforcement-Based Market-Driven Approach

To solve these problems, we use Reinforcement Learning (RL) to learn the role assignment process without changing the actual implementations of the roles. RL is a learning method, which can be used when the agent is only informed about the consequences of a sequence of its actions. The RL implementation replaces the role assignment in the original market algorithm mentioned above, with a Q(λ)-Learner (Peng and Williams, 1996). Q(λ)-Learning is a variant of RL and an extension to simple Q-Learning. Q-learning algorithm uses only one step data while updating Q-values. Eligibility traces can be used to keep track of all the actions taken by the agent to reach a terminal state (Sutton *et al*, 1996). Q(λ) is widely used and it is generally believed to outperform simple one-step Q-learning, since it uses single experiences to update multiple state/action pairs (SAPs) that have occurred in the past. Generally, the Q-functions learned by the agents are represented in tabular form with one output value for each input tuple. But it is not possible to represent more realistic worlds with this approach, where the number of states can be prohibitively larger or continuous. One way of handling such problems is to use function approximation.

For function approximation and state generalization in RL, Cerebellar Model Articulation Controller (CMAC) is used. CMAC was introduced by Albus (Albus, 1975) as a simple model of the cortex of the cerebellum. It is a biologically inspired learning method similar to neural networks. The main reason for using the CMAC is its efficiency in learning and operation, which makes it suitable for function approximation.

As in the previous strategies, the goalie role is statically assigned to an agent and does not change. Since it is always feasible to control the ball, the closest agent to the ball is assigned as the attacker and advances to the ball. The remaining three agents select the best role for themselves in the current situation according to the team policy (Figure 4). State representation consists of perceptual and logical parameters. The perceptual parameters are relative distance to the ball, two goals, and other players (4 teammates and 5 opponents in this case).

Each relative distance variable is composed of two parameters which are distance angle between the normal line and the agent. The logical parameters are the cost values (4 players' offensive and defensive cost values) and the closest player to the ball. There are 24 perceptual parameters and 9 logical parameters so totally 33 parameters are used to construct state vector. Unfortunately, this method suffers because of the large state vector. (Kose *et al*, 2004).

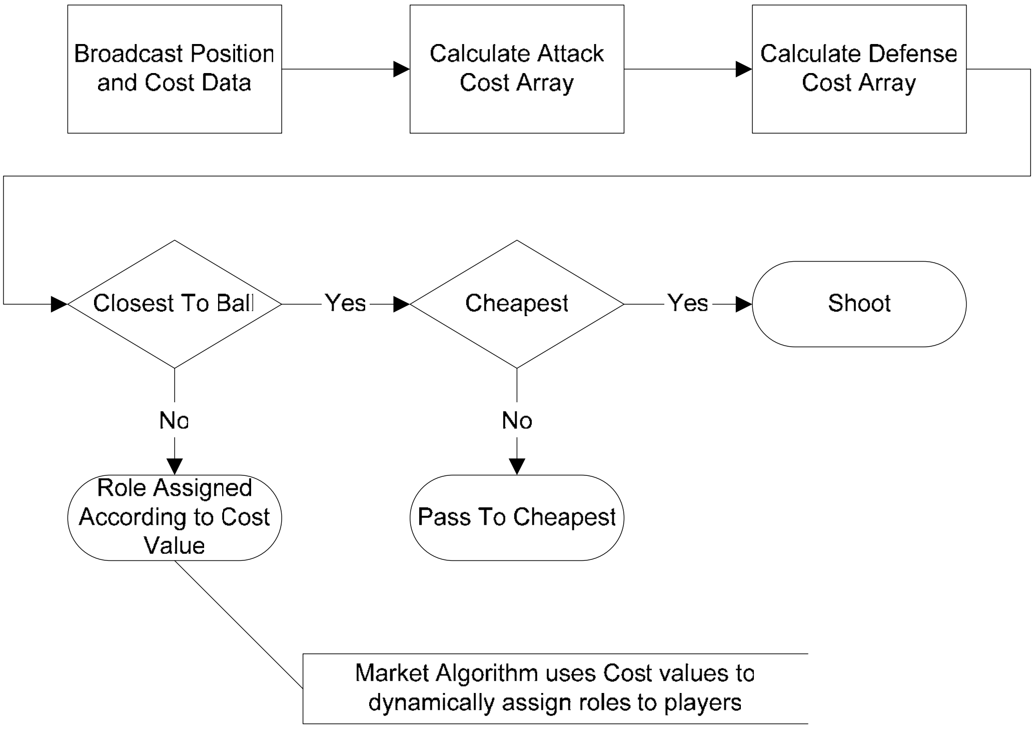

Broadcast Position and Cost Data
Calculate Attack Cost Array
Calculate Defense Cost Array
Closest To Ball
Yes
Cheapest
Yes
Shoot
No
No
Role Assigned According to Cost Value
Pass To Cheapest
Market Algorithm uses Cost values to dynamically assign roles to players

Figure 4. Flowchart for task assignment

4.4 New Approach

The latest team developed in this study is an extension of the above described RL based approach. However, the state vector is modified here. In general, the state vector should include information about the agents and the ball. However, raw position data is not feasible to encode in the state vector. Therefore a grid decomposition for the field is proposed. In a real soccer game, the field is usually divided into three horizontal sections, where the upper and lower sections are the wings. Similarly, the field can be divided into three main vertical sections, which are forward, midfield and backward. Nevertheless, midfield can be further divided into two subsections. As a result the field is divided into 12 grids as shown in Fig. 5.

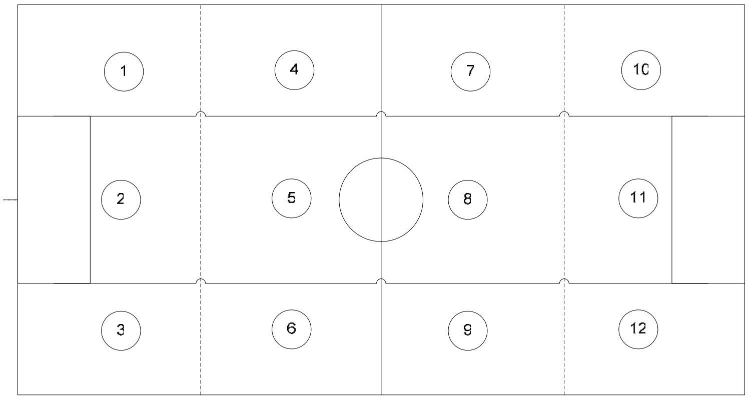


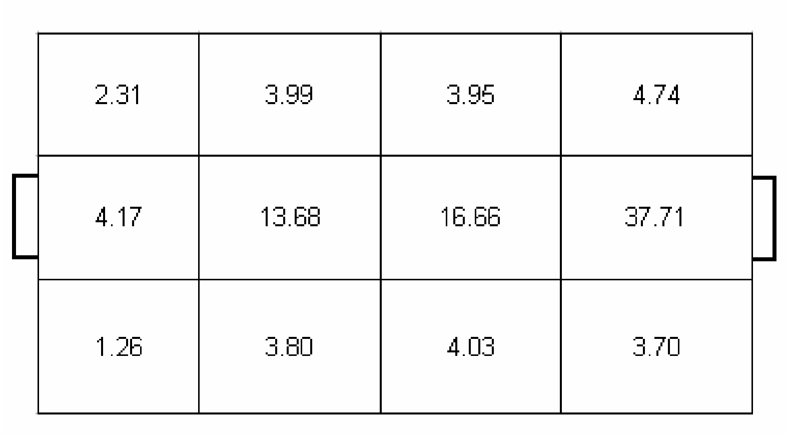
Figure 5. The grid decomposition of the field

The state vector has the following metrics,

- **Ball position:** The number of the grid which contains the ball. (1 state vector element)
- **Ball possession:** The number of the team which possesses the ball. The possession is simply calculated by finding the agent closest to the ball. (1 state vector element)
- **Own role:** The role number assigned to the agent by the market team strategy. (1 state vector element)
- **Teammate positions:** The grid numbers of the teammate agents other than the goalie and the attacker. (3 state vector elements)
- **Opponent positions:** The number of opponents in each grid. We do not use the number of the grids because the density of the opponent agents in each grid is more important than the individual opponent agent positions. (12 state vector elements)

This state vector reduces the number of state variables from 33 to 18. The possible actions are the selection of attacker role, defender role, and secondary attacker role. The implementation details are the same as the previous reinforcement learning based team (Kose et al, 2004). After training, the average percentages of the positioning of the opponent robots on the playground are given in Figure . The percentages are averages of three matches.

According to the 3 points system, in which the winner takes 3 points and each team take 1 point for draw, the performance of the learning team is given in Figure, where y-axis is the cumulative point for 50 match epochs (Tatlidede *et al*, 2005).



2.31	3.99	3.95	4.74
4.17	13.68	16.66	37.71
1.26	3.80	4.03	3.70

Figure 6. Percentages of positioning of opponent robots

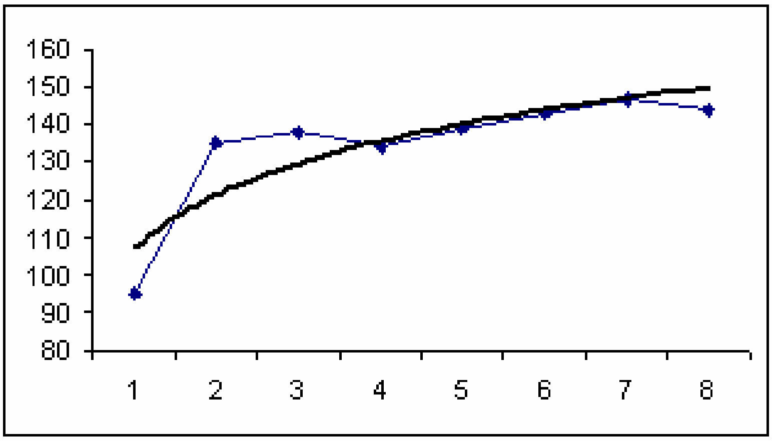


Figure 7. Results during learning

4. Results

The new team, which is based on reinforcement learning with the reduced state vector, is compared to four other teams. The first opponent is *SchemaNewHetero*, which uses perceptual and motor schemas. This is a moderate built-in team delivered with the *Teambots* simulator. The second team, *AIKHomoG* is one of the strongest built-in opponents. *AIKHomoG* team uses dynamic role assignment for strategy and potential fields for movement. The third team is the *RIYTeam*, which has very simple role assignment strategy as mentioned before. The next opponent team, is the *MarketTeam*. The final opponent is the *MarketQL*, which is the RL based team with large state vector.

As seen in Table 1, the proposed team defeats all other opponents. It should be stressed that the only difference between the *RIYTeam* and the *MarketTeam* and the new team is the role assignment strategy. The new team is trained to find the optimum role assignment strategy without any constraint. The previous teams suffer from the assumptions which are made by human experts. For example in *RIYTeam*, the robot closest to the ball is selected as the attacker. This assignment is subject to the assumption that the distance is the only metric which affects the role assignment strategy. However, in the new team, there is no such assumption, which means at the beginning of learning phase each agent is free to choose any role, except goalie.

Team	Play	Win	Draw	Lost	For Goal	Against Goal
SchemaNewHetero	90	60	28	2	163	19
AIKHomoG	90	78	9	3	203	38
RIYTeam	90	50	40	0	81	5
MarketTeam	90	36	48	6	56	15
MarketQL	90	33	50	7	53	17

Tabele 1. Results

5. Conclusion

In this work, the target is the coordination problem among the members of a robot soccer team. In order to solve this problem several methods which are extensions of a market-driven approach are implemented. In this work these approaches are studied and compared in detail.

The first developed method was the method with static role assignment. Since it has many drawbacks, a novel market-driven approach was implemented to increase the team success by using the full benefits of collaboration. In this first version, roles are fixed, and the agents are assigned suitable roles according to the available cost functions to increase success, in the current situation. This strategy was quite successful and takes good results in during the matches done by other teams, but there are different teams with different game strategies like in the real life case, so there is a need to change the game strategy (e.g. playing offensive or defensive) according to the opponent team strategy. So the original *MarketTeam* is extended by the addition of reinforcement-based learning method, which allows the team to learn new strategies, as it plays matches with other teams, and use a dynamic strategy to choose the roles for the players. Later this strategy which uses market-based cost values and other domain specific values in its state vector is further extended to eliminate the drawbacks, and increase success.

The results show that reinforcement learning is a good solution for role assignment problem in the robot soccer domain. However, encoding of the problem into the learner is an important issue. When the configuration space is quite large, the policy may not cover all possible states. As a result, the agent is forced to select random actions and the system performance decreases. The communication problem is not addressed in this work. It is assumed that each agent can broadcast limited amount of data. The controller simply collects available data from any other agent. The data may be noisy. Since, at each frame the communication data is refreshed, the error is not cumulative.

The solution can also be used in other highly dynamic environments where it is possible to introduce some reinforcement measures for the team. In the robot soccer domain, the reinforcement measures are the goals scored by either our team or the opponent team.

8. References

Albus, J.S. (1975). Data storage in the cerebellar model articulation controller (CMAC), *Transactions of the ASME: Journal of Dynamic Systems, Measurement, and Control,* pp. 228-233

Arkin, R. C. and Balch, T. (1998). Cooperative multi-agent robotic systems, *Artificial Intelligence and Mobile Robots,* D. Kortenkamp, R. P. Bonasso, and R. Murphy, Eds. Cambridge, MA: MIT/AAAI Press

Balch, T. (2000). *Teambots, Available from:* http://www.cs.cmu.edu/~trb/*Teambots*/ Domains/SoccerBots, *Accessed:* 2000

Cao, Y.U.; Fukunaga, A.S. and Kahng, A.B. (1997). Cooperative mobile robotics: antecedents and directions, *Autonomous Robots*, 4, pp.2–7

Dias, M. B. and Stenz, A. (2001). A Market Approach to Multi-robot Coordination, *CMU-RI-TR-01-26*, August

Dudek, G.; Jenkin, M.R.M.; Milios, E.and Wilkes, D. (1996). A taxonomy for multi-agent robotics, *Autonomous Robots*, 3(4), pp. 375–397

FIRA (2003). *Available from:* http://www.fira.net/, *Accessed:* 2003.

Frias-Martinez, V.; Sklar, E. and Parsons, S. (2004). Exploring auction mechanisms for role assignment in teams of robots, *Proceedings of the RoboCup Symposium,* Lisbon, 2004

Gerkey, P.B. and Mataric, M.J. (2002). Sold!: Auction methods for multi-robot coordination, *IEEE Transactions on Robotics and Automation, special issue on Advances in Multi-Robot Systems,* 18(5), 758-786

Kaplan, K. (2003). Design and Implementation of fast controllers for Mobile Robots, Master Thesis, January

Kose, H.; Mericli, C.; Kaplan, K. and Akin, H. L. (2003). All Bids for One and One Does for All: Market-Driven Multi-Agent Collaboration in Robot Soccer Domain, *Proceedings of Computer and Information Sciences-ISCIS 2003, 18th International Symposium*, pp. 529-536, LNCS 2869, Antalya, Turkey, November 2003

Kose, H.; Tatlidede, U.; Mericli, C.; Kaplan, K. and Akin, H. L. (2004). Q-Learning based Market-Driven Multi-Agent Collaboration in Robot Soccer, *Proceedings of TAINN 2004, Turkish Symposium On Artificial Intelligence and Neural Networks*, pp.219-228, Izmir, Turkey, June 10-11, 2004.

Peng, J. and Williams, R. (1996). Incremental multi-step q-learning, *Machine Learning*, pp. 283-290

ROBOCUP (2003). *Available from:* http://www.robocup.org/, *Accessed:* 2003.

Song, P. and Kumar, V. (2002). Potential field based approach to multi-robot manipulation, *In Proceedings of the 2002 IEEE Int'l Conf. on Robotics and Automation,* Washington D.C., May

Stone, P. and Veloso, M. (2000). Multi-agent systems: A survey from a machine learning perspective, *Autonomous Robots*, 8(3)

Sutton, R. S. (1996). Generalization in reinforcement learning: Successful examples using sparse coarse coding, *Advances in Neural Information Processing Systems,* MIT Press

Svestka, P. and Overmars, M. (1998). Coordinated path planning for multiple robots, *Robotics and Autonomous Systems*, 23, pp:125 – 152

Tatlidede, U.; Kaplan, K.; Kose, H and Akin, H. L. (2005). Reinforcement Learning for Multi-Agent Coordination in Robot Soccer Domain, *Fifth European Workshop on Adaptive Agents and Multi-Agent Systems*, Paris, March 21-22, 2005 (accepted)

Zlot, R.; Stenz, A.; Dias, M. B. and Thayer, S. (2002). Multi-Robot Exploration Controlled by a Market Economy, *Proceedings of the IEEE International Conference on Robotics and Automation*, May

The SocRob Project: Soccer Robots or Society of Robots

Pedro U. Lima & Luis M. M. Custodio

1. Introduction

Cooperative Robotics is a modern research field, with applications to areas such as building surveillance, transportation of large objects, air and underwater pollution monitoring, forest fire detection, transportation systems, or search and rescue after large-scale disasters (Balch, T. & Parker, L., 2002). In short, a population of cooperative robots behaves like a distributed robot to accomplish tasks that would be difficult, if not impossible, for a single robot. Many lessons important for this domain can be learned from the Multi-Agent Systems field of Artificial Intelligence (AI) concerning relevant topics for Cooperative Robotics, such as distributed continual planning (desJardins, M. E., *et al*, 1999), task allocation (Ferber, J., 1999), communication languages or coordination mechanisms (Decker, K. S., & Lesser, V. R., 1995).

Robotic soccer is a very challenging problem, where the robots must cooperate not only to push and/or kick an object (a ball) towards a target region (the goal), but also to detect and avoid static (walls, stopped robots) and dynamic (moving robots) obstacles while moving towards, moving with or following the ball. Furthermore, they must cooperate to defeat an opposing team. All these are features common to many other cooperative robotics problems.

This paper surveys the several research problems addressed by the SocRob project, carried out by the Intelligent Systems Laboratory at the Institute for Systems and Robotics - Instituto Superior Técnico (ISR/IST) in Lisbon, building a Systems Theory standpoint on AI concepts. In Section 2, we describe our view of the general problem involving multiple robots that act as a team, cooperating and coordinating their actions to attain the team goal. Needless to say, single-robot "traditional" research problems are covered, both from the sub-system and from the integration standpoints. Natural extensions to cooperative multi-robot teams are also detailed. The problems addressed so far and the solutions we obtained for them are described in Section 3. Open problems of interest for the project and clues on how we intend to approach their solution are discussed in Section 4. We end the paper drawing some conclusions in Section 5.

2. A General Multi-Robot Cooperation and Coordination Problem

Many researchers around the world are designing mobile robots capable to display increasing autonomy and machine intelligence properties. Most groups concentrate on specific subsystems of a robot, such as the planner, the navigator, or the sensor fusion. What usually is missing in their design is a systematic way to glue together all these subsystems in a consistent fashion. Such a methodology, should one be available, would help engineering the mobile robots of the future.

One of the key factors of success for a robot lies on its capability to perceive correctly its surrounding environment, and to build models of the environment adequate for the task the robot is in charge of, from the information provided by its sensors. Different sensors (e.g., vision, laser, sonar, encoders) can provide alternative or complementary information about the same object, or information about different objects. Sensor fusion is the usual designation for methods of different types to merge the data from the several sensors available and provide improved information about the environment (e.g., about the geometry, color, shape and relevance of its objects).

When a team composed of several cooperating robots is concerned, the sensors are spread over the different robots, with the important advantage that the robots can move (thus moving its sensors) to actively improve the cooperative perception of the environment by the team. The information about the environment so obtained can be made available and regularly updated by different means (e.g., memory sharing, message passing, using wireless communications) to all the team robots, so as to be used by each robot subsystems. Once the information about the world is available, one may think of using it to make the team behave autonomously and machine-wise intelligently.

Three main questions arise for the team:

- Where and which a priori knowledge about the environment, team, tasks and goals, and perceptual information gathered from sensors, should be kept, updated and maintained? This involves the issue of distributed knowledge representation adequate to consistently handle different and even opposite views of the world.
- What must be done to achieve a given goal, given the constraints on time, available resources and distinct skills of the team robots? The answer to this should provide a team plan.
- How is the actual execution of a plan handled, ensuring the consistency of individual and team (sub)-goals?

So far, a bottom-up approach to the implementation of a cooperative multi-robot team has been followed in the SocRob project, starting from the development of single robot subsystems (e.g., perception, navigation, decision-making) and moving towards relational behaviors, comprehending more than one robot.

However, a key point is a top-down approach to system design. The design phase establishes the specifications for the system:

- **qualitative specifications** - concerning formal logical task design so as to avoid deadlocks, livelocks, unbounded resource usage and/or sharing non-sharable resources, and to choose the primitive tasks that will span the desired task space;
- **quantitative specifications** - concerning performance features, such as accuracy (e.g., the spatial and temporal resolution, as well as the tolerance interval around the goal, at each abstraction level), reliability and/or minimization of task execution time given a maximum allowed cost.

To support this top-down design and bottom-up implementation philosophy, suitable functional and software architectures, respectively, must be conceived prior to the development of all the sub-systems.

2.1 Single-Robot Research Problems

Most of the problems tackled so far within the SocRob project concern the sub-systems of the individual robots composing a team. From our standpoint, relevant topics are:

Functional and Software Architectures: Modern robots should be designed based on a top-down design from specifications to ensure desired performance levels (both qualitative and quantitative). Therefore, the designers should start by specifying a functional architecture which will guide the design of the robot sub-systems in an integrated fashion, i.e., each sub-system is not necessarily designed to optimize its performance but rather aiming at optimizing the overall system performance. Another important issue is to determine, given the desired task space (i.e., the set of tasks that will have to be carried out by the robot in a particular application), the minimal set of primitive tasks that will span that task space. Moreover, the final implementation should be supported on a suitable software architecture designed to allow real-time multi-processing, information sharing and mutually exclusive allocation of shared resources among the robot sub-systems.

Single-Robot Task Planning: Given the primitive task set referred in the previous item, the robot must be able, given the current and past world states (including its own internal state), to compose primitive tasks so as to come up with a plan that carries out a given desired task. There may be more than one plan that accomplishes a task, but a posterior decision system should be able to determine, eventually based on machine learning, the one that achieves the best performance, based on the available information and prediction horizon.

Single-Robot Task Coordination: Plans must be such that they allow continuous handling of the environment uncertainties and unexpected events. Once a plan is determined, task coordination deals with its execution. Plan execution must, at least, take into account the detection of events, smooth transitions between primitive tasks, synchronization of primitive tasks executed concurrently, mutual exclusion when two or more tasks attempt to access shared resources, iterative estimation of primitive task performance, learning how to improve a plan over time by choosing more convenient algorithms among those available for each primitive task, and so on.

Navigation: The navigation system is an important sub-system of a mobile robot. In many applications one important feature of the navigation system concerns the ability of the robot to self-localize, i.e., to autonomously determine its position and orientation (posture). Using posture estimates, the robot can move towards a desired posture, i.e., by following a pre-planned virtual path or by stabilizing its posture smoothly (Canudas de Wit, C., et al, 1996). If the robot is part of a cooperative multi-robot team, it can also exchange the posture information with its teammates so that appropriate relational and organizational behaviors may be established. In robotic soccer, these are crucial issues. If a robot knows its posture, it can move towards a desired posture (e.g., facing the goal with the ball in between). It can also know its teammate postures and prepare a pass, or evaluate the game state from the team robot locations. Most approaches to navigation determine with high accuracy the posture of the robot with respect to a given coordinate frame. However, this approach is typically resource-consuming, requiring a robot to spend a significant percentage of its processing time with the navigation sub-system, disregarding other important sub-systems, such as perception or planning, to name but a few. Furthermore, high accuracy is not always required for navigation purposes. One may be just interested to move closer to an object, rotate to see a given landmark, or move to another region. In those cases, another approach to navigation, known as topological (or relative) navigation, is advisable.

Object Recognition and Tracking Using Sensor Fusion: The ability to discriminate and recognize its surrounding objects, to distinguish the relevant ones and to track them, are major problems for any robot. For soccer robots, this problem is simplified since the relevant objects are distinguished by their colors (e.g., the ball is orange, the goals are blue and yellow). Nevertheless, fast and reliable color segmentation is not a trivial problem and requires some attention too. Furthermore, object detection may be performed by more than one sensor, such as different virtual sensors based on the vision transducer (e.g., mass center, edge detector, color segmentation), sonars, infrared and others. Therefore, sensor fusion arises as an important topic.

2.2 Cooperative Multi-Robot Research Problems

Functional and Software Architectures: If a team of cooperative robots is involved, the single-robot architectures of each of the team members must be integrated in the overall team architecture. The most usual solutions concerning the software architecture are

- centralized, where one of the robots (or an external machine) processes the data acquired and sent by all the team members, takes all the team decisions and sends commands to the others;
- distributed, where local data processing is made at each of the robots but then information is sent to one of them to take the decisions;
- fully decentralized, where each robot takes its own decisions based on its own data and on information exchanged with its teammates.

The functional architecture of a behaviour-based multi-robot team must also classify behaviours according to the distribution of responsibilities by the team robots. One such division consists of considering organizational, relational and individual behaviours (Drogoul, A., and Collinot, A., 1998), further described below.

Multi-Robot Task Planning and Allocation: In the multiple-robot case, plans must take into account the distributed nature of the task at hand. Different tasks must be allocated to the different robots in the team, according to their skills and performance. So, the planning and task allocation system must be able to establish (sub)groups of robots within a team, and the robots must have and know how to deal with the notion of "belonging to a group". Therefore, plans must also include synchronization and communication among team members involved in the task. Moreover, if a robot cannot fulfill its assigned task, the task may simply be re-assigned to a robot within the group, a new robot may be integrated in the group to perform that task, or in the worst case a re-planning strategy has to be applied.

Multi-Robot Task Coordination: The extension of task coordination to a team of multiple robots introduces issues related to knowledge distribution and maintenance, as well as communications and related problems (e.g., noise, protocols, limited bandwidth). Furthermore, communication can be explicit (e.g., through wireless radio-frequency channels) or implicit (e.g., through the observation of teammate actions, should an a priori

model of the teammates behaviour exist). The coordination of a task carried out by a team of cooperating robots involves signalling events detected by one robot which are relevant for some or all of its teammates and/or to exchange information obtained locally by the different robots of the team. Whenever a formation is required, several formation topologies are possible and the one suitable for the task at hand must be chosen as part of the coordination process. Although not inevitable, communications among team members are also required to keep the formation under control.
When the population is composed of heterogeneous robots, if a robot has to perform a particular task for which it does not have the necessary skills, it may ask another robot with the adequate skills to carry it out. In the particular case of the SocRob robotic team, where the robots are homogeneous, examples of cooperative behaviour are the cooperative localization of the ball, the execution of a pass, the dynamical exchange of player roles or the decision of which robot should go for the ball. All of them require some form of inter-robot coordination and underlying teamwork methodologies.

Distributed World Modeling: A team composed of multiple robots, possibly heterogeneous concerning on board sensing, can benefit from the availability of a world model, obtained from the observations made by the different team members and its on board sensors. This world model can be richer than if it were obtained by a single robot, due to the coverage of a broader area by a more diversified set of sensors. It can also be distributed through the teammates, e.g., by keeping in a robot information which is only relevant locally and by broadcasting information gathered locally but which is of interest for the team as a whole. The sensor fusion problem is similar to the single-robot case, with the important difference that the sensor subsets are now independently mobile and can be actively positioned to improve the determination of object characteristics.

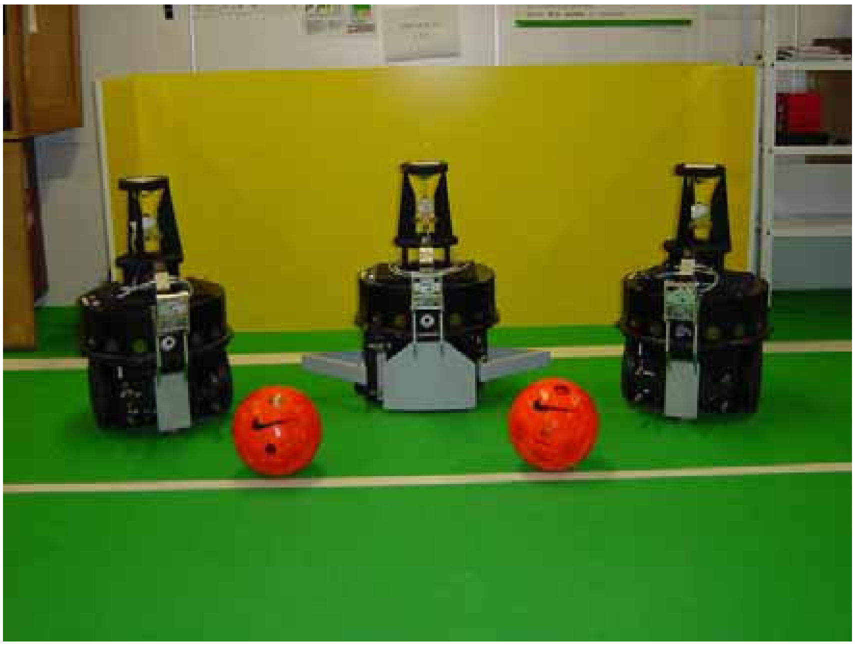


Figure 1. Three robots of the current SocRob team

3. Problems Already Addressed

A key issue of the research work developed under the SocRob project is the application of conceptual results to real robots participating in the Middle Size League (MSL) of RoboCup. The current robot team, displayed in Fig. 1, is composed of 4 Nomadic Super

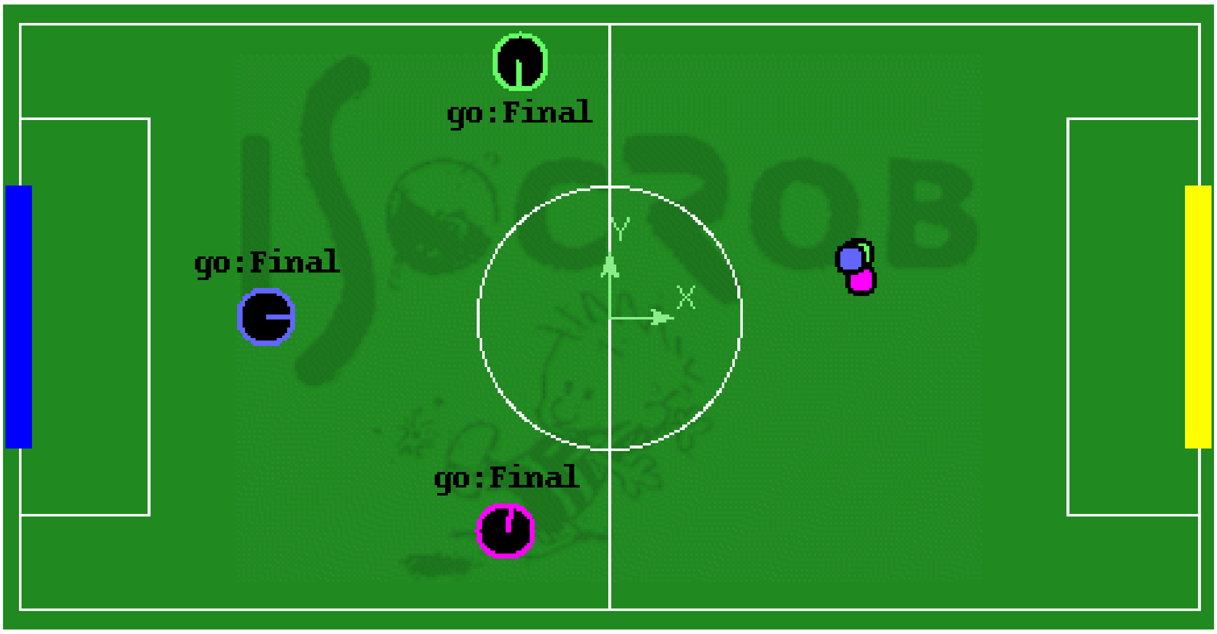
Scout II commercial platforms, later significantly modified by our group, each of them including:

- Two-wheel differential drive kinematics;
- Sixteen sonar sensors radially distributed around the robot, equally spaced;
- Motorola MC68332 based daughter board with three-axis motor controller, sonar and bumper interface, and battery level meters;
- Two 12V batteries, 18Ah capacity;
- Pentium III 1000MHz based motherboard, with 512MB RAM, 8GB disk;
- Two Philips USB WebCam 740K Pro. One of the WebCams looks ahead of the robot (front cam), while the other, together with a convex mirror, designed to directly obtain the soccer field bird's eye view, preserves the field geometry in the image (up cam);
- IEEE 802.11b wireless Ethernet PCMCIA card;
- Pneumatic kicking device, based on Festo components, plus one bottle for pressurized air storage;

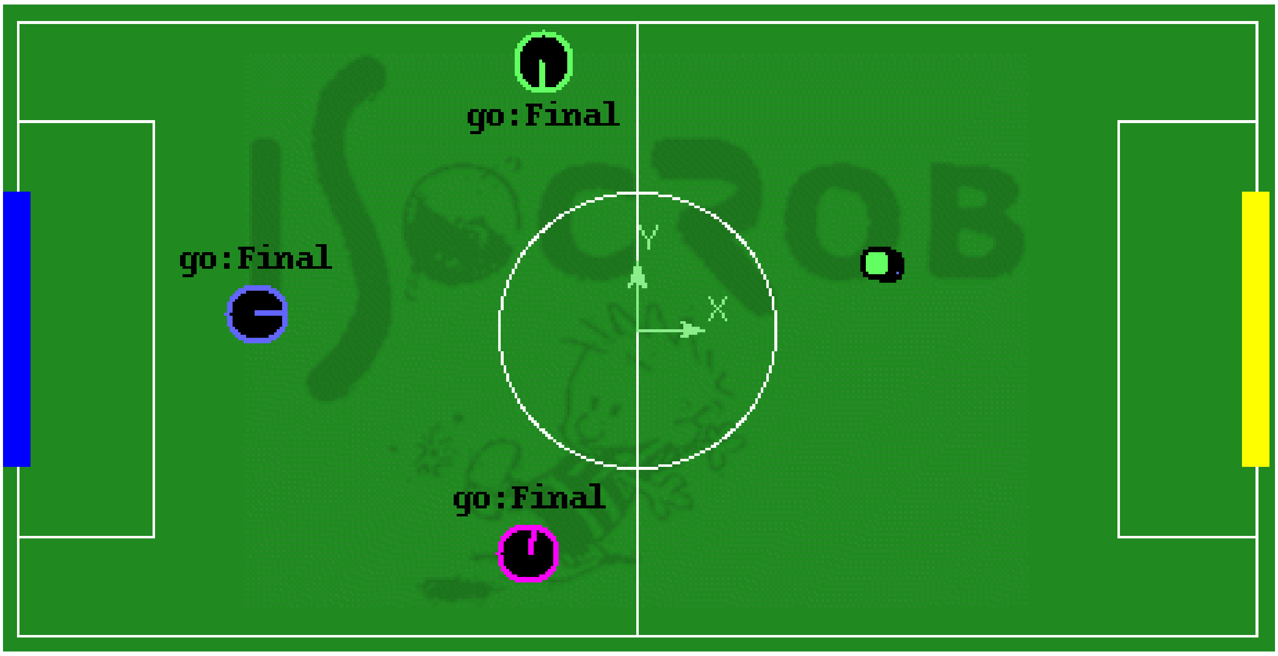
In the remaining subsections, we describe some of the research problems addressed and solved for this team of robots.

3.1. Color Segmentation and Cooperative Object Recognition

A color segmentation interface was developed, providing two alternatives to discriminate the relevant MSL colors in HSV (Hue-Saturation-Value) color space (Gonzalez, R., & Woods, R., 1992): i) adjusting HSV intervals or ii) graphically selecting regions with a given pixel color. The two approaches are cumulative. Furthermore, object segmentation is a topic directly related to the previous one, as we discriminate objects, namely the ball and the goals, not only based on their color, but also on their shape (e.g., by fitting circles to observed orange bulbs and identifying the ball with the closest and more circular bulb).



a)



b)

Figure 2. Example of sensor fusion. In the snapshots of the team interface, the larger circles with a mark denoting orientation are robots. The smaller circles represent the ball positions, estimated by each of the robots: a) local (internal to each robot) sensor fusion enabled and global (among team robots) sensor fusion disabled; b) both local and global sensor fusion enabled.

A topic of current research within the project is the use of sensor fusion for world modeling. The goal is to maintain and update over time information on the relevant objects, such as ball position and velocity, teammates pose and velocity, opponents pose and velocity, or position of the goals with respect to the robot.

Such information is obtained by each robot from the observations of its front and up cameras and then fused among all the team robots (Pinheiro, P. & Lima, P., 2004), using a Bayesian approach to sensor fusion, as depicted in Fig. 2. Currently this approach is used to provide information on ball position to all the team members, therefore enabling robots that do not see the ball to know where it is, besides improving ball localization reliability. Fusion is not used when two robots disagree (in probabilistic terms) on the ball localization.

3.2. Vision-Based Self-Localization

An algorithm that determines the posture of a robot, with respect to a given coordinate system, from the observation of natural landmarks of the soccer field, such as the field lines and goals, as well as from a priori knowledge of the field geometry, has been developed within the SocRob project (Marques, C., & Lima, P., 2001). The algorithm is a particular implementation of a general method applicable to other well-structured environments, also introduced in (Marques, C., & Lima, P., 2001).

The landmarks are processed from an image taken by the up cam omni-directional vision system, an image of which is depicted in Fig. 3. The image green-white-green color transitions over a pre-determined number of circles centered with the robot are collected as the set of transition pixels.

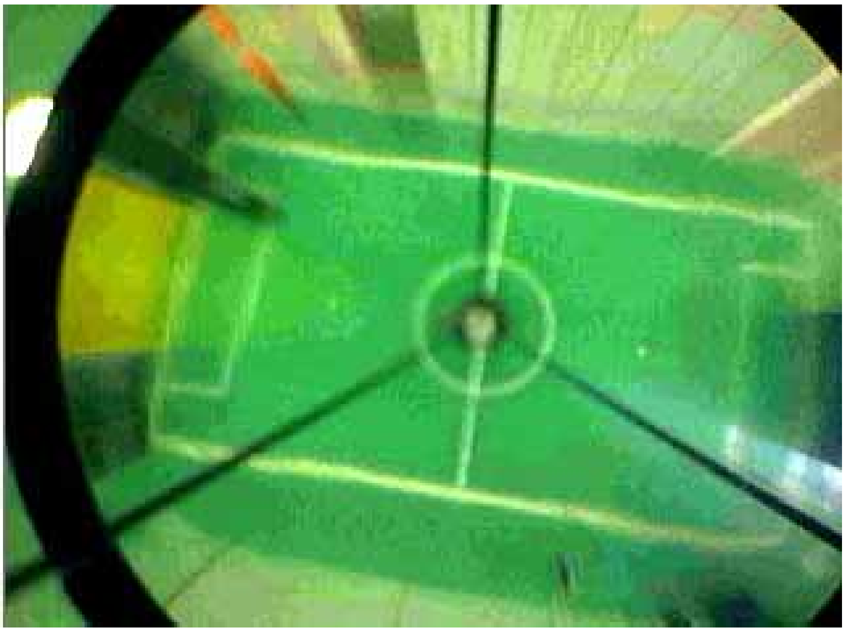


Figure 3. Bird's eye-view of the field obtained by the top catadioptric systems of the robots in Fig. 1

The Hough Transform is applied to the set of transition pixels in a given image, using the polar representation of a line (Gonzalez, R., & Woods, R., 1992):

$$\rho = x_i^t . \cos\phi + y_i^t . \sin\phi \tag{1}$$

where (x_i^t, y_i^t) are the image coordinates of transition pixel p^t and ρ, ϕ are the line parameters. The q straight lines $(\rho_l, \phi_l), \ldots, (\rho_q, \phi_q)$ corresponding to the top q accumulator cells in Hough space are picked and, for all pairs $\{ (\rho_j, \phi_j), (\rho_k, \phi_k), j,k=1, \ldots,q, j \neq k \}$ made out of those q straight lines the following distances in Hough space are computed:

$$\begin{aligned} \Delta\phi &= \left|\phi_j - \phi_k\right| \\ \Delta\rho &= \left|\rho_j - \rho_k\right| \end{aligned} \tag{2}$$

Note that a small $\Delta\Phi$ denotes almost parallel straight lines, while $\Delta\rho$ is the distance between 2 parallel lines. The $\Delta\Phi$ and $\Delta\rho$ values are subsequently classified by relevance functions which, based on the knowledge of the field geometry, will filter out lines whose relative orientation and/or distances do not match the actual field relative orientation and/or distances. The remaining lines are correlated, in Hough space, with the geometric field model, so as to obtain the robot posture estimate. An additional step must be taken to disambiguate the robot orientation. In the application to soccer robots, the ambiguity is due to the soccer field symmetry. The goal colors are used to remove such ambiguity and to detect situations where the localization values obtained are not trustable.

Currently, an efficiently coded version of the algorithm is used by each of the ISocRob team robots to obtain its self-localization during a game every second. The algorithm runs in parallel with all the other processes and can compute self-localization in about 13 ms on the average, using Intel IPP library.

The knowledge of each robot localization is useful for individual robot navigation, but it is also used by the robot to share information with its teammates regarding team postures and ball location.

3.3 Multi-Sensor Guidance with Obstacle Avoidance

The ability to navigate at relatively high speeds through an environment cluttered with static and dynamic obstacles is a crucial issue for a mobile robot. Most robotic tasks require a robot to move to target postures adequate to carry out its planned activities. In robotic soccer, relevant activities include facing the opponent goal with the ball in between or covering the team goal by positioning itself between the ball and the goal, while avoiding the field walls and the other (stopped or moving) robots. Also relevant is the capability to move towards a given posture while avoiding obstacles and keeping the ball (also known as dribbling). A guidance control method for non-holonomic (differential drive) vehicles, using odometry, regularly reset by the vision-based self-localization algorithm described before, was first introduced in (Marques, C., and Lima, P., 2002). The vehicle uses a sonar ring for obstacle avoidance.

An alternative guidance method has been introduced in (Damas, B., et al, 2002), consisting of a modified potential fields method for robot navigation, especially suited for differential-drive non-holonomic mobile robots. The potential field is modified so as to enhance the relevance of obstacles in the direction of the robot motion. The relative weight assigned to front and side obstacles can be modified by the adjustment of one physically interpretable parameter. The resulting angular speed and linear acceleration of the robot can be expressed as functions of the linear speed, distance and relative orientation to the obstacles. This formulation enables the assignment of angular and linear velocities for the robot in a natural fashion. Moreover, it leads to an elegant formulation of the constraints on angular speed, linear speed and acceleration, that enable a soccer robot to dribble with the ball, i.e., to move while avoiding obstacles and pushing the ball without losing it, under severe restrictions to ball holding capabilities. It is shown that, under reasonable physical considerations, the angular speed must be less than a non-linear function of the linear speed and acceleration, which reduces to an affine function of the acceleration/speed ratio when a simplified model of the friction forces on the ball is used and the curvature of the robot trajectory is small.

3.4 Behavior-Based Architectures

The basic functional architecture of the SocRob team is organized in three levels of team member responsibility, similar to those proposed in (Drogoul, A., and Collinot, A., 1998): individual, which is responsible for all functionalities that involve only one robot; relational, which is responsible for the relationships between the robot and its teammates; and organizational, which is responsible for the strategic decisions that involve the team as a whole. Behaviours are classified according to this division: we consider organizational, relational and individual behaviours.

Since behaviours are externally displayed and emerge from the application of certain operators, the functional architecture can also be viewed from an operator standpoint, with three levels of decision:

- **Team Organization Level**, where, based on the current world model, a strategy (i.e., what to do) is established, including a goal for the team. This level considers issues such as modelling the opponent behaviour to plan a new strategy. Strategies may simply consist of enabling a given subset of the operators at each robot, in result of role assignments to each team member. In robotic soccer, basic roles can be Goalkeeper, Defender, Attacker and Full Player (both defender and attacker). Only the captain robot will have the organization level enabled. Should the captain "die", the next robot in a pre-specified list will have its organization level enabled and become the captain.

- **Behaviour Coordination Level**, where switching among operators, both individual and relational, occurs so as to coordinate behaviour execution, at each robot and among the team robots, towards achieving the team goal, effectively establishing the team tactics (i.e., how to do it). Both a finite state automaton or a rule-based system were used to implement this level, but other alternative representations are possible, such as Petri nets.
- **Behaviour Execution Level**, where primitive tasks run and where they interface the sensors, through the blackboard, and the actuators, through the navigation functions at each robot. Primitive tasks are linked to each other to implement an operator. Currently, every operator (representing a given behaviour) is implemented as a finite state automaton whose states are the primitive tasks and transitions are associated to logical conditions on events that are detected by the system. Behaviours can be individual, if their corresponding operators run in one robot only, or relational, if two or more robots are running operators that are coordinated through commitments and synchronisation messages to achieve a common goal (e.g., to pass a ball, to avoid moving simultaneously towards a ball, to cover a field region while the teammate advances in the field through role exchanges). Any team member may have relational operators available. Each operator has a pre-conditions set and, when this set is satisfied, establishes communications with the relational operator(s) of designated teammates, asking them to start a negotiation process which may end up in a coordinated action among this temporary sub-team. As a result, a relational behaviour is displayed.

The software architecture is the practical implementation of the functional architecture, which could be done in any programming language and using different software technologies. In the SocRob project, the software architecture was defined based on three essential concepts: micro-agents (μA for short), blackboard and plugins.

Inspired by the idea of Society of Agents, proposed by Minsky (Minsky, M., 1988), each functional module of the SocRob architecture was implemented by a separate process, using the parallel programming technology of threads. In this context a functional module is named μA. In the current implementation of the SocRob architecture there are nine different threads, but only the four most important ones are mentioned here: μA Vision, responsible for processing the data acquired from the cameras, μA Fusion, which fuses information concerning the same object from different sensors, μA Machine, responsible for deciding which behavior should the robot display, and μA Control, responsible for the execution of the corresponding operator.

The concept of threads was chosen to improve module performance and simplify the information passing among the threads. This was accomplished by the blackboard concept (memory space shared by several threads), further sophisticated here by the development of a distributed blackboard, in what information availability is concerned. Instead of being centralized in one agent, the information is distributed among all team members and communicated when needed.

As mentioned before, the decision making involved for each agent is twofold: which behavior should be displayed, and how the operator which displays such behavior is executed. This separation between behavior decision and operator execution allows the μA Machine, the one responsible for behavior decision, to work with abstract definitions of behaviors, and choose among them without knowing details about their execution. So, new operators could be easily added and removed without affecting the existing ones, and these can also be easily replaced by others with the simple restriction of maintaining the name. This was accomplished using the concept of plugin, in the sense that each new operator is added to the software architecture as a plugin, and therefore the μ A Control can be seen as a multiplexer of plugins. Examples of already implemented operators are: dribble, score, go, standby, to name but a few. The same idea of plugins was also used for the μ A Vision, as each particular functionality related to vision data is defined as a different plugin, and multiplexed by the μ A Vision (e.g., a plugin for the front camera, a plugin for the up camera, a plugin for the self-localization algorithm, etc.).

The individual operators have been implemented as state machines, where the states represent primitive tasks, while the arcs between states (if any) are traversed upon the validation of given logical conditions over events (e.g., see ball, distance < x). The relational operator state machines could be defined similarly, but events include synchronization signals between the state machines running in the sub-team robots.

However, the way the functional architecture was conceptualized allows the implementation of these operators and the switching among them using different approaches, as for example AI production systems. So, in order to have a more abstract way to deal with operator/behaviour switching, the □A Machine has been implemented using a distributed decision-making architecture supported on a logical approach to modeling dynamical systems (Reiter, 2001), based on situation calculus, a first order logic dialect.

This architecture includes two main modules (see Fig. 4):

- a basic logic decision unit, and
- an advanced logic decision unit.

Both run in parallel; the former intends to quickly suggest, using simple logical decision rules, the next operator/behaviour to be executed, whereas the latter uses more sophisticated reasoning tools (situation calculus) capable of planning, learning and decision-making, both for individual and cooperative (teamwork) situations. This configures an hybrid architecture where the basic (reactive) unit only controls the robot if the advanced (deliberative) unit takes too long to make a decision, assuming a situation urgency evaluation. A partial implementation of this architecture, the basic logic decision unit, was already performed using Prolog (Arroz, M., et al, 2004). Its modeling convenience allowed the quick development of different roles for field players (Attacker, Defender, Full-Player), as well as dynamic role change between field players (defenders switch with attackers, depending on who is in a better position to get the ball).

The advanced (deliberative) unit, Advanced Logic Based Unit, has been developed using an action programming language called Golog Golog (Levesque, H., et al, 1997) and it is based on situation calculus. This unit is responsible to determine plans (sequences of operators/behaviours) that allow the team to achieve something (like scoring on the opposite goal). Situation calculus is an extension to first-order logic, specially suited to handle dynamic worlds. The changes in the world are the results of actions that have pre-conditions and effects.

Our objective is to develop a tool capable of planning and performing task control execution in a distributed environment. To do so we assume that: the agents (robots) can generate, change and execute plans; a plan can be generated, and executed by one or more agents; decisions over the generated plans are based on hypotheses, i.e., assumptions over future states that cannot be guaranteed; and the agents have the capacity to communicate among them, and share information about plans or environment states.

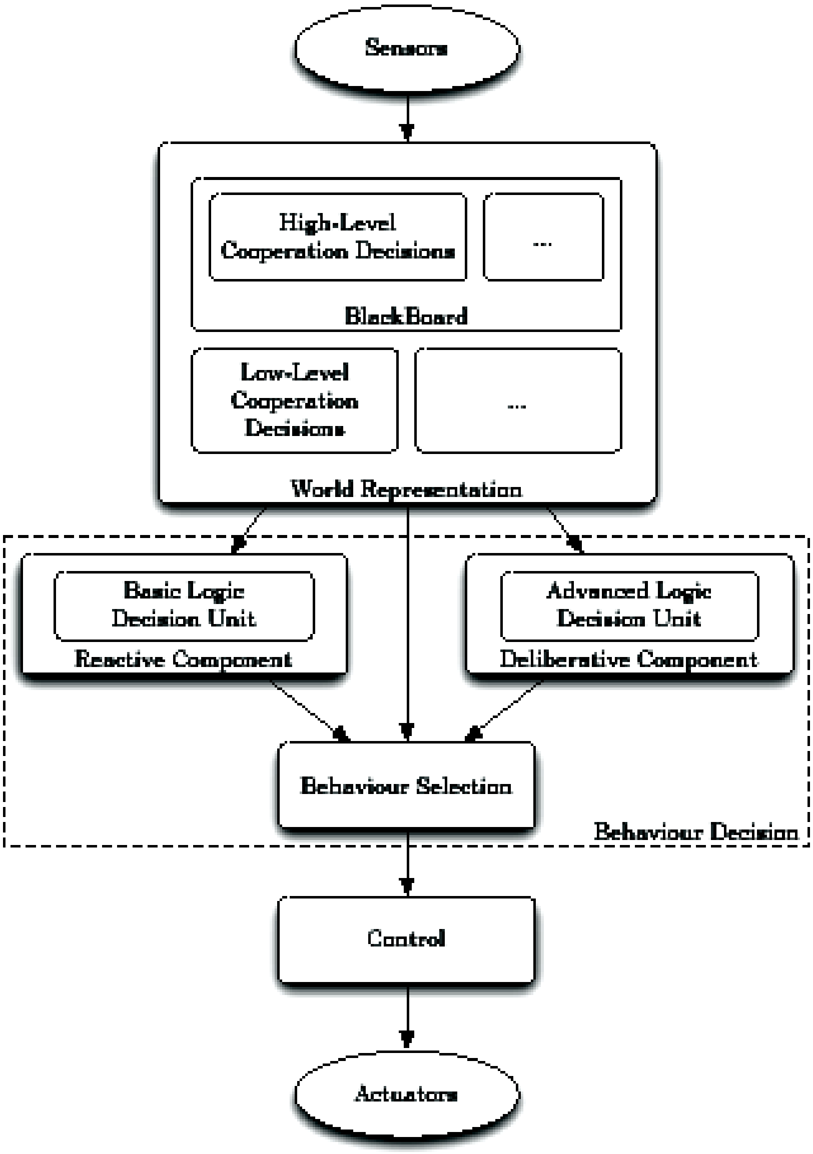


Figure 4. Hybrid architecture of the μ A machine

3.5 Relational Operators

Another recent topic in the project research is the design and implementation of relational behaviors, where teamwork between two or more robots is required to perform a certain task, like a ball pass (Vecht, B., & Lima, P., 2004). These behaviors have a general formulation based on Joint Commitment Theory (Cohen, P. R., & Levesque, H. J., 1991), and use the navigation methods already developed in the project.

Currently, the robots are capable of committing to a relational pass behavior where one of the robots is the kicker and the other the receiver. Should any of the robots end the commitment (e.g., due to a failure to achieve the behaviour goal or after succeeding in it), it inform the other(s) robot(s) involved in the relational operator, so that they can take the appropriate actions to end the commitment and switch to another operator.

One cooperation mechanism, implemented in 2000, consists of avoiding that two or more robots from the same team attempt to get the ball. A relational operator was developed to determine which robot should go to the ball and which one(s) should not. In the current implementation, each robot that sees the ball and wants to go for it uses a heuristic function to determine a fitness value. This heuristic penalizes robots that are far from the ball, are between the ball and the opposite goal and need to perform a angular correction to center the ball with its kicking device. Each robot broadcasts its own heuristic value, and the robot with the smallest value is allowed to go for the ball whereas the others execute a Standby behavior.

3.6 Task Planning and Allocation

Though not tested yet in real robots, formal work on Stochastic Discrete-Event Systems modeling of a multi-robot team has been recently carried out within the project with interesting results (Damas, B., & Lima, P., 2004).

The environment space and each player (opponent and teammate) actions are discretized and modeled by a Finite State Automaton (FSA) representing a 2 vs 2 players game. Then, all FSA are composed to obtain the complete model of a team situated in its environment and playing an adversarial game. Controllable (e.g., shoot_p1, stop_p2) and Uncontrollable (e.g., lost_ball, see_ball) events (i.e., our robots actions) are identified and exponential distributions are assigned to their inter-event times. Dynamic programming is applied to the optimal selection of the controllable events, with the goal of minimizing the cost function

$$\min_{\pi}\left[\int_{0}^{\infty} C\left[X(t), u(t)\right] dt\right] \tag{3}$$

where π is a policy, X(t) the game state at time t, and u(t) is a controllable event, with the cost of unmarked states equal to 1, and all the other states have zero cost. If the only marked states are those where a goal is scored for our team, and there are no transitions from marked to unmarked states, this method obtains the minimum (in a stochastic sense) time to score a goal for our team, constrained by the opponent actions and the uncertainty of our own actions. Some of the chosen actions result in cooperation between the two robots of the team.

4. Problems to be Addressed

Naturally, several interesting problems remain to be tackled and solved within the project research. We will only mention the currently most important ones.

Behavior Modeling: A consistent model for individual and relational behaviors, or more precisely, for the operators implementing them, is required to provide a systematic methodology for behavior synthesis and analysis. FSA have been used for this purpose up to now. They have the advantage of the availability of several tools for analysis and synthesis in the literature (Cassandras, C. G., & Lafortune, S, 1999), but suffer from limited modeling capabilities. Petri nets (Cassandras, C. G., & Lafortune, S, 1999) extend the modeling capabilities of FSA and provide a more convenient modeling methodology starting from the identification of the system components and events. A wide range of analysis (e.g., concerning boundedness, liveness, stochastic and deterministic time) and synthesis (e.g., concerning admissible marked languages) tools is also available, and the non-decidability of some analysis problems can be overcome with no significant expenses. Furthermore, modularity and system design can be achieved by interconnecting several sub-systems, each modeled as a Petri net. This is particularly convenient to model relational behaviors, where more than one teammate is involved. So, Petri nets are being investigated as an alternative tool for behavior modeling. Behavior switching can also be modeled as discrete-event systems supervision, for which there are results available regarding FSA and Petri nets. Production systems also have modeling characteristics that make them suitable for this purpose. However, further work must be done to study its design and analysis properties.

Distributed Planning: The available behaviors among which switching is possible are currently designed "by hand". However, a more appropriate approach would be to develop a planner capable of periodically (or when invoked) analyzing the world state and providing a new set of individual and relational behaviors appropriate for the current conditions. A suitable approach should be the continuous interleaving of plan generation and execution. Task allocation among the team robots and distributed world modeling are relevant issues to be further investigated under this topic. Another relevant issue under investigation is the translation into behaviour models (e.g., FSA, Petri nets) of logical specifications for a mission to be carried out by a team of robots.

Cooperative Learning: One possible way of designing plans which continuously adapt to new situations and are fine tuned to the actual surrounding environment is to use reinforcement learning (RL) algorithms, especially those which guarantee convergence properties (Sutton, R., and Barto, A., 1998). However, learning is usually slow. An envisaged approach that overcomes this problem is to provide plans with alternative paths among which the RL algorithms can learn to switch over time. Cooperative learning arises when a robot takes its decisions from information learned and provided to it by its teammates.

Control as a Game: Modern views of control state the control problem as a game against an adversary (i.e., the disturbances). In the particular case of soccer, there is an actual opponent whose modeled behavior, once estimated (e.g., using Hidden Markov Models), can be used as information for game-playing algorithms, as part of the planning process.

5. Conclusions

This paper described the SocRob project (on the development of methodologies for analysis, design and implementation of multi-robot cooperative systems), its objectives, past, current and intended future work. One interesting feature of the project is that it enables different approaches to the solution of the problem at hand. This naturally motivates competing research approaches, as well as research on analysis methods to compare the different results.

Furthermore, the project fosters education in AI and Robotics related topics, because so many issues must be solved to handle the overall problem. Students from different levels (undergraduate, graduate, post-doctorate) can get involved at different difficulty levels and accomplish project sub-goals. They also learn how to accomplish teamwork under hard time deadlines. The SocRob project has involved so far 20 undergraduate and 8 graduate (5 MSc and 3 PhD) students, besides 2 doctorates who have been supervising the project. All these students have participated regularly in RoboCup - The World Cup of Soccer Robots, since 1998. We believe that RoboCup is a very attractive long-term scientific challenge that brings together people from several different scientific fields in an exciting fusion of research, education and science promotion which are actually the driving forces of our project too.

Some of the methodologies developed within the project, namely its software and functional architectures, have been applied meanwhile to other projects, such as an European Space Agency project on Formation Guidance and Navigation of Distributed Spacecraft, and a Cooperative Navigation for Rescue Robots project currently underway at ISR/IST. The project team is now developing new robots, in the framework of a national research project, in partnership with two Portuguese small companies. These new robots are omnidirectional, with a new modular construction, so that it will be easily modified, e.g., the up camera module can switch between a catadioptric system and a stereo image system.

The new robots will also incorporate a controlled kicker mechanism, so that one can choose the kicking force, using an electromechanical solution with a DC motor pulling a spring and an infrared sensor to measure the pulled distance, both coupled to the kicking device. In order to make new and more complex behaviors and for ball handling, there is a ball reception mechanism, that will allow the implementation of ball passes behaviors. Two new sensors will be used: a rate-gyro for angular velocity measurements, and an optical mouse to track the robot position in the field. Both will provide data to be fused with odometry and vision-based self-localization, so as to improve navigation.

6. References

Balch, T. and Parker, L. (eds), Robot Teams: From Diversity to Polymorphism, AK Peters, 2002.

Canudas de Wit, C. and Siciliano B., and Bastin G. (Eds) (1996), Theory of Robot Control, CCE Series, Kluwer

Cassandras, C. G. and Lafortune, S. (1999), Introduction to Discrete Event Systems, Kluwer Academic Publishers

Cohen, P. R., and Levesque, H. J. (1991), "Teamwork". Nous, Vol 35, pp. 487-512

Damas, B. and Lima P. and Custódio (2002), “A Modified Potential Fields Method for Robot Navigation Applied to Dribbling in Robotic Soccer”, Proceedings of RoboCup-2002 Symposium, Fukuoka, Japan

Damas, B., and Lima, P., (2004), "Stochastic Discrete Event Model of a Multi-Robot Team Playing an Adversarial Game", 5th IFAC/EURON Symposium on Intelligent Autonomous Vehicles - IAV2004, Lisboa, Portugal

Decker, K. S., and Lesser, V. R. (1995), “Designing a Family of Coordination Algorithms”, Technical Report No. 94-14, Department of Computer Science, University of Massachussets, Amherst, MA01003

desJardins, M. E., and Durfee, E. H., and Ortiz Jr, C. L., and Wolverton, M. J. (1999), “A Survey of Research in Distributed, Continual Planning”, AI Magazine, Winter, pp. 13-22

Drogoul, A., and Collinot, A. (1998), “Applying an Agent-Oriented Methodology to the Design of Artificial Organizations: A Case Study in Robotic Soccer”, Autonomous Agents and Multi-Agent Systems Journal, Vol. 1, pp. 113-129

Ferber, J. (1999), Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence, Addison-Wesley

Gonzalez, R., and Woods, R. (1992), Digital Image Processing, Addison-Wesley

Levesque, H., and Reiter, R., and Lesprance, Y., and Lin, F., and Scherl, R. (1997), “Golog: A Logic Programming Language for Dynamics Domains”. Journal of Logic Programming

Marques, C., and Lima, P. (2001), “A Localization Method for a Soccer Robot Using a Vision-Based Omni-Directional Sensor”, RoboCup-2000: Robot Soccer World Cup IV, P. Stone, T. Balch, G. Kraetzschmar (Eds.), Springer-Verlag, Berlin

Marques, C., and Lima, P. (2002), “Multi-sensor Navigation for Soccer Robots”, RoboCup-2001: Robot Soccer World Cup V, A. Birk, S. Coradeschi, S. Tadokoro (Eds.), Springer-Verlag, Berlin

Minsky, M. (1988), The Society of Mind, Touchstone Publishers

Pinheiro, P., and Lima, P. (2004), "Bayesian Sensor Fusion for Cooperative Object Localization and World Modeling", 8th Conference on Intelligent Autonomous Systems (IAS-8), Amsterdam, The Netherlands, May 2004.

Pires, V., Arroz, M., and Custódio, L. (2004), “Logic Based Hybrid Decision System for a Multi-robot Team”, Proceedings of the 8th Conference on Intelligent Autonomous Systems (IAS-8), Amsterdam, The Netherlands.

Pires, V., Arroz, M., Lima, P., Ribeiro, M. I., and Custódio (2004), L., “Distributed Deliberative Decision System for a Multi-Robot Team”, Proceedings of the ROBOTICA 2004 Symposium, Porto, Portugal, April 2004.

Reiter, R. (2001), Knowledge in Action. MIT Press

Sutton, R., and Barto, A. (1998), Reinforcement Learning, MIT Press, Cambridge, MA

Vecht, B., Lima, P., 2004, “Formulation and Implementation of Relational Behaviors for Multi-Robot Cooperative Systems”. Proceedings of RoboCup 2004 Symposium, Lisbon, Portugal

RoboCup is a Stage which Impulse the Research of Basic Technology in Robot

Cheng Xian-yi & Xia De-shen

1. Introduction

RoboCup is an international joint project to promote Artificial Intelligence (AI), robotics, and related field. It is an attempt to foster AI and intelligent robotics research by providing a standard problem where wide range of technologies can be integrated and examined. RoboCup chose to use soccer game as a central topic of research, aiming at innovations to be applied for socially significant problems and industries. The ultimate goal of the RoboCup project is by 2050, develop a team of fully autonomous humanoid robots that can win against the human world champion team in soccer (Fig 1).

In order for a robot team to actually perform a soccer game, various technologies must be incorporated including: design principles of autonomous agents, multi-agent collaboration, strategy acquisition, real-time reasoning, robotics, and sensor-fusion. RoboCup is a task for a team of multiple fast-moving robots under a dynamic environment. RoboCup also offers a software platform for research on the software aspects of RoboCup (Burkhard02).

One of the major applications of RoboCup technologies is a search and rescue in large-scale disaster. RoboCup initiated RoboCup rescue project to specifically promote research in socially significant issues.

In the next section we will introduce the origin, organization and leagues of RoboCup. Section 3 we will discuss the relative technology in RoboCup.

Figure 1. Soccer racing in the future

2. The Origin, Organization and Leagues of Robocup

The concept of RoboCup was first introduced by professor of Alan Mackworth in 1993. The main goal of RoboCup is to propose a challenged research issue to develop robotic. Following a two-year feasibility study, in August 1995, an announcement was made on the introduction of the first international conferences and football games. Now RoboCup Soccer is divided into the following leagues: Simulation league(2D,3D), Small-size robot league (f-180), Middle-size robot league (f-2000), Four-legged robot league, Humanoid league.

In July 1997, the first official conference and games were held in Japan. The annual events attracted many participants and spectators.

2.1 RoboCup 2D-Simulation League

The RoboCup 2D-simulation league uses a simulator called the Soccer Server to do the soccer simulation. The Soccer Server provides a standard platform for research into multi-agent systems. The Soccer Server simulates the players, the ball and the field for a 2D soccer match.22 clients (11 for each team) connect to the server, each client controlling a single player. Every 100ms the Soccer Server accepts commands, via socket communication, from each client. The client sends low level commands (dash, turn or kick) to be executed (imperfectly) by the simulated player it is controlling. Clients can only communicate with each other using an unreliable, low bandwidth communication channel built into the Soccer Server. The Soccer Server simulates the (imperfect) sensing of the players, sending an abstracted (objects, e.g. players and ball, with direction, distance and relative velocity) interpretation of field of vision to the clients every 150ms. The field of vision of the clients is limited to only a part of the whole field. The Soccer Server enforces most of the basic rules of (human) soccer including off-sides, corner kicks and goal kicks and simulates some basic limitations on players such as maximum running speed, kicking power and stamina limitations (Bom99).

An extra client on each team can connect as a "coach", who can see the whole field and send strategic information to clients when the play is stopped, for example for a free-kick. The Soccer Monitor (Fig 2) connects to the Soccer Server as another client and provides a 2D visualization of the game for a human audience. Other clients can connect in the same way to do things like 3D visualization, automated commentary and statistical analysis.

There are no actual robots in this league but spectators can watch the action on a large screen, which looks like a giant computer game. Each simulated robot player may have its own play strategy and characteristic and every simulated team actually consists of a collection of programmers. Many computers are networked together in order for this competition to take place. The games last for about 10 minutes, with each half being 5 minutes duration.

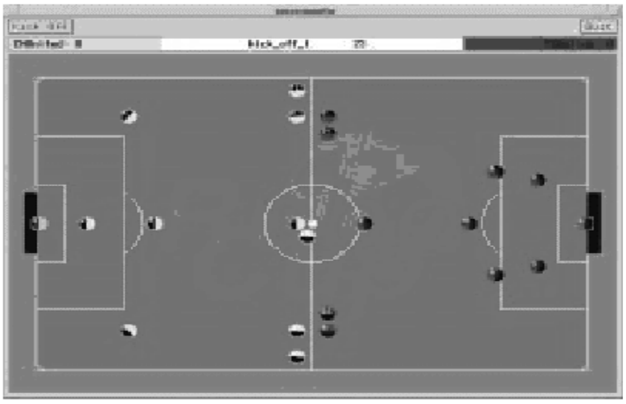


Figure 2. RoboCup 2D-Simulation league

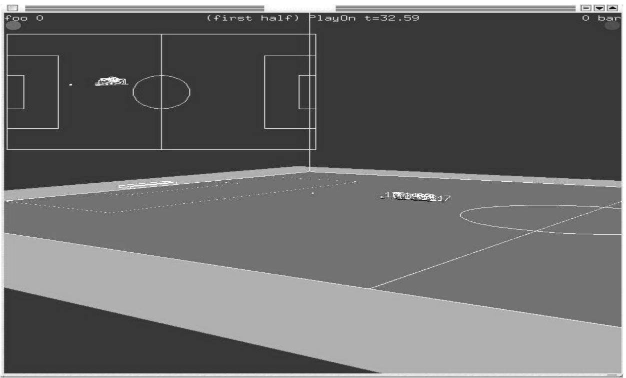


Figure 3. RoboCup 3D-Simulation league

2. 2 RoboCup 3D-Simulation League

The 3D competition makes use of the simulator that is based on the simulation system introduced at the RoboCup 2003 symposium and the spades simulation middleware system introduced at the RoboCup 2002 symposium. It can be downloaded from source forge (Fig 3). One of the goals for future 3D soccer competitions is to have simulated robots with articulated bodies, for example like humanoid robots. Prior to compiling and installing the rcssserver3D, you need to install a few software packages. You can compile and install rcssserver3D in two different ways, a "light" installation and the full installation. With the full installation, you get an additional library (called kerosin), which is useful to visualize objects nicely, especially articulated objects (this are objects consisting of more than one geometry linked with joints). These features are not (yet) required for the soccer simulation. The light installation, which is the default, you get a not so fancy OpenGL visualization. To enable the full installation, pass the "--enable-kerosin" flag to the `configure' shell script. For the generic installation instructions, see the text below the specific instructions here.

Required libraries for the default installation:

(1) spades
- working versions: 1.0, older versions may also work;
- get it at: http://sourceforge.net/projects/ spades-sim;
- description: agent middleware, handles timing issues and networking;
- additional info: you need a recent version of expat for spades.

(2) ruby
- working versions: 1.8.0 or newer;
- get it at: http://www.ruby-lang.org/;
- description: scripting language;
- additional info: if you compile ruby yourself, configure with enable-shared.

(3) boost
- working versions: 1.30.2, 1.31.0;
- get it at: http://www.boost.org/;
- description: c++ programming extensions.

(4) ode
- working versions: 0.039;
- -get it at: http://sourceforge.net/projects/ open- de;
- -descriptions: for simulating articulated rigid body dynamics.

(5) OpenGL, GLUT

You need the OpenGL and GLUT headers for the visualization. This may be dependent on your graphics card. (GLUT is the OpenGL Utility Toolkit).
- part for example of XFree86-Mesa-devel;
- you should get it with your linux distro.

2.3 Small-size Robot League

The field of play must be rectangular. The dimensions include boundary lines. Length: 4900mm, Width:3400 mm. A small-size RoboCup takes place between two teams of five robots each (Fig 4).

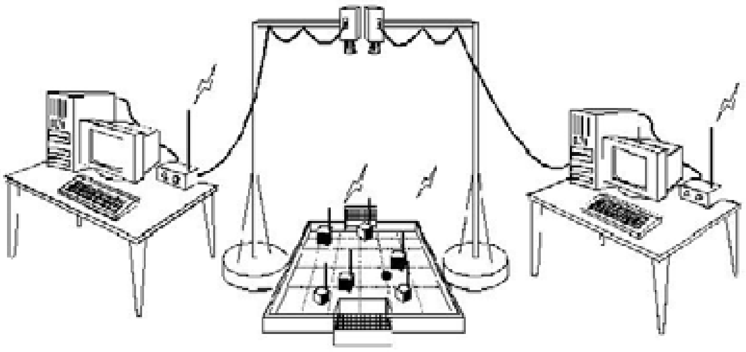


Figure 4. Small-size robot league

Each robot must conform to the dimensions as specified in the F180 rules: The robot must fit within a 180mm diameter circle and must be no higher than 15cm unless they use on-board vision. The robots play soccer on a green carpeted field that is 2.8m long by 2.3m wide with an orange golf ball. Robots come in two flavors, those with local on-board vision sensors and those with global vision. Global vision robots, by far the most common variety, use an overhead camera and off-field PC to identify and track the robots as they move around the field. The overhead camera is attached to a camera bar located 3m above the playing surface. Local vision robots have their sensing on the robot itself. The vision information is either processed on-board the robot or is transmitted back to the off-field PC for processing. An off-field PC is used to communication referee commands and, in the case of overhead vision, position information to the robots. Typically the off-field PC also performs most, if not all, of the processing required for coordination and control of the robots. Communications is wireless and typically uses dedicated commercial FM transmitter/receiver units although at least one team has used IRDA successfully.

2.4 Middle Size League

Two teams of 4 mid-sized robots with all sensors on-board play soccer on a field. Relevant objects are distinguished by colors. Communication among robots (if any) is supported on wireless communications. No external intervention by humans is allowed, except to insert or remove robots in/from the field.

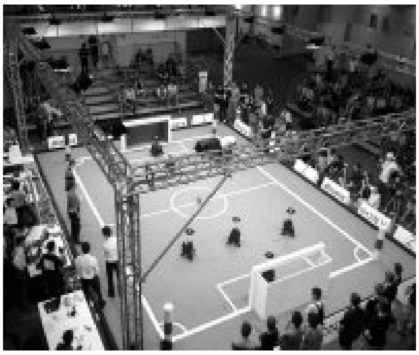


Figure 5. Middle Size League

2. 5 The Four-Legged League

In The Four-Legged League, participating teams have to use robots specified by the Competition Committee without any modification on its hardware.
In 2004 there are choices of either using
-Sony Entertainment Robot AIBO ERS-210/210A, or
-Sony Entertainment Robot AIBO ERS-7, or
-A combination of both in the team
The four main technical issues associated with the SSL are the following:

Robust color processing and color tracking. The lighting at tournament halls is very irregular; there are shadows and unpredictable variations during a game. The software has to surmount these difficulties while processing video frames provided by inexpensive cameras. In recent years, most good teams have solved these issues, and we do not see them losing the robots or the ball.
Robust mechanical design. A robot able to play at a good level in the SSL must be fast (1-2 m/s maximal speed) and able to resist strong collisions. Typically, SSL robots can fall from a table and continue playing. There has been a new emphasis in mechanical design during the last two years with the introduction of such innovations as omni directional drive (Cornell 2000) and dribbling bars that allow robots to control the ball and pass it (Cornell 2001).
Robust wireless communications. This might be considered the single most important unsolved issue in the SSL. Most teams use the same RF chips and this has led to significant interference problems in the past. Tournaments have become too long because it is very difficult to schedule simultaneous games. A solution such as WaveLan cards or Bluetooth

modules will be explored in the future.
Good programming of robot behavior. It can be safely said that most teams in the SSL have adopted a pure reactive design with simple strategic goals. The fact that the field of play is too small relative to the size of the robots means that it does not pay to compute too complicated strategies. The horizon of most systems is just a few frames into the future, since the robots are so fast relative to the size of the field. Thus, enlarging the field has to become a major research issue if more sophisticated strategies are to be programmed.

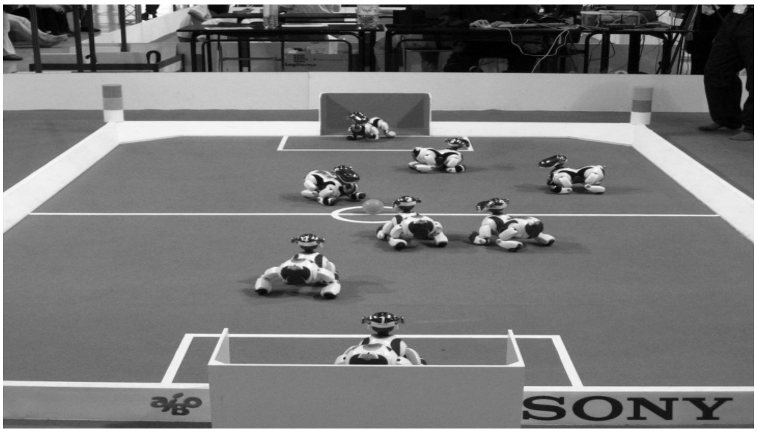


Figure 6. 4 legged league

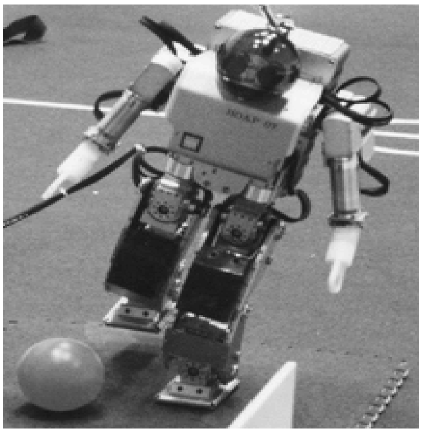


Figure 7. Humanoid league

2.6 Humanoid League

Humanoid robots show basic skills of soccer players, such as shooting a ball, or defending a goal. Relevant objects are distinguished by colors. External intervention by humans is allowed, as some of the humanoid robots are tele-operated.

3. Viewing a Soccer Game as a Multi-Agent Environment

A soccer game is a specific but very attractive real time multi-agent environment from the viewpoint of distributed artificial intelligence and multi-agent research. If we regard a soccer team as a multi-agent system, a lot of interesting research issues will arise.
In a game, we have two competing teams. Each team has a team-wide common goal, namely to win the game. The goals of the two teams are incompatible. The opponent team can be seen as a dynamic and obstructive environment, which might disturb the achievement of the common team goal. To fulfill the common goal, each team needs to score, which can be seen as a sub-goal. To achieve this subgoals, each team member is required to behave quickly, flexibly, and cooperatively; by taking local and global situations into account.
The team might have some sorts of global (team-wide) strategies to fulfill the common goal, and both local and global tactics to achieve subgoals. However, consider the following challenges:
-the game environment, i.e. the movement of the team members and the opponent team, is highly dynamic.
-the perception of each player could be locally limited.
-the role of each player can be different.
-communication among players is limited; therefore, each agent is required to behave very

flexibly and autonomously in real-time under the resource bounded situation.

Summarizing these issues, a soccer team can be viewed as a cooperative distributed real-time planning scheme, embedded in a highly dynamic environment. In cooperative distributed planning for common global goals, important tasks include the generation of promising local plans at each agent and coordination of these local plans. The dynamics of the problem space, e.g. the changing rate of goals compared with the performance of each planner, are relatively large, reactive planning that interleaves the plan generation and execution phases is known to be an effective methodology at least for a single agent to deal with these dynamic problems.

For cooperative plan schemes, there are frequent changes in the problem space or the observation of each agent is restricted locally. There is a trade-off between communication cost, which is necessary to coordinate the local plans of agents with a global plan, and the accuracy of the global plan (this is known as the predictability/responsiveness tradeoff). The study of the relationship between the communication cost and processing cost concerning the reliability of the hypotheses in FA/C, and the relationship between the modification cost of local plans and the accuracy of a global plan in PGP illustrate this fact.

Schemes for reactive cooperative planning in dynamic problem spaces have been proposed and evaluated sometimes based on the pursuit game (predator-prey) (Hiroaki01). However, the pursuit game is a relatively simple game, the environment is basically for the study of a single agent architecture.

We see that a robot soccer game will provide a much tougher, fertile, integrated, exciting, and pioneering evaluation environment for distributed artificial intelligence and multi-agent research.

4. Research Issues for Robocup with Real Robots

In this section, we discuss several research issues involved in realizing real robots for RoboCup.

(1) Design of RoboCup player and their control: Existing robot players have been designed to perform mostly single behavior actions, such as pushing/dribbling/rolling. A RoboCup player should be designed so that it can perform multiple subtasks such as shooting (including kicking), dribbling (pushing), passing, heading, and throwing a ball; which often involves the common behavior of avoiding the opponents. Roughly speaking, there are two ways to build RoboCup players:

- Design each component separately, which is specialized for a single behavior and then assemble them into one.
- Design one or two components that can per form multiple subtasks.

Approach 1 seems easier to design but more difficult to build and vice versa. Since the RoboCup player should move around quickly it should be compact; therefore, approach 2 should be a new target for the mechanical design of the RoboCup player. We need compact and powerful actuators with wide dynamic ranges. Also, we have to develop sophisticated control techniques for as few as possible multiple behavior components with low energy consumption. The ultimate goal of a RoboCup player would be a humanoid type, that can run, kick and pass a ball with its legs and feet; can throw a ball with its arms and hands, and can do heading with its head. To build a team of humanoid type robots currently seems impossible.

(2) Vision and sensor fusion: Visual information is a rich source of information to perceive, not only the external world, but the effects of the robot's actions as well. Computer Vision researchers have been seeking an accurate 3D geometry reconstructing

from 2D visual information, believing in that the 3D geometry is the most powerful and general representation. This could be used in many applications, such as view generation for a video database, robot manipulation and navigation. However, the time-consuming 3D reconstruction may not be necessary nor optimally suited for the task given to the RoboCup player. In order to react to the situation in real time, the RoboCup player quickly needs information to select behavior for the situation, we are not suggesting a special-purpose vision system, just that the vision is part of a complex system that interacts in specific ways with the world. RoboCup is one of these worlds, which would make clear the role of vision and evaluate the performance of image processing which has been left ambiguous in the computer vision field. In addition to vision, the RoboCup player might need other sensing devices such as: sonar, touch, and force/torque, to discriminate the situations that cannot be discriminated from only the visual information nor covered by visual information. Again, the RoboCup player needs the real time processing for multi-sensor fusion and integration. Therefore, the deliberative approaches with rough estimation using multi-sensor system do not seem suitable. We should develop a method of sensor fusion/integration for the RoboCup.

(3) Learning RoboCup behaviors: The individual player has to perform several behaviors, one of which is selected depending on the current situation. Since programming the robot behaviors for all situations, considering the uncertainties in sensory data processing and action execution is unfeasible, robot-learning methods seem promising. As a method for robot learning, reinforcement learning has recently been receiving increased attention with little or no a priori knowledge giving higher capability of reactive and adaptive behaviors (Balch00). However, almost all of the existing applications have been done only with computer simulations in a virtual world, real robot applications are very few(Silvia 99). Since the prominence of the reinforcement learning role is largely determined by the extent to which it can be scaled to larger and complex robot learning tasks, the RoboCup seems a very good platform. At the primary stage of the RoboCup tournament, one to one competition seems feasible. Since the player has to take the opponent's motions into consideration, the complexity of the problem is much higher than that of simple shooting without an opponent. To reduce the complexity, task decomposition is often used. Fredrik proposed a method for learning a shooting behavior avoiding a goal keeper (Fredrik00). The shooting and avoiding behaviors are independently acquired and they are coordinated through the learning. Their method still suffers from the huge state space and the perceptual liaising problem, due to the limited visual field. Kum proposed a reactive deliberation approach to the architecture for real time intelligent control in a dynamic environment (Kum99. He applied it to a one to one soccer-like game. Since his method needs global sensing for robot positions inside the field, it does not seem applicable to the RoboCup that allows the sensing capability only through the agents (see the rule section). At the final stage, a many-to-many competition is considered.In this case, collective behaviors should be acquired. Defining all the collective behaviors, as a team seems infeasible, especially, the situations where one of multiple behaviors should be performed. It is difficult to find a simple method for learning these behaviors, definition of social behaviors. A situation would not be defined as the exact positions of all players and a ball, but might be perceived as a pattern. Alternatives, such as "coordination by imitation," should be considered.

In addition to the above, the problems related to the RoboCup such as task representation and environment modeling are also challenging ones. Of course, integration of the solutions for the problems mentioned above into a physical entity is the most difficult one.

5. Relative Technology in Robocup

The robot football game is taken on by hardware or imitated robot human. The rule is similar to the true human football game. The research of robot football match taken by hardware is concerned with computer, automatic control, sensing technology, wireless communication, precise mechanism, imitated materials and numerous forward-position researches and synthesizes integration. Imitated robot football game is carried on the standard software platform and it fully embodies the technologies of control, communication, sensing and some other aspects. The key points of the researches are some advanced functions, such as cooperation in the system, dynamic decisions, timely plans, the learning of machine and some hot points in the current artificial intelligence. Therefore, in the realm of international artificial intelligence, robot football is regarded as a standard problem in the future 50 years, just as the international chess games between human and computer.

The robot football game can do benefit to apply the theories of AI to practice. It also can help to examine the new thoughts, new techniques, and promote the related development of technology. A series of new techniques used by Robot football games will do favor to the development of social economy and culture. Robot football games are not only a kind of front competition with high techniques, but also provide amusement, enjoyment and incentive, which the true game provides. We can anticipate that this activity will produce tremendous market's need and new industrial opportunities, and will bring inestimable utilities of economy and society.

The target of the research of RoboCup is to provide a test platform for the distributed system. In a specific way, it includes the research targets as follows:

- The posture of robot. Now the robot uses wheel and bedrail, the human player don’t play with it in court. So we must build the robot like human such as gesture, structure and weight.

- The body of robot. If the robot is full of iron and plastic, people are afraid of touch in it. So the robot must own the muscle and can collide with people.

- The energy of robot. Now the soccer robot’s power is battery, but can only use few minutes. in the future, the soccer robot must run and move in 45-50 minutes, that means the battery must has little volume, the light weight and full of power.

- The skill of robot. Now the two-logger robot can move in stair. The best soccer robot is four-legged dog of SONY corporation. After 50 years, the robots must can run, move jump, shoot, dribble like human being. People can do, so the robots can do.

- Intelligence of robot. The high level player plays with ball using their brain, so the intelligence of star must high-class. In 1997,IBM’s deep blue beat down Kasparov , but IBM use 16 RISK6000.so in the future, the micro computer in soccer robot must very well.

- The sense of robot. The sense parts are arranged at will. for example, it can own six eyes, use sonar and wireless communication network, now the tech of sense can not solve the comprehension of image, the power of touch and the function and efficiency of inside sensor. So we must solve these problems.

6. The Relationship Between Robocup and MAS

6.1 Agent, MAS and RoboCup

RoboCup is a typical model of MAS(Multi-Agent System). We can take on the eleven robots as eleven Agents. This will involve some related techniques within MAS such as communication and coordination. These techniques are exactly the core in the MAS.

The Agent is an important concept within realm of computer science in recent years. This concept has already been extensively applied to the realm of AI, distributed system of computer science and so forth, and provides a brand-new path for distributed open system. It is regarded as "an important breakthrough of software development once more". In the AI realm, people treat Agent as a computerized entity, which can play a role independently and has the life cycle's calculation under some environments. People also call the systems that be consisted of multi Agents and full of interactions and connections within them as a MAS.

Generally think, the research of Agent can mostly be divided into the Intelligent Agent, MAS and Agent Oriented Programming (AOP). They are not isolated with each other. The Intelligent Agent can be seen as a micro level in the Agents' researches, but AOP and the developing tool or platform of AOP are aimed to serve for MAS researches. Therefore, we can come up to say from a certain degree, the three can be unified to the researches of MAS. This is also consistent with our realistic circumstance because the most of the realistic systems are belong to the MAS.

MAS are the basic technology of RoboCup. RoboCup is a prototype of MAS. Obviously, the technology of RoboCup involves some relative technology in Agents such as coordination, regulation, valid communication, dead lock and some other technologies. These technologies are core in MAS.

For a researcher who wants to attend the RoboCup team, the basic problem is to design a Multi Agent System. The systems can response in time. It also performs a rational action toward the target. Because of the enormous space of soccer game, it is impossible coding all of the possible conditions, situations and behaviors of Agents. Therefore it is vital to make Agents learn strategies. And this is related closely to some technology in the researches of Agents.

These include:

- Machine learning in cooperative environment.
- Construction of MAS; Timely regulation and execution of MAS for team cooperation.
- Modeling opponents.

Some readers maybe ask that MAS is a type of science and technology while football

6.2 Coordination in MAS

Since RoboCup is Multi Agent system, the problem of coordination and regulation between Agents is then the most important to resolve. If could be properly resolved, it would bring advantages to MAS, Such as:

Low the expenses of coordination; speed up the response of system.

Establish the relationship between the utility of the community and individual behavior.

Protect the Agent which with poor ability of reasoning and benefit the Agent which with strong spirit of enterprise.

Settle the problems such as waiting for others' success, coordination within teams, and so on. The problems of coordination and regulation in MAS are actually triggered by the

Agent social activity (the concept of sociality means that in the social activities, the actors will benefit itself in the way that it exchange resources with others which are insignificant to itself). Therefore study the MAS' sociality is then becoming the most basic problem.

Now, the researches in sociality of MAS have already risen to certain step. But because of lacking further analysis of MAS' sociality, the current researches of coordination and regulation still exist some problems to settle, such as:

- Lack popularity.
- It is difficult to weigh the advantages and disadvantages within all kinds of methods.
- Too many researches toward individuals etc.

The coordination methods can be classified as Fig 8(Cheng03). Here, we only discuss the implicit of collective target.

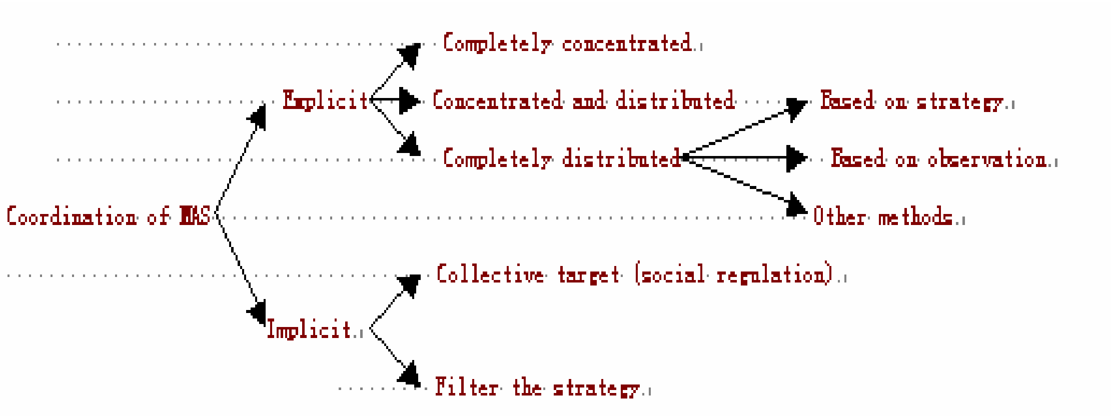


Figure 8. Coordination methods classified

As a method of implicit, the collective target can not only be used to low the coordination expenses, but also be used to embody the MAS sociality, balances the benefits between the individuals and the collective. Collective target is a good breakthrough to the researches of MAS. If the p is a collective target, then: (the c is collective)

- Each Agent i regards p as its own target.
- Each member in c has intention to accomplish the aim with others.
- Each member regards p as its own object because they trust each other.
- They trust each other that if the member Agent i in c reach its target p, then the other members reach target p too.

Now, the research of the collective target contains two kinds of tendencies. One is the research before the collective of Agent come into being other is after it.

6.3 Communication Technique in MAS

The communication problem in MAS researches is also very important. It can judge the advantage and disadvantage of the communications in Agents. If they communicate fluently, it will do benefit to the speed of communications in Agents. Therefore, under the circumstance of having settle the communication problem, we are to discuss the methods of how to make this kind of communication more efficient.

A main reason that the computer is difficult to comprehend the natural language is that it needs of many resources of knowledge. If the side of sending message could mostly understand the side of accepting message, he then would send out the more short and less information. For this reason, validating communications needs to resolve two problems as follows.

The first is the use of the context. If the side of the accepting message can share the same context with the side of the sending message, then the context can decide the meaning of the conversation as a resource of knowledge. The context includes the pronoun, index, current environments, and so on. For example: Tom left home and he would back soon. If the side of the accepting message comprehended the first half sentence 'Tom left home', he then could suppose that 'he' is pointed to 'Tom'. Thus, 'he' can replace 'TOM'. Although this is just a very little economy, lots of economy like this can be added up to raise efficiency. Here, the environment of the contexts can also be called the pre-established of using language, and the pre-established reasonability will do favor to sending out the speech behavior correctly, and make the side of the accepting message comprehend this speech behavior correctly.

The next is to solve the different meanings. Person who uses the natural language can distinguish the expected meanings of the words and phrases of different meanings by kinds of resources of knowledge. But for the Agents, even this kind of information was delivered with no second meanings that also needs of plenty of words and very complicated expressions. There are many kinds of different aspects, primarily include:

(1) Different meanings of words
The same word may have several different meanings, for example: Tom is hot. This not only can mean 'Tom does well in his work', but also can mean 'Tom is very hot'. The different meanings caused by this are called different meanings of words. Resolving it needs of additional knowledge and environments about the subject of a sentence.

(2) Different meanings of sentence construction
Sometimes the sentence can be expressed not only by a method, namely a phrase can be constituted in different ways. The different meanings caused by this are called different meanings of sentence construction. For example, He saw she at home. This not only can suggest that 'he' is doing the action 'see', but also can mean that 'he' is at home. The method to resolve it is to let the side of accepting words knows the position of 'he' or 'she'.

(3) Different meanings of quotations
The repeated usage of the pronoun and other headwords may cause different meanings. The procedure of solving it is concerned with the complicated reasoning of some contents of the sides of sending and accepting.

(4) Different meanings of the use of language

If the common knowledge and the learning of the contexts of the side of accepting is uncertainty, this is called the different meanings of the use of language. The method to solve it is to determine the knowledge of the contexts or make the two sides have common knowledge.

7. Learning in Robocup

The key problem in RoboCup research is how to enhance the agent intelligence by learning, namely how to improve the competitive ability of player. Among many learning approaches, reinforcement learning, which tends to solve the problem how a self-rule agent capable of apperceiving environment can select the optimal actions to attain its goal, has gained widespread attentions. The mechanics of a reinforcement learning can be showed as in Fig.9: let the environment an agent in be state set S, and the arbitrary action set the agent can possibly take is A, if at state S_t perform action a_t, then the agent get a reward of r_t, which is the immediate value of state-action transformation, so as to the ensemble of (S_t, a_t, r_t), the agent's task is to learn the control policy $\pi: S \rightarrow A$, which can maximize the expectation of the reward sum(Cai03) that is when there is $S_0 \xrightarrow{a_0,\gamma_0} S_1 \xrightarrow{a_1,\gamma_1} S_2 \xrightarrow{a_2,\gamma_2} \cdots$ to maximize the $r_0 + \gamma r_1 + \gamma^2 r_2 + \ldots \quad (0 \le \gamma < 1)$.

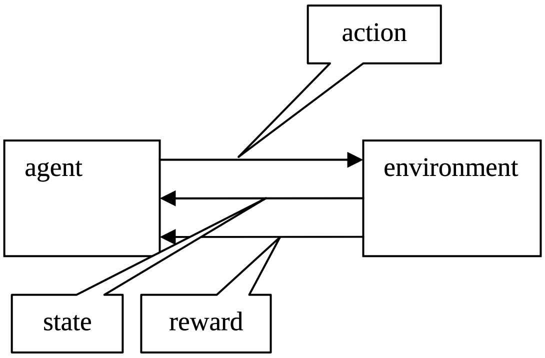


Figure 9. Sketch map of reinforcement learning

As to a RoboCup emulation competition, the competing pattern is a model of Client/Server, in which the environment adopts software platform of SoccerServer standard, participating teams write their own Client programs to simulating a really football game, and every player in Client program can be look upon as agent interacting with the environment. In this game, every agent's actions such as kick, dash, turn, etc. are not only bottom strategies, but also are the key problem to the competition. Taking kick as interaction with the environment namely SoccerServer, and the feedback of ports as the return state, then the task of a player is to design a strategy to maximize the effect of the kick, that is, to control the ball and make good kicks, and at the same time, how to coordinate the whole team to obtain a cooperation learning is also key to winning.

As showed in Fig.9 besides agent and environment, there are four important parts in reinforcement learning include policy, reward function, and value function and environment model as a selective part. Policy is the mapping from some observed state to the action taking which will arrive the state; Reward function defines the goal of reinforcement learning, it maps an apperceived environmental state(or a pair of state and action) to a value which implies the inherent needs of a state, namely a reward. Since the target of single agent in reinforcement learning is to maximize the whole rewards got in the long learning, so the value function shows what is useful in the learning. Because the value of state is the sum of all reward get from current time to the future, and the reward determines the immediate inherent need of environment, so the state value indicates the needs of state followed by possible states in the long future, and the reward that will be obtained in those future states. Since environment model simulates the model action, so using such a model, agent can forecast that how environment will react to agent's action.

Reinforcement learning can solve a large dynamic programming without any a prior. A general form of reinforcement learning is:

(1)Initial the inherent state I of learner as I_0,

(2)Loop:

observe the current state S;

using value function V to select an action $a = V(I, S)$;
perform action a;
let r the immediate reward by performing action a at state s;
using updating function to update the inherent state $I = U(I, s, a, r)$.

Usually, using a table of state and action data, the inherent state I will code the environment information stored by learning algorithm. Abide by command of reinforcement adjusting the current state, updating function maps current inherent state, input, action and reinforcement to a new inherent state, and according to information stored in inherent state, value function V maps inherent state and inputs to an action. There are a few difference of definition of U and V between different reinforcement learning.

Q-learning is just a typical reinforcement learning, in which training sample does not like $< s, a >$, but is the reward of agent's action, so it is difficult to obtain the optimal policy $\pi : S \rightarrow A$ though learning, this can be solved by learning a value function defined on the state and action, by learning the value function, an agent obtain the primal policy. A distinctively good value function is V^*, when $V^*(s_1) > V^*(s_2)$, agent will hope to enter state s_1, because thus can get a larger reward, and of course, agent only can select among actions, not among states (Cheng04).

8. The Significance of Researching Robocup

Thinking carefully, we can suggest more contents and difficult points. We seem to have reasons to deny the imagination of "the battle between human and machine". Because it is unimaginable to reach such achievement today.

But look back to the history, nowadays, there are so many scientific achievements which are unimaginable for the forefathers, aren't there? People will have an unusual eye on the scientific development in 50 years.

It's about half century from the first plane of which the Wright brothers' having trial flight to the successful landing on the moon of Aporo airship. While it's also 50 years from the first computer to computer of "Deep blue" defeating human genius. Now we can see that we should not say "no" in advance for "the battle between human and machine" about 50 years later.

Which we need now is the spirit of innovation, active participation. What we should do is to try our best to improve this process.

It's easy to see that we should innovate more. It contains outstanding progress of artificial life, energy power, material and so on. And it's also contains the great break of many sciences about the project of mechanics, electricity, control, information and computer which are related to the robot. We also need the intersection and combination of multi sciences.

It's the deep meaning of having the research of robot's football.

Although RoboCup is high-tech, only three players' game, there shows some intricate scene. Such as robot bump the wall, two robots badger with each other, and some robots are in the daze, don't concern about ball. People don't understand why the robots' intelligence is not as good as the children.

That is to say, it is not easy to make robot own the human's intelligence-sense, thinking, and action, even the three older children. By 2050, scientists want to develop a team of fully autonomous robots, which can win against the human world champion team in soccer. It is a great goal.

9. Conclusions

This thesis discussed some main technologies in MAS and RoboCup. The aim is to let readers know more about Multi Agent System and cause the Agent-oriented technology mature faster.

There are four steps in the development of programming: procedure oriented Programming, module oriented Programming, object oriented Programming and the last step of Agent oriented Programming. Each process is a more and more abstract procedure, a more and more obscure modeling procedure, till in the end reaches to automatic design of programming. Therefore the emergence of Agent-oriented is inevitable for programming. RoboCup is a stage which impulse the research of basic technology in robot.

Acknowledgement

This work was supported by the Foundation of Doctor Innovation in China under Grant (xm04-35)

10. References

Balch T, Mhybinette (2000), Social Potentials for Scalable Multi-Robot Formation. IEEE International Conf.on Robotics and Automation (ICRA 2000):73-80.

Magnus Boman(1999), Agent Programming in RoboCup'99. AgentLink NewsLetter, (4), November 1999.

Burkhard H D,et al (2002),The Road to RoboCup 2050. IEEE Robotics & Automation Magazine. Jun. 2002: 31-38.

Cai Qing-sheng, Zhang Bo (2003), An agent team reinforcement learning model and its application. (J). Journal of Computer Research and Development. 2003,37(9): 1087-1093. In China.

Cheng Xian-yi (2003), Agent Computing. Haerbin(China): Hei Longjiang science and technology press. 2003.

Cheng Xian-yi et al(2004),.Reinforcement Learning in Simulation RoboCup Soccer. Proceeding of 2004 International Conference on Machine Learning and Cybernetics (ICML2004),in China, IEEE Catalog Number:04EX826. pp244-248.

Fredrik Heintz (2000), RoboSoc a System for Developing RoboCup Agents for Educational Use. Master's thesis, Department of Computer and Information Science, Link.oping university, March 2000.

Hiroaki Y et al (2001), A Distributed Control Scheme for Multiple Robotic Vehicles to Make Group Forma- tions.Robotics and Autonomous systems,2001, 125 –147.

Silvia Coradeschi and Jacek Malec(1999), How to make achallenging AI course enjoyable using the RoboCup soccer simulation system. In RoboCup-98:The Second Robot World Cup Soccer Games and Conference, pages 120{124. Springer verlag, 1999.

Johan Kummeneje, David Lyb.ack, and H_akan L. Younes (1999), UBU – an object-oriented RoboCup Team. In Johan Kummeneje and Magnus Boman, editors, Int7 1999 Papers. 1999.

Johan Kummeneje (1999), Simulated Robotic Soccer and the Use of Sociology in Real Time Mission Critical Systems. In L. R. Welch and M. W. Masters,editors, Proceedings of RTMCS Workshop, IEEE, December 1999

-VI-

Human-Robot Interaction

A Multi-Robot System Architecture for Trajectory Control of Groups of People

Edgar A. Martinez-Garcia, Akihisa Ohya & Shinichi Yuta

1. Introduction

How to conduct a group of humans towards a target destination by a team of robots is the key-problem discussed in the present context. A suitable multi-robot system (MRS) architecture has been investigated and implemented for guiding groups of people. The present system can be seen as guiding-tours, nevertheless further than such concept this implementation can be though, or it is closer to the model given by several dogs flocking herds of sheep, guiding them towards a targeted place. Dogs and sheep have a minimal way of explicit communication. Sheep herd's trajectory is controlled by a team of dogs (even one). The dogs do bark and/or approach to the herd if there is any situation disordering guidance. The proposed context differs from it, since does not exist any type of explicit signal for guidance, and trajectory control is given by a way based on natural reactions of angle-velocity motions between humans and robots. An extensive theoretical description and experimental results are discussed. Recent progress in robotics and artificial intelligence has made possible to build interactive mobile robots that operate highly reliably in crowded environments. There exist in the research field community several works concerned with guiding-tours, nevertheless tackling different problems and/or deploying different architectures. Few successful works concerned with guiding-tours have been developed (Nourbakhsh et al., 1999; Thrapp et al., 2001; Burgard et al., 1999). Our context has some differences, such as our system is compounded by a team of mobile robots, due to the importance of the task our architecture is centralized and deliberative, the MRS controls the people trajectory motion, and communication between robots and people is based on motion reactions. Section 2 discusses in the limits and scope of this research; section 3 details the architectural framework of the MRS. In section 4 a methodology for people localization by the MRS is presented in deeply. The section 5 briefly describes a proposed strategy for control of the conduction task, while in section 6 the part of robots motion planning is treated. Finally, section 7 and 8 shows simulation results of the proposed methodologies, and the conclusions respectively in each section.

2. Aim of Study

In the present context guided tours are defined as walking or moving from one point to a target location by performing certain conduction tasks, essentially involving mechanisms to crowd people. Three main issues have been regarded as part of the process of conduction:

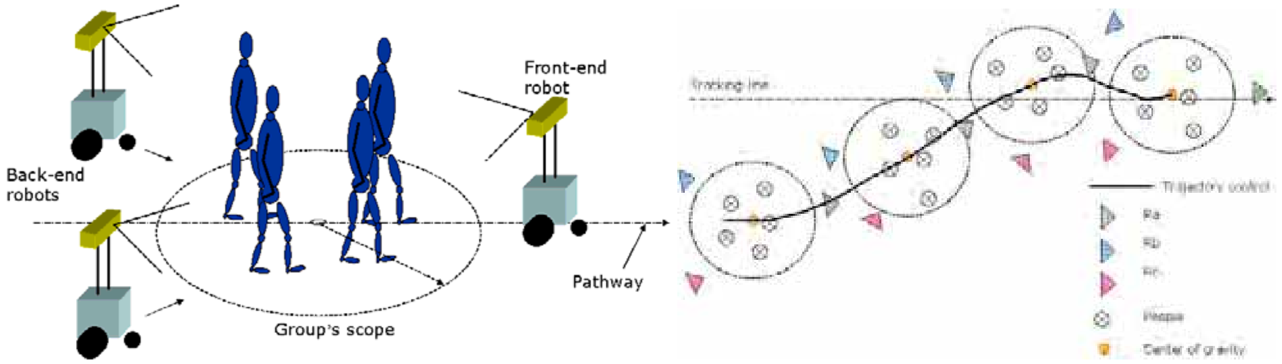
1.) Guiding is defined as the conduction of the group of people through the pathway, by the Ra easily followed by the group of humans (all conditions are kept in a normal status);

2.) Crowding (group size Control). It is the process of collecting the group together still closer than it actually is, while the group keeps striding on the pathway. So, in this context an undesirable situation is when the size of the group area (radius r_k) increases at discrete time k, such that becomes bigger than a desired size or radius r_{ref};

3.) Interception. Another situation is when a person intents to leave the group, or moving away from its scope. It is called interception, because a given robot approximates to he/she attempting to yield the person going back the group. However, despite this task has been considered, it leads to face other challenging problems, but for now it is out of the scope of this chapter.

2.1 Strategy for MRS-based Guidance

The main reason of configuration in Fig. 1 is because only one robot is needed as long as for guidance or conduction is concerned. On the other hand, two vehicles are settled at back-side to observe the people's behavior and/or crowd the group whether it is necessary by means of special guiding tasks.



(a)

(b)

Figure 1. (a) Robots' formation conducting a group of people; (b) The team of robots in formation steering the group of people's trajectory

In order to represent the area coverage of a group of people, we established a circular model which encompasses all the members together as depicted in Fig.1) and Fig.2. Until up now, we have restricted it with a number of people between 2 up to 6 persons in hallways of the University of Tsukuba. The CG is expressed by its components (x; z; θ, v, w), and the MRS senses a measure of it at location (x; z) and heading angle θ, with lineal displacement in XZ-space vk, and angular velocity wk at discrete time k. Thus, CG heading angle represents the group's direction towards it is being displaced or will be displaced.

The presented strategy was planned for accomplishing conduction and it can be divided as follows:

1) Stereo vision-based people tracking.

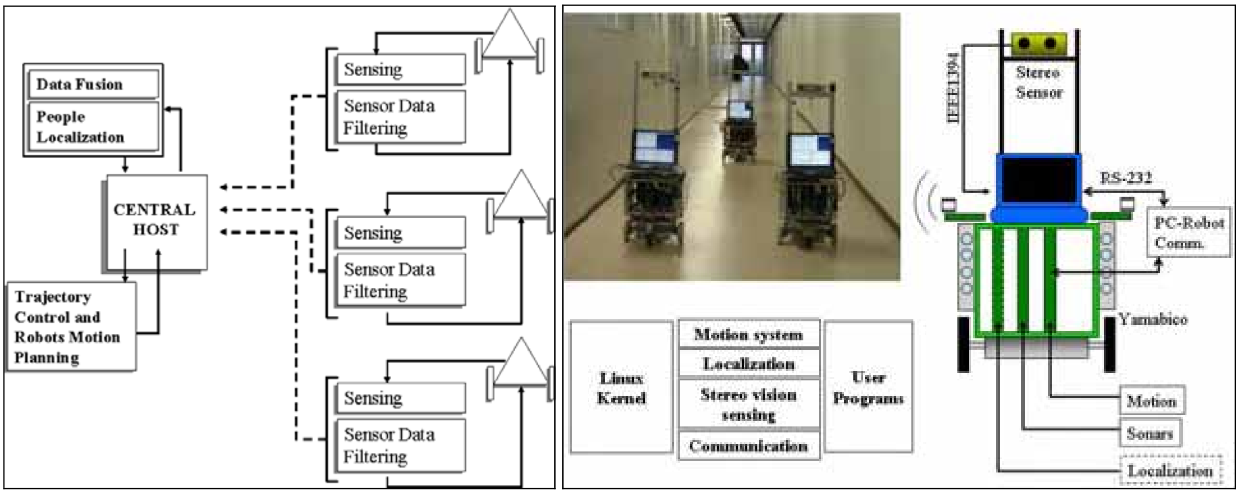
2) MRS architecture design.

3) People trajectory control and robots motion planning.

The item number 1 will be discussed in latter sections, but for further reference it also was presented by the authors in references (Martinez et al., 2003; Martinez-Garcia et al., 2004). The item 2 was presented in (Martinez-Garcia et al., 2005). And item 3 is roughly discussed in latter sections.

3. Multi-Robot System Architecture

The robots conducts the people by a non-active cooperative modality (Murphy, 2000), sharing sensor data, cooperating for tracking and conducting the people without knowing on the existence of other robots. The architecture is compounded by a central host and a team of 3 self-contained mobile robots mechanically homogeneous, although we may think of it, that they can be considered heterogeneous since at least one robot's differences (front-side) arise more functional rather than physical (depicted in Fig.3-(b)). Deliberation is performed in the central host that remains during the entire mission duration as similar architectur presented in (Iocchi et al., 1995) and described in (Cao et al., 1997). A first endeavor of this architecture is to share distributed sensory information. It was established that a reliable way to share distributed percepts was by (1) performing a separate filtering process in each robot; and (2) to carry out with multi-sensor data fusion in the central host as depicted in Fig.3-(a).

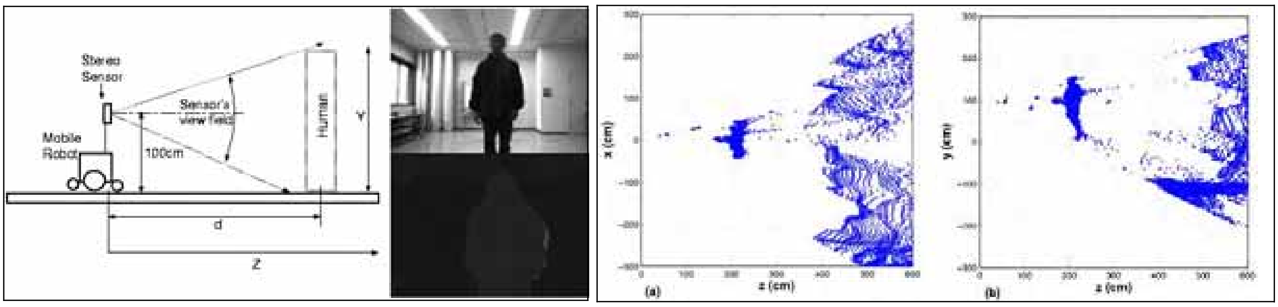


(a) (b)

Figure 2. (a) MRS architecture; (b) robots platform and configuration

The MRS exploits the stereo vision model; because stereo-based ranging data has several favorable points for world sensing in this context. It facilitates: (1) 3D spatial coordinates in real-time; (2) object segmentation (Beymer & Konolige, 1999), although, such advantages could be obtained by other methods and sensors. Each mobile robot was equipped with commercially available stereo vision sensors that provide disparity maps (sum of absolute differences), gray scale images (160x120 pixels) in real-time acquired by an IEEE 1394 bus communication. Each laptop on-board is 900MHz Pentium-III running under Linux. The Fig. 3-(a) depicts the configuration of the robots required for sensing a person (including gray level image and disparity map obtained from experiments). The disparity map represents a matching of the ranged environment by levels of gray. The farther a point is from the sensor, the more darken the pixel becomes. The center of the sensor was fitted approximately at 100cm height. In addition, the stereo parameters were established a priori by choosing and changing such settings manually until they met our needs. In earlier experiments world was measured within an empty area of approximately 800x700cm with a human standing at 200cm away from the sensor for evaluating stereo parameters, sensor's accuracy, measuring timing and sensory info analysis. A first complication in using this model was high rate of noise due to stereo occlusion, and light conditions.

The experimental results are depicted in Fig. 3-(b) that shows the top and side views of sensor data plotting, robot's sensor at (0, 100, 0).



(a) (b)

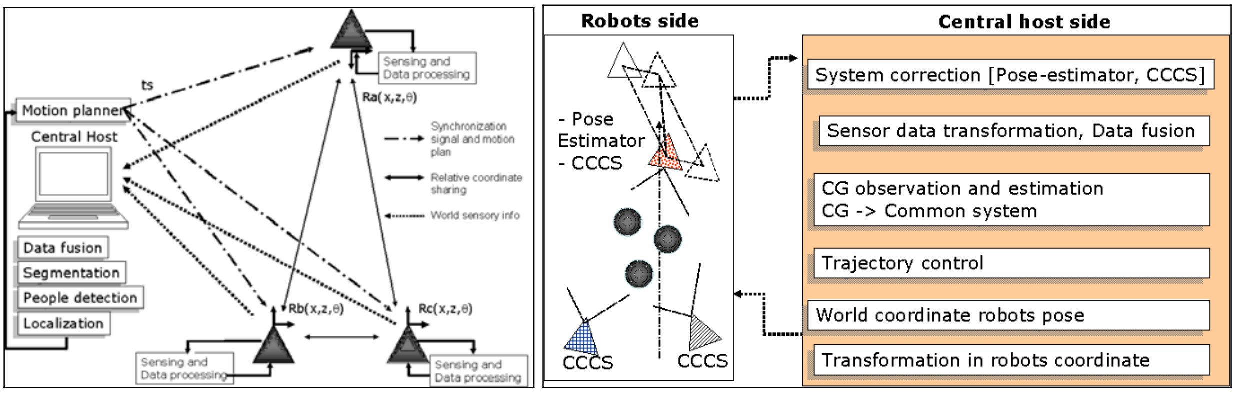
Figure 3. (a) Robot configuration, gray level image and disparity map; (b) ranged data, top and side views

Furthermore, the multi-robot communication system is based on functions for spreading messages and a group-communication philosophy (Fig.4-(a)), as similar architecture presented in (Iocchi et al., 2003). Each laptop on-board has wireless technology via IEEE802.11b. Experimental results on the MRS trade-offs such as sensory info and robots pose sharing are shown in table 1, real needs include transferring messages of about 1kb, nevertheless transactions were performed with 100kb. See (Martinez-Garcia et al, 2005) for further details on the communication architecture. In addition, another issue that depends on the MRS communication architecture is during the guiding-task performance robots localization, which is a critical issue for the team of robots.

Experiments	Ra to the central-host	Rb to the central-host	Rc to the central-host
1	8	9	7
2	10	8	9

Table 1. Time (ms) spent in a round trip for 100 kbytes

Self-localization is accomplished in a cooperative framework, where robots do share a relative Common Cartesian Coordinate System (CCCS). The CCCS development and its usage were presented by the authors in (Martinez-Garcia et al., 2005; Yoshida et al., 2003). The CCCS facilitates the problem of robots localization by sharing a relative system among robots without any world map in advance. The CCCS is an architectural framework embedded in each robotic platform (Yoshida et al., 2003). The method is based on matching measurements arising from ultrasonic range sonar and odometers. The 3 mobile robots get their pose respect to the objects in the world; if there are differences, the robots self-correct such measurements. Nevertheless, since CCCS is unable to self-correct error pose in navigation time, only Ra deploys a special pose estimator system (Watanabe & Yuta, 1990) aimed to correct positional errors. Measurements arising from CCCS and the pose estimator are combined into the central host, which have a global model of the world to overcome the problem of localization and navigation of the robots. The sequence of steps for such process is described in Fig.4-(b).


ts
Motion planner
Central Host
Data fusion
Segmentation
People detection
Localization
Sensing and Data processing
Ra(x,z,θ)
Synchronization signal and motion plan
Relative coordinate sharing
World sensory info
Rb(x,z,θ)
Rc(x,z,θ)
Sensing and Data processing
Sensing and Data processing
Robots side
Central host side
- Pose Estimator
- CCCS
CCCS
CCCS
System correction [Pose-estimator, CCCS]
Sensor data transformation, Data fusion
CG observation and estimation
CG -> Common system
Trajectory control
World coordinate robots pose
Transformation in robots coordinate

(a) (b)

Figure 4. (a) MRS architecture; (b) robots' relative Cartesian system and world representation in the central host

4. Multiple People Localization

In this section, the methodology for multi-human localization is discussed in detail, theoretical foundation and experimental results are shown. Firstly, a team of mobile robots was deployed to localize all the humans in a target-group. A team of robots overcome in great extent the problem of partial occlusion generated by the members themselves. In Fig. 5, the configuration of an experiment is depicted; Basically, for the purpose of evaluation the environment was set a priori within 2 rows of people, 3 and 2 people at front and back respectively, between Ra (front) and Rb, Rc (left, right also respectively).

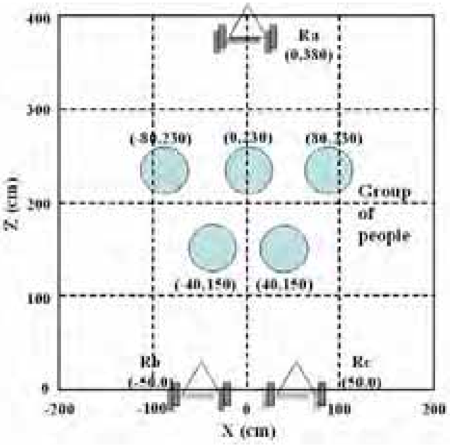
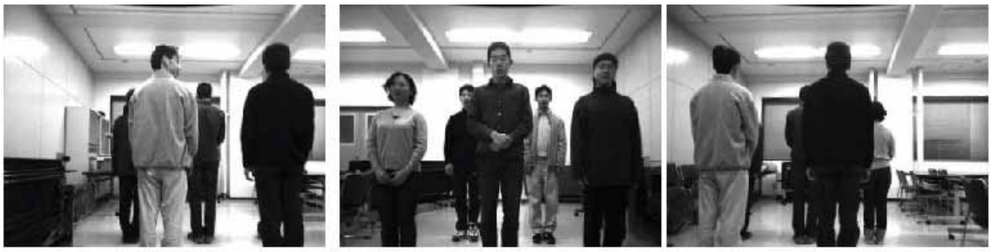


Figure 5. Experiment configuration with 5 people, and 3 mobile robots

From this experimental configuration, the Fig.6 depicts gray level images arising from each robot (from one stereo camera). Actually they do provide their fields of view, at their given geometrical positions, which can be matched with the robots' position at depiction of Fig.6.



a) b) c)

Figure 6. Image views from a) Rb, b) Ra and c) Rc

In Fig. 6, from any given location, humans are difficult to perceive totally due to multi-human occlusion. In the case of Ra (Fig. 6-b), it can only sense partial regions from people at the back, and Rb and Rc barely perceive 3 or 4 persons. From the stereo images a ranged world model was obtained, and sensor data (3D) have been plotted in Fig. 7. It shows the sensor readings top view for each of the 3 robots. All plots have been arranged into a common coordinate system.

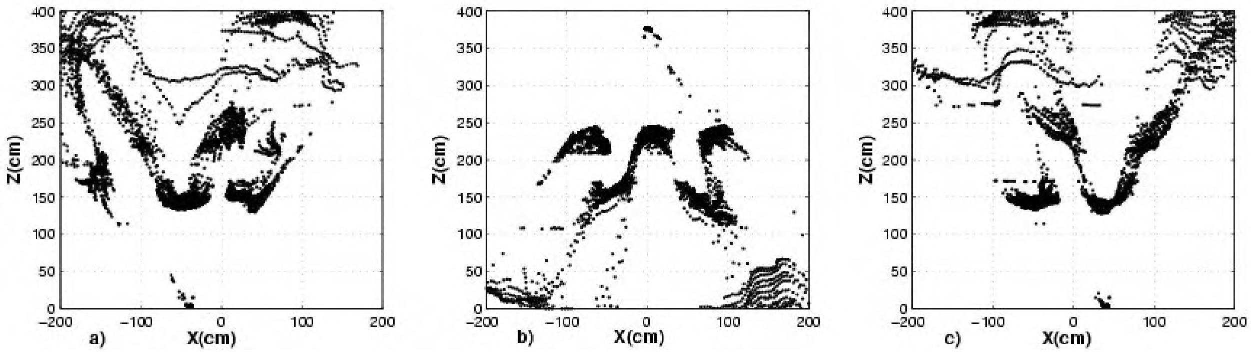


Figure 7. Sensors reading top view, a) Rb, b) Ra and c) RC

As depicted in Fig. 7, due to noise generation, there is uncertainty in sensor data representing the humans.

4.1 Data Processing

A methodology for sensor data filtering, and human localization have been developed, mainly organized by 5 general stages:

1.) Environment sensing (previously introduced)
2.) Filtering:
    a) Zones discrimination
    b) Noise reduction
    c) Quantization filtering
3.) Distributed multi-sensor data fusion
4.) Clustering based segmentation
5.) People localization

Data filtering is considered as an essential part as a preamble for multi-sensor data fusion and people segmentation process. Filtering essentially, is compounded by 3 main parts: 2-(a) zones discrimination, 2-(b) noise reduction and 2-(c) a routine of data quantization working as a spatial filtering. In multi-sensor data fusion (3), the information is arranged onto a common coordinate system to construct a short-term world model. In addition, we have implemented a segmentation method relaying on a clustering algorithm (4). Eventually, people are detected and localized (5).

4.2 Zones Discrimination

A first important regarding in the present strategy for filtering involves a spatial modality, which deals with an early elimination of unnecessary spaces. It has been called zones discrimination process because filtering relays on 2 thresholds. Both thresholds exhibit a couple of heights of the human body (shoulders and keens). Discrimination is applied to points falling out such thresholds; meanwhile, points between them (20 and 40 cm) are considered information with high likelihood to be within the set of points of ranged people. In general, it was assumed about 170cm as a statistical average for people's height. Let us consider a set called RAW of vectors raw^i, such that RAW={raw^1, raw^2,..,raw^n}, and each vector $raw^i \in \Re^3$, thus raw^i ={raw^i_x, raw^i_y,..,raw^i_z} corresponding to the 3 spatial components. Likewise, let's establish that V is a subset, such that V ⊂ RAW. Thus, our model for point discrimination is expressed in by V={raw^i / th_1<=raw^i<=th_2}. Where, the set RAW represents the raw sensor data, V is the set with the points of interest evaluated by raw^i_y. The range is delimited by $th_{1,2}$ representing knees and shoulders, and points which were ranged out are floor, ceiling and even mismatched points are zones from which sensor generates a considerable large amount of points, and are not useful for our purpose. It can be noticed the difference of quantity and points remaining after the discrimination process. By preserving the points between shoulders and knees, their quantity was approximately 15% of the total being still useful for later processing, while about 85% was removed (see results in Fig.8).

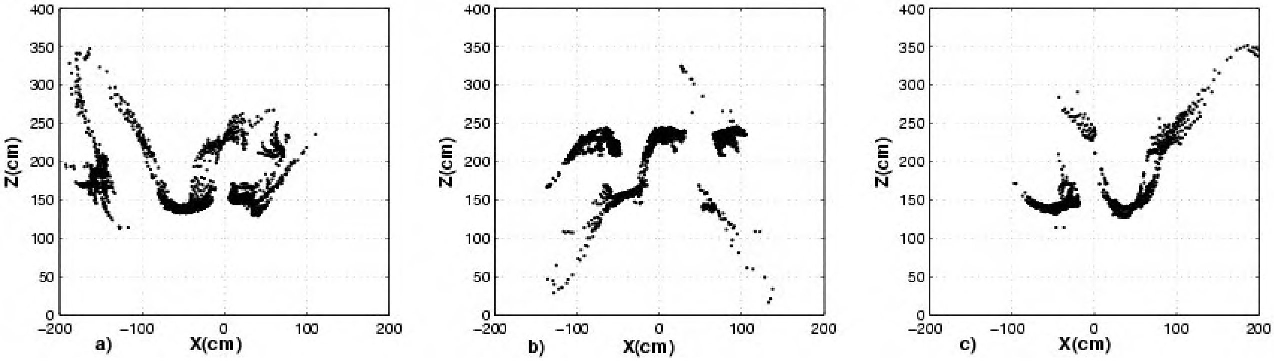


Figure 8. Top view, results of point discrimination, a) Rb, b) Ra, and c) Rc

4.3 Noise Reduction

Noise reduction is an essential task to avoid undesired nosy regions, and keeping a suitable world model, as sensory data is not a perfect noiseless data model. It was implemented a noise reduction spatial filtering for dealing with noisy areas defined as small 3D spaces containing poor density and low uniformity distribution of sensor data, which were about less than 15 3D-points in a volume of 10x10x120cm (XZY) for one robot. Humans on the other side keep a high density and a uniform distribution of points. A suitable solution to tackle this problem was by implementing a 3D spatial filtering window with a threshold of point's number. The filtering window slides over X and Z directions over the ground plane. The filtering discriminates sets of points if their number in the cell, is lower than the threshold. This process is represented by the set V={v^1,v^2,...,v^m}. Thus, defining the space D={D^1, D^2,..,D^p}, from where each subspace is defined by D^k={d^j_1,d^j_2,..,d^j_l} containing a number l of points $(dj^i)^l_{i=1}$ to be evaluated in expression (1), the filtering-window in process state is called the j^{th} cell, such that its upper left coordinates are given by (x_j,z_j) and size by fwin. Thus, results are then evaluated by (2) into the set H. Regarding a number of points called l and restricted by a threshold th_n, the action of filtering is given by (3),

$$D^j = \{v^i \mid x_j \le v^i_x \le x_{j+fwin} \wedge z_j \le v^i_z \le z_{j+fwin}\} \tag{1}$$

$$H^k = \begin{cases} D^j, & l \ge th_n \\ \phi, & \text{other} \end{cases} \tag{2}$$

Where H $\subset$ V, such that H={H^1,H^2,..,H^p} and H^j={h_j^1,h_j^2,..,h_j^l} for $hj^i \in \Re^3$. Likewise, the filtering area was given by fwin=10cm and threshold equal to 20 points for the present experiments. The results from each robot are plotted in figure 9-a), b), c). With this algorithm in all experiments, data were still reduced about 27% less (from the resulted after zones discrimination), remaining approximately 73% for latter processing (although it was depending on the cell size).

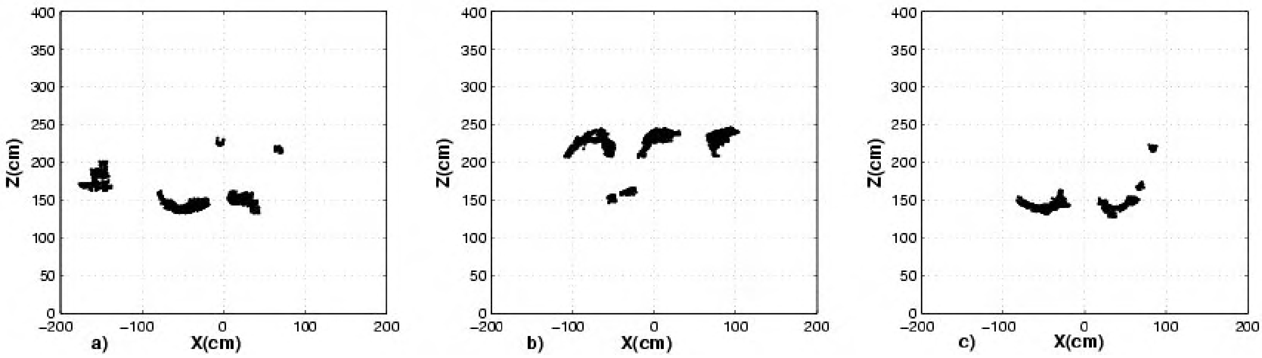


Figure 9. Results of noise reduction process. a) Rb, b) Ra and c) Rc

The people who were partially or completely occluded by other people hardly might appear in the sensory model after noise reduction, due to poor density of points.

4.4 Quantization Filtering

The purpose of quantization is a reduction of points that decreases considerably more the burden of computation, and projects 3D points into a 2D model, as the latter is enough to represent humans' position. Moreover, this task aims to project objects' within a lower density of points. The principle of this algorithm is a quantization of points in the XZ space, performing a filtering through a small square cell. Regardless the number of points within the cell, if it keeps at least one point, then only the central value of the actual cell will represent such cell-area. It was found that it did not affect the final occupancy data representation on the XZ space, because of the small size of the cell. Thus, our model is given by two sets of vectors in $\Re^3$, where H is the set of vectors of the filtered data from (3), and C is a new set, which its space XZ is divided by a new size of cells represented by hwin. Where C={C^1, C^2,..,C^q} and C^k={k, k,..,k}, and C^k contains the kth cell index as many times as the number of points in such cell. For points reduction, XZ space will be split and referenced by cell addresses as expression (3), whereby h^i_x is transformed by $((h^i_x)/(hwin)+1)>0 \;\forall \mathbf{Z}$ being Z the set of integers. Then transforming the space H into a C space,

$$C^k = \left\{ k \mid \left(\frac{h^i_x}{hwin} \right) + 1 = x_k \wedge \left(\frac{h^i_z}{hwin} + 1 \right) = z_k \right\} \tag{3}$$

Thus, the set of points in each cell are defined in the domain of the set C^k, additionally let's define the center of the cell H^k by (x_k+hwin/2) and (z_k+hwin/2), where every cell origin is given by (a_k,b_k) in cm. In Fig. 11-b), each cell has a determined number of points, and those points are labeled or referenced with their respective number of cell given by (3). The sense of this technique is for representing the content of a cell, instead of considering all the points in it, as in Fig. 11-c). From here, as depicted in Fig. 11-d), only one 2D point is considered, and it will appear at every non-empty cell.

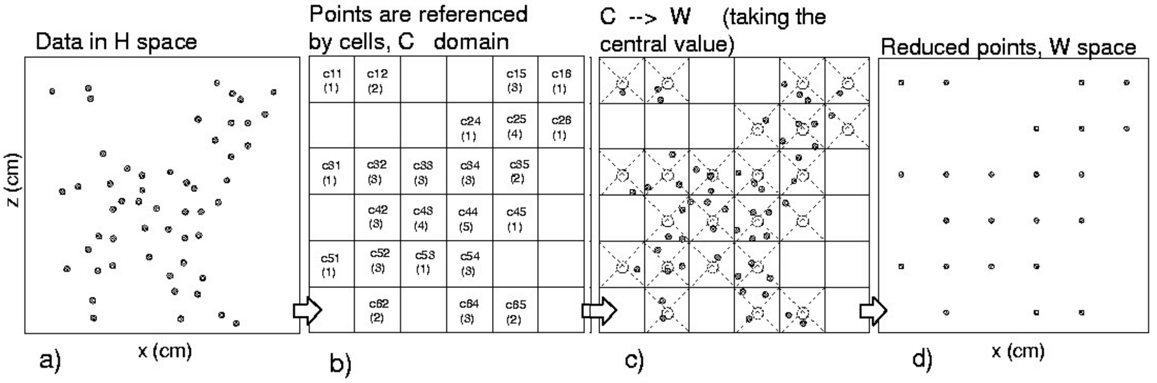


Figure 10. Point reduction process

Eventually, the process of data reduction is calculated, resulting the set W={W^1,W^2,..,W^q}. The mechanism of point reduction was by the expression (4) which not only converts cell dimension data into the original XZ-space points, but also calculates the central cell-value.

$$W^k = \begin{cases} \phi, & C^k = \phi \\ (x_k + \frac{hwin}{2}, z_k + \frac{hwin}{2}), & \text{otherwise} \end{cases} \quad (4)$$

The space is basically a grid where each unique cell contains a different number of xz-points, such that by considering the C space it resulted more suitable to compute small number of cells with points having a same label, than a large number of points addressed by an XZ coordinate (H domain), see figure 10-a). In general, the rate of 2D points was highly reduced as well as time computation for latter data processing. Its significance comes from the fact that once data fusion is performed the total number of points (from the team of robots) will generate a certain huge burden for segmentation performance. Results of point quantization are depicted in Fig. 11.

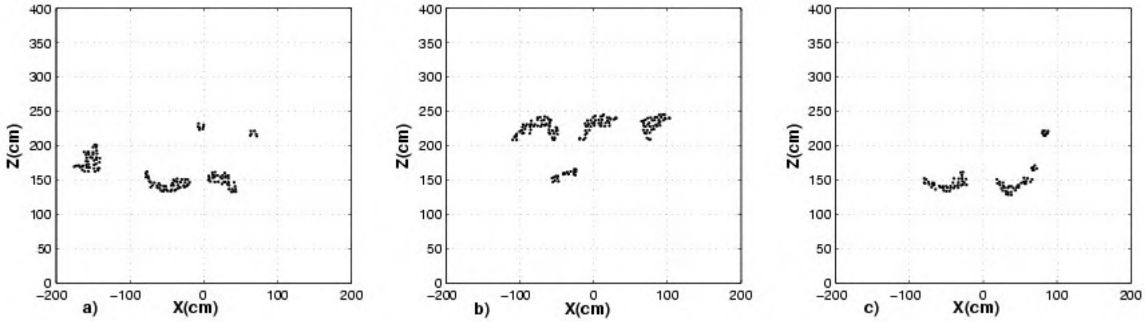


Figure 11. Point reduction results, robots a) Rb, b) Ra and c) Rc

The number of 3D points was reduced over 15,000 from the original ones (raw data) only in Ra. Practically, by means of this process nearly 93% of the data was still removed without lose 2D occupancy of ranged objects (people and furniture). Thus, for the purpose of data fusion about 7% of the points in each robot have resulted. Now on, these results can be more advantageous, and segmentation can be more likely attained.

4.5 Data Fusion, Segmentation and People Localization

The aim of multi-sensor fusion in the proposed method is to yield a global model of the world by sharing local sensory data arising from each robot's location. The way of data association is by setting onto a common coordinate system the sensory information filtered by each robot. Multi-sensor data fusion makes possible to reconstruct objects that do not exist in the data model of a determined robot's sensory info. In addition, a sensor-sensor calibration was required for attaining accurate data fusion. To overcome the problem of dynamic distributed sensor data calibration, 4 main parts (among other very particular details) were implemented in the architecture:

1.) Multi-robot positions (attained by the CCCS).
2.) Sensor data calibration (correction of Cartesian errors in sensors fixation).
3.) Synchronized sensing. (same spatial-temporal sensor data is fused).
4.) Transformation of sensor data (for representation in a common world coordinates).

Once data fusion process has been carried out, clustering based segmentation is then performed as a preamble to typify the nature of the objects. Multi-sensor data fusion is carried out in the central host at every discrete time t_k. Further than improve the state of a world model at every updating, it has only a short-term purpose and its time of life is limited to less than the updating time. Thus, the points in $w^i=(w^i_x, w^i_y, w^i_z)$ were translated by (5) according to their positions (x_r, z_r) and heading angles θ_R by,

$$\begin{pmatrix} \omega^i_x \\ \omega^i_z \end{pmatrix} = \begin{pmatrix} \mathrm{Cos}(r_\theta) & -\mathrm{Sin}(r_\theta) \\ \mathrm{Sin}(r_\theta) & \mathrm{Cos}(r_\theta) \end{pmatrix} \begin{pmatrix} w^i_x \\ w^i_z \end{pmatrix} + \begin{pmatrix} r_x \\ r_z \end{pmatrix} \tag{5}$$

Global coordinates were computed in (ω^i_x, ω^i_z) as a previous step for data fusion process. Now on, the set K holds a model representation of the environment which is the result of data fusion given by the following expression $K=[\omega]_a \cup [\omega]_b \cup [\omega]_c$. Finally, such expression determines the space K which represents the short-term world model (see results in Fig.12-(a)). This data model is now a key-issue for the process detailed in next section.

How to determine what set of points are representing a unique object was established by grouping 2D points which their unique common feature is a distance that differentiates what group of near points belongs to a particular object. The purpose of any clustering technique is to evolve a K x r partition matrix of data set $S(S=s^1,s^2,..,s^r)$ in $\Re^N$, representing its partitioning into a number of sub-clusters, say k, of clusters (Bandyopadhyay & Maulik, 2000). The method for segmentation is based upon a threshold distance between two points in the set K. If two points are close enough, then those given points are labeled as part of a same subgroup. A determined point in process of clustering is compared with the rest in the total set.

The end of such process ends up with a group of clusters representing the objects in the area surrounded by the team of robots. The method presented in this context has three main bases as core for segmenting:

1. Unclassified point vs. unclassified point

When 2 points $p=(p_x,p_y,p_z)$ and $q=(q_x,q_y,q_z)$ have not been labeled yet, and are close enough that an distance d between them is shorter than the established threshold d_{th} ($d \leq d_{th}$), then both points are classified as part of the same sub-cluster S^1, represented in expression (6) as,

$$\sqrt[2]{(px-qx)^2+(pz-qz)^2} \leq d_{th} \Rightarrow S^1 = \vec{p} \cup \vec{q} \tag{6}$$

2. Unclassified point vs. classified point

When a point p ∈ $\Re^3$, which is non-classified yet and is close enough to the point s^i in sub-cluster S^1 that the distance between them is such that $d \leq d_{th}$, then set and point will be classified as part of the same sub-cluster, in S^2 represented by the subset of the equation (7).

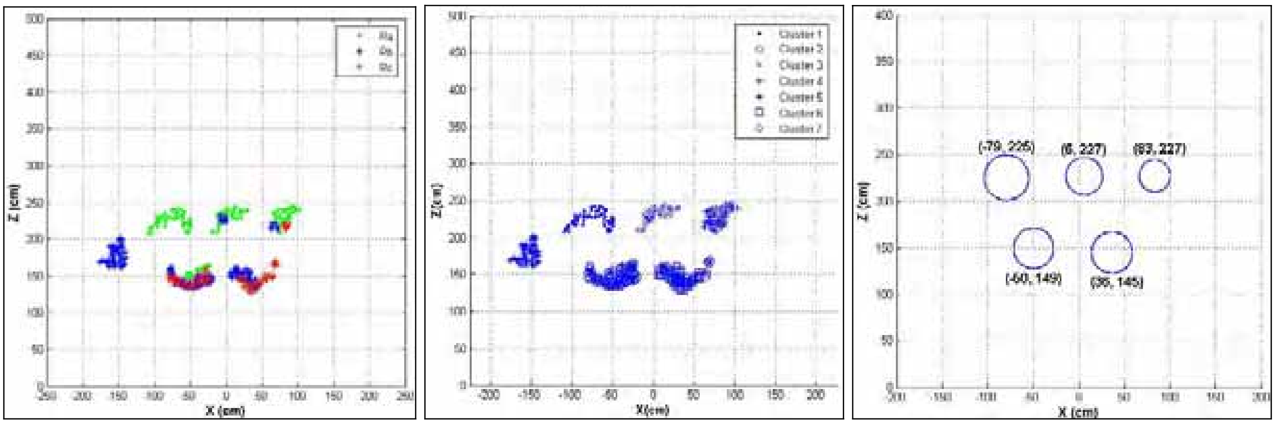
$$S^1 \neq \phi, \sqrt[2]{(p_x - s^{1^i}_x)^2 + (p_z - s^{1^i}_z)^2} \leq d_{th} \Rightarrow S^2 = \vec{p} \cup S^1 \tag{7}$$

3. Classified point vs. classified point

When 2 sub-clusters of points S^1 and S^2, which are already pre-classified and are close enough that at least a point of both groups (S^1 and S^2) are close enough that $d \leq d_{th}$, then set S^1 and set S^2 will be classified as part of the same sub-cluster, in S^3 represented by the subset of the equation (8).

$$S^1, S^2 \neq \phi, \sqrt[2]{(s^{1^i}_x - s^{2^j}_x)^2 + (s^{1^i}_z - s^{2^j}_z)^2} \Rightarrow S^3 = S^1 \cup S^2 \tag{8}$$

In situations like the previously explained once the points has been totally labeled, the set K is partitioned in r^{th} subgroups of vectors called $S^k=\{s^{k1}, s^{k2},..,s^{kr}\}$, where each of such points has in common a label value. Now let's say that the distance threshold was denoted by d_{th}=15cm. Segmented objects are classified by a determined numeric label generated and assigned automatically during the process as depicted in Fig.12-(b).



(a) (b) (c)

Figure 12. (a) Multi-sensor data fusion; (b) segmentation results; and (c) multi-people localization results

In Fig. 12-(b), only 6 clusters of points were found by performing the algorithm. Noting that clusters 5th and cluster 7th, the former one represents a section of a furniture, which at the time of the experiment it was in the field of view of Rb. Likewise, cluster 7th, is a small fragment of bit of noise associated with a part of data people. In spite of that, cluster 7th is considered as noise because in any possible manner it does not have enough size-feature as humans, such as the number of points, height, width and depth. A similar consideration was taken for the cluster 5th, because it is not as tall enough as a human. Basically size constraints were established a priori to differentiate humans from other objects, for this process we regarded some parameters such as clusters' center of gravity (x,z), number of points, maximum and minimum values of the XZ-space as height of each object. Fig. 12-(c) shows how humans have been successfully localized; each circle expresses humans' positions. Their scopes surround the highest concentration of ranged points close to their center of gravity. Basically human's positions were represented by their center of gravity. For depiction of humans' scope the clusters' standard deviation σ_d was deployed as radius d for drawing the circles. For the subfigure 12-(c), circles were plotted with radius of $3\sigma_d$. Similar works with distributed networked systems for multi-people tracking but different contexts were presented by (Nakazawa et al., 1998; Tsutsui et al., 2001).

5. Strategy for People Trajectory Control

Localizing every human in a group is an approach that allows an estimation of the people group's center of gravity (CG). In such estimation, a noisy computation of CG is regarded for several causes such as missing the measurement of a human due to temporal members' occlusion, or bad light conditions and so forth. A strategy for improving the observation of the CG is a critical issue to yield trajectory control. A proposed framework for controlling the trajectory of the group's CG is presented in Fig.13.

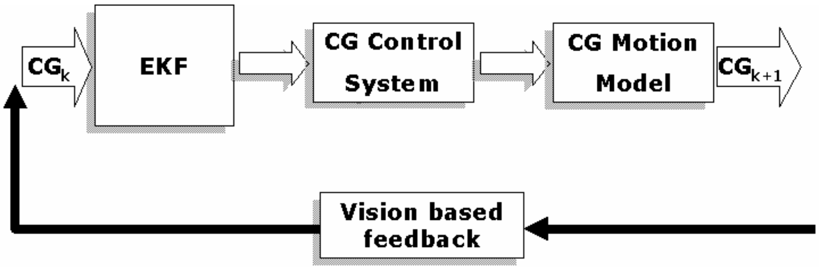


Figure 13. Block diagram of the vision-based feedback control

The CG observation is easily accomplished by calculating it after multi-people localization. CG estimation is a filtered version of the CG carried out by a traditional extended Kalman filter. The trajectory control system provides a way for steering the CG leading to track a desired pathway. Eventually a motion model allows predicting into discrete time k+1 a desired CG position.

5.1 Trajectory Control Model

A principle in the proposed method is that the team of robots must steer the GC towards a desired pathway. The equation (10) expresses a model of the CG angular acceleration (α_k), it yields a trajectory tracking while moving from the actual GC location towards the tracking-line having a distance Δx between to be reached. Nevertheless, the robots team can not explicitly control such α_k, but can in some extent affect the CG heading angle θ at navigation time. The equation also requires the CG's angular velocity wk feed at every cycle. In this context the equation (9) models a lineal feedback control.

$$\alpha_k = -k_1 \Delta x_k - k_2 \theta_k - k_3 w_k \qquad (9)$$

The gain is established by the constants k_1, k_2 and k_3 and were determined by trial and error for the robots. The steps for calculating a desired CG's location at time k+1 is by correlating the previous equation value α_k with the set of equations that model the kinematics of motion of the CG (not the control). The fig.14-(a) shows a depiction of the kinematical parameters regarded for its control already discussed. CG is a dynamic particle surrounded by a circular scope, which is defined by the magnitude of the crowding (members of the group). For its control every member's dynamic status is averaged in its CG's behavior.

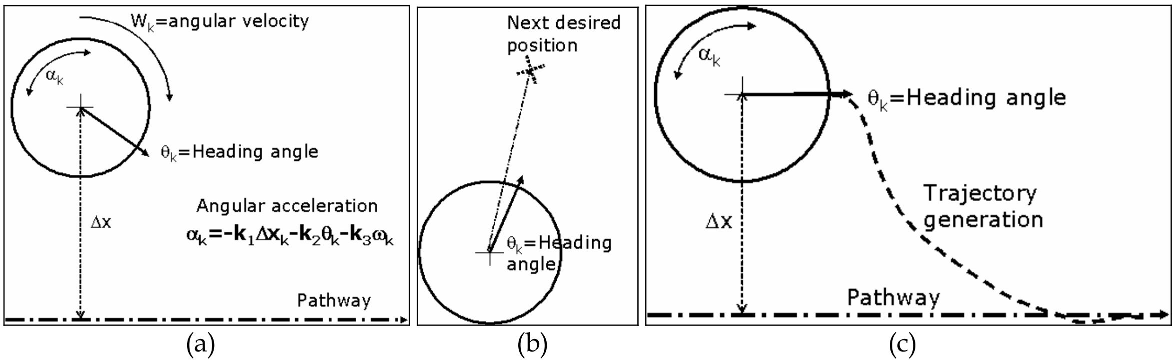


Figure 14 (a) Feed-back control model; (b) Motion model; and (c) Control and trajectory generation

5.2 Group's Motion Model

As depicted in Fig.14-(b) regardless the actual kinematical tendency at time k of the CG, there exist need to determine a desired future position value at k+1 to properly yield control over x and z coordinates for the CG's trajectory. A proposed motion model for the CG correlates a set of simple equations for getting regarding a new pose Pk+1 at certain desired velocity. A projection of the group's angular velocity wk+1 is given by the equation (10), involving a measurement of the actual w_k and α_k. The result from equation (10) allows to correlate a prediction of the current angle θ_k into next discrete time θ_{k+1} by (11) as

$$w_{k+1} = w_k + \alpha_k \Delta t \tag{10}$$

$$\theta_{k+1} = \theta_k + w_{k+1}\Delta t \tag{11}$$

Basically, the previous scalar results become fundamental to obtain lineal results in velocity and distance of the CG as vector representations yielded by expressions (12) and (13).

They exhibit a more representative condition of the CG's motion behavior. Certainly, lineal velocity and distance vectors in $\mathcal{R}^2$ (x, z components). Being γ the gain that provides a control behavior to the vector velocity v on how fast it is stabilized in (12). A desired velocity v_{ref} is a constant value established a priori. Eventually, the model for predicting a next desired position of the CG is yielded by equation (13) as,

$$\vec{v}^{\,x}_{k+1} = \vec{v}_k + \gamma\ (v_{ref} - \vec{v}_k) \tag{12}$$

$$\vec{P}_{k+1} = \vec{P}^{\,x}_k + \vec{v}_{k+1}\Delta t \tag{13}$$

Thus, angular velocity, heading angle, velocity and position vectors are the variables that define the proposed CG motion model. In fact the group of people kinematics is seen and analyzed throughout its CG considered as a self-driven particle.

The Fig.15 shows the results obtained by merging the control and the motion model for the CG. In the simulation the CG started at 100 cm away from the tracking line heading towards 60° with a sequence of Gaussian noise a long with the trajectory generation data. The sampling time was established in 0.125 sg, CG displacement over time was vx=0.1m/sg, vz=0.5m/s and angular velocity w_k=10°/s. Likewise for this simulation results k_1=3.66, k_2=2.5 and k_3=-7.54. Besides in the plotting the angle control, x-velocity and z-velocity behaviors are depicted.

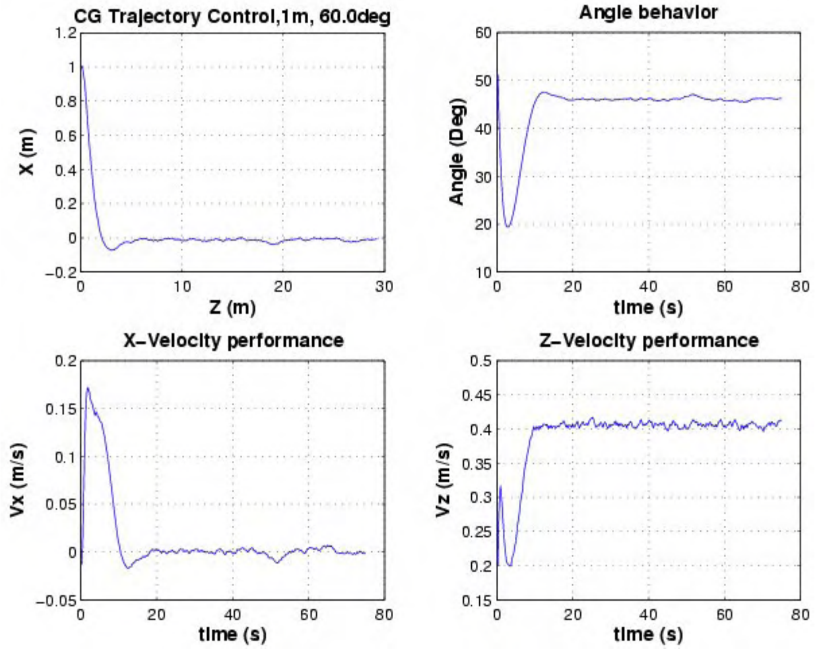


Figure 15. Results of the trajectory control merged with the motion model results

5.3 Center of Gravity Estimation

One of the major important issues for the mechanism to control the CG is the measurement of it, which yields noisy observations. A proposed solution is the implementation of a traditional version of the Extended Kalman Filter (EKF) (Kalman, 1996 ;Meybeck, 1979; Welch & Bishop, 2002) that resulted to be suitable enough to overcome the problem of noise filtering and to get present and future estimations of the CG kinematics. The state n-vector of the process $x_k = (x, z, \theta, v, w)$ at discrete time k defines the group's pose (x,z), lineal and angular velocities v,w respectively. The observation of the system, which relates the sensory information is expressed in $z_k=(x;z)$ as in equation (14) and Kalman gain K in (15).

$$\vec{z}_k = H\vec{x}_k + \vec{u}_k \tag{14}$$

$$K_k = P_k H^T (HP_k H^T + R_k)^{-1} \tag{15}$$

Likewise, the present implementation also included the variable $H_{[2x5]}$ stationary over time matrix noiseless connection between the vectors x_k and z_k, and the u_k that correlated a Gaussian white sequence.

$$\bar{x}_k = \hat{x}_k + K_k(\vec{z}_k - H\hat{x}_k) \tag{16}$$

$$\hat{P}_k = P_k - K_k HP_k \tag{17}$$

It was possible to write an update equation for the new estimate x_{k+1}, combining the old estimate x_k with the measurement data z_k. Additionally, a subsequent part of estimation process suggests also the update covariance matrix over time. Basically, the projection of estimate x_{k+1} and the vector state error covariance matrix.

$$\bar{x}_{k+1} = \Phi\hat{x}_k + q_k \tag{18}$$

$$\bar{P}_{k+1} = \hat{P}_k + (A + A^T)\hat{P}_k\Delta t + (A\hat{P}_k A^T + \Sigma_w)\Delta t^2 \tag{19}$$

Also the EKF equations express the projection into k+1 of previous estimate x_k, and it relates the state transition matrix of the process **Φ** (also non-stationary), and projection into k+1 of the covariance is involved. Fig.16 depicts the results obtained from merging the EKF with the trajectory control and motion models.

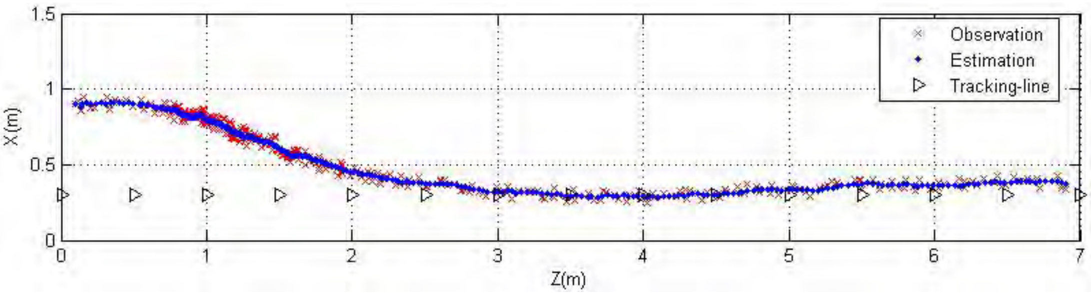


Figure 16. Kalman filter with trajectory control simulating cg's behavior

The observations of the CG are symbolized by a cross character, at less than 90 cm away from the tracking-line heading 10° (observations include Gaussian noise), and sampling time was set to 0.10 sg for 350 discrete samples and an angular velocity w=10°/sg.

7. Robots Motion Planning

The basic principle relays on the fact that a single robot (Ra) is suitable enough to provide guidance (not control). Meanwhile the rest of the robots at back are purposed to share observations and controlling the size of the people dispersion. Figure 17-(a) depicts a circular model encompassing the group and its model for to determine a desired size r_{k+1} is given by the expression (20). The main element for affecting the crowd dynamics relays on changing the actual radius r_k until reaching a desired radius r_{ref} (established a priori) by deploying the robots at certain positions and speeds. If the condition r_k is $r_k > r_{ref}$ exists, the process of crowding is performed (the smaller the r_k, the more the crowding). In this strategy the team of robots gets closer or farther from the CG, forcing the people to modify their inter-space. Once the trajectory control yields the next desired CG, the radius is measured and a robots motion plan takes is performed

$$r_{k+1} = \begin{cases} r_k + \beta(r_{ref} - r_k), & r_k > r_{ref} \\ r_{ref}, & r_k \le r_{ref} \end{cases} \tag{20}$$

The Ri (i={a,b,c}) poses are determined based on the CG location as depicted in Fig.17-(b), according to the angles given by δ_i. Δs is the distance required for the field of view of the sensors settled as a constant. Moreover, the heading angle for the team of robots is already given by the Kalman filter equations leading towards the desired pathway.

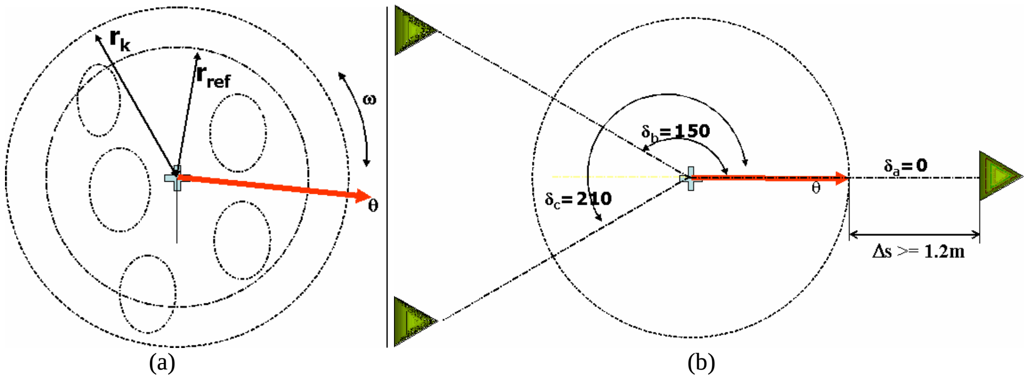


Figure 17. (a) Group's size control model and parameters; (b) configuration of robots' formation

Since the team of robot must reach certain position at k+1 for yielding control of the group's size, the change of speeds are given by the equation (21), where Ri is the vector pose of robot i at every discrete time interval Δt.

$$V_{k+1} = \frac{\left|Ri_{k+1} - Ri_{k+1}\right|}{\Delta t} \tag{21}$$

7. Simulation of People Group under Trajectory Control

In reference (Kirkland & Maciejewski, 2003), an attempt to simulate crowd dynamics by pedestrians affected by the presence and introduction of mobile robots was presented. Such context considers a large number of pedestrians and few robots in order to study and understand its impact and effect in wide areas people behavior. That reference as the present work is considering the usage of the Social Force Model (SFM) originally introduced by (Helbing & Molnar, 1995). However, in the present work, the model proposed by Helbing has a different application as we attempt to adapt the model to simulate a reduced number of pedestrian behaving as a group following the leader robot Ra and affected by the presence of robots Rb and Rc. It is suggested that the motion of pedestrians can be described as if they would be subject to social forces. The corresponding SFM can be applied to several behaviors. It describes the acceleration towards a desired velocity of motion; it also terms reflecting that a pedestrian keeps a certain distance from other pedestrians and borders; and a term modeling attractive effects. The equations of the SFM involve: (1) A model for the desired direction of each pedestrian; (2) models repulsive effects (avoid obstacles and/or other member of the group); (3) models attractive effects (pursuing Ra, a chatting with other members); and (4) models some random variations of the behavior. Until this stage we have obtained experimental results in laboratory with the team of robots and sensors data. However, a simulation model can give us good approaches to prove the effectiveness of the proposed trajectory control model, and the verification of the methodology and strategy. Moreover simulation let us confirmation of the proposed people trajectory control with the robots motion planning. Finally, a human motion modeling can be generated. The algorithm that was implemented for simulation is described as following:

1) An initially random location of the members.

2) Members pursuit to Ra's orthogonal line respect to Ra's heading angle.

3) The CG observation and group size (rk) is provided.

4) Estimation of the CG (filtering) is yielded.

5) With the CG estimation a next desired (controlled) position is obtained by merging of the trajectory control and CG motion model.

6) The next desired radius is determined (the farthest member is the actual maximum radius).

7) Robots move towards their angles depending on CG's position and heading.

8) Again from step 2).

With this algorithm, figure 11 depicts the simulation results for the conduction task, by merging all the models proposed in this paper.

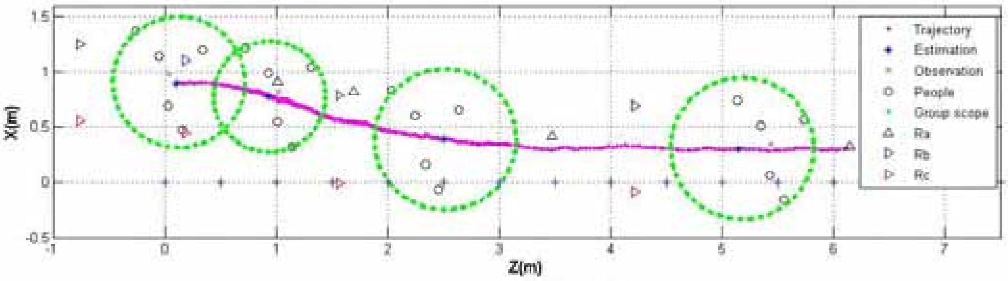


Figure 18. Simulation results of the guiding-task. Merging of the Kalman filtering, the CG-trajectory control model, the CG-prediction motion model and the robots motion plan

8. Conclusions and Future Work

It has been considered that an important issue of this research work relies on the regarding of this modality for people conduction. A given contribution has been the methodology for conduction, its strategy for accomplishing the task-goal and the implementation of the system itself. It has been considered that most important issues featuring the present architecture are synthesized as:

1.) A framework for people trajectory control model.
2.) Guidance is mainly constrained by being of implicit communication type.
3.) Motions reactions are the means of interaction for trajectory control between robots and people.
4.) Non-active cooperation modality is given by the robots in this architecture.

Furthermore, there are some important points of the guiding situation style, which were regarded as the basis for the MRS architecture. Likewise, some of the functionalities and part of the strategy for accomplishing conduction tasks that still deserve further considerations for investigation. For mentioning some of them; while conduction, people's assumptions are tied to the fact that they follow Ra, and/or just follow the crowd towards Ra's direction (specifying that direction for navigation is determined by Ra). The philosophy for attaining conduction is leader-based robots' formation, and several robots surround the group of people (it does not happened in other related work). Part of what affect psychologically the target-group people during the conduction task is that they walk feeling of being observed and the approach of back's robots. The team of robots affects the group's crowd dynamics depending on positions and speeds. Besides, summing up in the scope of this work a multi-robot system architecture purposed to guide a group of people, a vision-based multi-people tracking by team of robots, a trajectory control model and a robots motion planning system were discussed. In the present context the approach has been to depict results of successful multiple people localization. We discussed in detail a method by deploying a MRS within a centralized architecture, since it has resulted enough for a task-oriented approach. The content of this paper is a critical part as a preamble to accomplish real experiments for interaction with different social groups of people aimed to provide guiding-tours. A methodology for processing and sharing distributed sensor data and multi-sensor data fusion was detailed with experimental results by deploying a team of 3 mobile robots, as well as evaluating robustness and reliability of the methods.

9. References

Bandyopadhyay S. & Maulik U. (2000). Performance evaluation of some clustering algorithms and validity indices, IEEE Tran. on Pattern Analysis and Machine Intelligence, 12, 1650-1654.

Beymer D. & Konolige K. (1999). Real-time tracking of multiple people using continuous detection. In Proceedings of IEEE International Conference on Computer Vision.

Burgard W., Cremers A., Fox D., Haehnel D., Lakemeyer G., Schulz D., Steiner W., & Thrun S., (1999) Experiences with an Interactive Museum Tour-Guide Robot, Journal of Artificial Intelligence, Vol. 114, No. 1-2, pp. 3-55.

Cao Y., Fukunaga A. & Kahng A., (1997) Cooperative mobile robotics: Antecedents and directions. Autonomous Robots, 1, 2-27.

Helbing D. & Molnar P., (1995) Social force model for pedestrian dynamics. Physical Review E, Vol. 51, No. 5, pp. 4282-4286.

Iocchi L., Nardi D., Piaggio M. & Sgorbissa A. (2003) A Distributed Coordination in

Heterogeneous Multi-Robot Systems. Autonomous Robots, 5, 155-168.

Kalman R., (1996) A New Approach to Linear filtering and Prediction Problems. Transaction on the ASME-journal of Basic Engineering, 82 (Series D), pp. 35-45.

Kirkland J. and Maciejewski A., (2003) A Simulation of Attempts to Influence Crowd Dynamics. IEEE Int. Conference on Systems, Man, and Cybernetics, pp. 4328-4333, Washington, DC, 5-6.

Martinez E., Ohya A., Yuta S., (2003) Recognition of people's positioning by multiple mobile robots for humans groups steering, In Proceedings of Computational Intelligence in Robotics and Automation, Kobe Japan, pp. 758-763, 2003.ISBN 0-7803-7866-0/03.

Martinez-Garcia E,, A., Ohya & S., Yuta, (2004) Multi-people Localization in by Multiple Mobile Robots: First Approach for Guiding a Group of Humans, International Journal of Advanced Robotics Systems, Vol.1, No.3, pp.171-182.

Martinez-Garcia E., Ohya A. & Yuta S., (2005) A multi-robot system architecture communication for human-guiding, to appear in the Journal of Engineering Manufacture Part B1, Vol.256, January.

Maybeck P., (1979) Stochastic models, estimation, and control. Vol. 1, Academic Press, Inc. LTD.

Murphy R., (2000) Introduction to AI Robotics, The MIT Press, ISBN 0-262-13383-0, Massachusetts Institute of Technology.

Nakazawa A., Kato H. & Inokuchi S. (1998). Human tracking using distributed vision systems. 14th International Conference on pattern recognition, pp. 593-596, Brisbane, Australia.

Nourbakhsh I., Bobenageb J. & Grangec S., Ron Lutzd, Roland Meyerc and Alvaro Sotoa, (1999) An Affective Mobile Robot Educator with a Full-time Job, Artificial Intelligence, Vol. 114, No. 1-2, October, pp.95-124.

Thrapp R., Westbrook C. & Subramanian D., (2001) Robust localization algorithms for an autonomous campus tour guide, In Proceedings of International Conference on Robotics and Automation, Vol. 2, pp. 2065- 2071.

Tsutsui H., Miura J. & Shirai Y. (2001). Optical flow-based person tracking by multiple cameras., In Proceedings of IEEE Int. Conf. On Multisensor Fusion and Integration in Intelligent Systems, Baden-Baden, Germany.

Watanabe Y. & Yuta S.,(1990). Position estimation of mobile robots with internal and external sensors using uncertainty evolution technique. In proceedings of IEEE Int. Conf. Robotics and Automation., Cincinnati, OH., pp. 2011-2016.

Welch G. & Bishop G., (2002) An Introduction to the Kalman Filter. UNCChapel Hill, TR 95-041.

Yoshida T., Ohya A. & Yuta S., (2003). Cooperative self-positioning system for multiple mobile robots. In Proceedings of IEEE International Conference on Advanced Intelligent Mechatronics, pp. 223-227.

Sharing and Trading in a Human-Robot System

Kai Wei Ong, Gerald Seet & Siang Kok Sim

1. Introduction

With the functions of physical robots now extended beyond academia into factories, homes and fields, the interactions between humans and robots have become increasingly extensive and ubiquitous (Haegele et al. 2001). The current state of human interaction with robots in comparison to simple "machines" that operate in structured environment, such as manufacturing automation, is quite different. Robots differ from simple machines in that they are mobile. Some may be autonomous and their actions are not predictable in advance. Hence, there is a need to look into different interaction roles between humans and robots. The issue of interaction roles is an emerging research area in robotics namely Human-Robot Interaction (HRI) (Murphy and Rogers 2001). HRI can be broadly defined as "the study of the humans, robots, and the ways they influence each other" (Fong et al. 2001b). To provide realistic experimental settings, researchers working in this area need to develop Human-Robot System (HRS) to facilitate the study of HRI (Murphy and Rogers 2001). Here, HRS is defined as a "mixed system in which both human and physical robot interact, each as a cooperative intelligent entity" (Hancock 1992).

In the context of HRI, an important concern is how human and robot cooperate in a HRS (Sheridan 1992; Murphy and Rogers 2001). In remote operation applications such as space explorations, military operations, automated security, search and rescue, etc., the human does not have direct visual awareness of the environment to perform the required tasks. In these applications, a tight interaction between the human and the robot is required for effective cooperation and coordination. This raises an interaction dilemma: on one hand the robot operating in the remote environment can be expected in a "better position" to advise/inform the human regarding navigation issues (i.e. react locally to the remote environment) and refuses consent to dangerous human commands (e.g. running into obstacles); on the other hand, due to its limited ontologies, the robot requires human assistance on tasks such as object recognition, decision-making, and so forth. Here, limited ontology means that the robot is not able to use constraints either from its knowledge-base or from the environment to control its unspecified parameters.

To overcome the above dilemma, adapting to appropriate roles that exploit the capabilities of both human and robot as well as crafting natural and effective modes of interaction are important to create a cooperative HRS. To this end, innovative paradigms have been proposed over the years to redefine the roles of human and robot from the traditional master-slave relationship (Hancock 1992; Sheridan 1992), such as to model the human as cooperator (e.g. Lee 1993; Bourhis and Agostini 1998; Fong et al. 2001b; Hoppenot and Colle 2002; Bruemmer 2003) rather than just as the master controller of the robot. On the other hand, the slave robot is modelled in such a way that it becomes an active assistant (e.g. Bourhis and Agostini 1998; Hoppenot and Colle 2002) or partner (e.g. Fong et al. 2001b; Lee 1993; Bruemmer 2003) of the human, supporting perception and cooperative task execution. To design a cooperative HRS based on the above paradigms, a basic research issue is to consider how to achieve cooperation via appropriate degrees of sharing and trading between human and robot which constitutes the main focus of this paper.

1.1 Definition of Sharing and Trading

It might be useful to first provide a working definition of sharing and trading as a basis for further discussion. In Webster's dictionary (Agnes 2003), "sharing" and "trading" are defined as: "to join with another or others in use of some (thing)" and "to exchange one (thing) for another" respectively. Here, "to join" means that both the human and the robot work together through the use of some "thing" to ensure the success of task performance; and "to exchange" means that both the human and robot give and receive an equivalent of "thing" which they own while working together. In the context of sharing and trading, the tasks are the actions both the human and robot undertakes to achieve their goals. Human, needs to be able to see those tasks, adjust them and add to them if necessary during sharing and trading. On the other hand, the robot needs to be equipped with the capability to scale its own degree of autonomy to meet with whatever level of input from the human. To facilitate this, both human and robot must adopt the same ontologies so as to prevent miscommunication when they share and trade.

1.2 Why Sharing and Trading?

Within the discipline of robotics, the concept of sharing and trading is widely used for incorporating the strengths of human and robot. The aim is to achieve mutual compensation of both the human's and the robot's individual weakness (Sheridan 1992; Hirzinger 1993; Lee 1993; Bourhis and Agostini 1998;; Fong et al. 2001b; Hoppenot and Colle 2002; Bruemmer 2003).

For instance, sharing of control and sharing of autonomy has often been described in both the literature of telemanipulation (Sheridan 1992; Hirzinger 1993; Lee 1993) and teleoperation of mobile robot (Bourhis and Agostini 1998; Fong et al. 2001b; Hoppenot and Colle 2002; Bruemmer 2003). In telemanipulation, an example of sharing is the manipulation of a task where the compliance control is done by the robot automatically while position control is achieved by human's manual control (Hirzinger 1993; Lee 1993). In mobile robot teleoperation, an example of sharing of control is described as follows: the human directly controls the robot on board pan-tilt-zoom camera to provide a movement direction, i.e. to provide perceptual guidance; and the robot will respond to the human command by scaling its autonomy to drive the mobile platform according in the direction of the gaze (Hoppenot and Colle 2002). In both cases, trading is normally used in conjunction with sharing to let human and robot assist each other via the exchange of control and task information when both have problem performing the assigned task (Sheridan 1992; Lee 1993; Kortenkamp et al. 1997; Bourhis and Agostini 1998; Fong et al. 2001b; Bruemmer 2003).

The basic questions in sharing and trading are as follows (Sheridan 1992): In sharing – "Which tasks should be assigned to human and which to the robot?" In trading – "Which aspects of the tasks to trade, and when should control be handed over and when should it resume control during task execution?" As a consequence, researchers from the domains of telemanipulation (e.g. Hirzinger 1993; Lee 1993) and mobile robot teleoperation (e.g. Bourhis and Agostini 1998; Fong et al. 2001b; Bruemmer 2003) have developed various novel robotics control architectures to address these questions. Although their solutions are application specific, the fundamental principles are similar, that is, to facilitate interactive task allocation and cooperative decision-making between human and robot. The purpose of interactive task allocation is to spatially/temporally distribute the task to the human and/or robot, based on their intellectual capabilities and performance during task execution. The purpose of cooperative decision-making is to provide for arbitration/fusion of task commands from the human and the robot.

Although the concept of sharing and trading has been widely adopted and studied, it is far from fully developed. This is due to the progressive introduction of more intelligent and autonomous robots equipped with powerful and versatile mechanisms for interacting with humans.

The nature of the above problems provides a wide range of "interaction" space to consider how human and robot might share and trade to achieve cooperation. In particular, it is important to address the role of sharing and trading in accordance to humans' interacting with current state of autonomous robots. To understand how human and robot share and trade in a HRS, it is important to first identify the basic requirements which constitute sharing and trading. The aim is to present the classifications in a framework to assist in the design and development of a cooperative HRS.

This paper is structured as follows. Section 2 discusses the current HRS and what essential requirements constitute in the design and development of a HRS. The purpose is to serve as a basis for the discussion of sharing and trading in the following sections. Based on the concept of task allocation, Section 3 describes the concept of sharing and trading in designing HRS. Here, sharing and trading are eminent to explain the cooperation between human and robot. Subsequently, to illustrate the concept of sharing and trading on the design and development of a HRS, a case study is presented in Section 4.

2. Human-Robot System

Current HRS takes many forms. This can range from manually controlled system, such as teleoperation (Sheridan 1992) to autonomous robotics system that employ artificial intelligence, machine perception, and advanced control (Giralt et al. 1993). A simple illustration of this spectrum is presented in Table 1.

Six types of HRS and their applications are depicted in Table 1, presented in order of increasing robot autonomy/intelligence. Type 1 represents traditional master-slave teleoperation system. Type 2 represents teleoperation system that employs video technology, computer technology and force feedback. This facilitates a finer-gain of control (as compared to Type 1) for performing more complex/intrinsic tasks. Type 3 represents an advanced form of teleoperation, called telerobotics. As compared to Type 1 and 2, the robot is not directly teleoperated throughout the whole work cycles, but can operate in autonomous or semi-autonomous modes depending on the situation context. Type 4 is another form of Type 3 configuration with an important difference: the human located on the robot mobile base (e.g. the wheelchair), has direct visual awareness of the robot environment. Type 5 represents a highly autonomous and intelligent robotics system that has the capability to work cooperatively with humans. Finally, Type 6 represents fully autonomous robotic system that can operate without any human guidance and control.

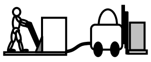
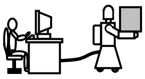
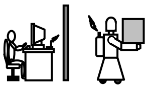
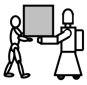
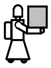
Human-Robot System	Descriptions	Possible Applications
Type 1: Teleoperation System (not computer-aided) 	The human is located remotely from the robot via the use of electric cable. However, the robot is directly controlled by human supervisor's own visual senses (line of sight). The robot extends the human's manipulation capability to a remote location so that he can work safely from the hazardous environment.	Underwater cleaning of reactor vessels, pipe inspection, etc. in nuclear power industry (Roman 1993).
Type 2: Teleoperation System (computer-aided) 	An extension of Type 1, but the human controls the robot through artificial sensing, computer, and displays. The robot extends both the human sensing and manipulation capabilities.	Robotics Surgery (e.g. the Da Vinci™ Surgical System, Thieme 2002), underwater operation (Roman 1993), etc.
Type 3: Telerobotics System (an advance form of teleoperation system) 	An extension of Type 2, but the human and the robot are separated by a barrier (environment, distance, time, etc.) that prevents direct interaction. The robot is normally equipped with high level of intelligence (such as safe navigation, path planning, etc.) while receiving higher-level instructions from the human instead of exercising continuous manual control as in Type 1 and 2.	Space exploration (Pedersen 2003), military operation (Gage 1995), automated security (Gage and Hower 1994), search and rescue (Casper and Murphy 2003), etc.
Type 4: Intelligent Mobility System 	A variant of Type 3, but the human and the robot are located closed together.	Rehabilitation, such as intelligent wheelchair (Bourhis and Agostini 1998) or mobility support system (Wasson and Gunderson 2001)
Type 5: Work Partner 	Robot is equipped with powerful and versatile mechanisms to communicate, interact and cooperate with human in a natural and intuitive way.	Robot as work assistants in factories, caretaker in home, etc. (Haegele et al. 2001)
Type 6: Autonomous Robot 	Robot replaces the human and performs the desired tasks autonomously.	iRobot Roomba Intelligent vacuum cleaner, tour guide, etc. (Burgard 1998)

Table 1. Different types of Human-Robot Systems

To deploy robotics technology effectively in a HRS, it is necessary to have a thorough understanding of the work environment and the tasks to be performed by the robot as well as to understand the nature of interactions between the human and the robotics system. Depending on the application settings, the work environment can be designed and engineered to facilitate the interactions between human and robot. For example, in the application of surgery (Thieme 2002), both the human and robot perform the tasks in a structured environment. On the other hand, in planetary surface explorations (Pedersen 2003), the work environment is unstructured (i.e. partially or entirely unknown beforehand). In unstructured environments, it is not feasible to preprogram the tasks of the robot because the environment is only known after the actual execution of the task (Giralt et al. 1993). This poses a great difficulty to the interactions between human and robot when performing the task.

Here, in order to facilitate HRI, the following requirements are considered:

- - *Methods of Control:* This determines how the robot is being commanded and controlled in a HRS from the perspective of human interacting with the robot (Sheridan 1992; Murphy and Rogers 2001). This is discussed in Section 2.1.
- Robot Autonomy: This determines the required degree of robot autonomy in a HRS from the perspective of robot interacting with the human (Jackson et al. 1991; Giralt et al. 1993). This consideration is directly related to the degree of human intervention (i.e. degree of control) required for the robot to perform a desired task. This is discussed in Section 2.2.
- Human-Robot Communication: This determines how human and robot communicate (Zhai and Milgram 1992; Klingspor et al. 1997; Fong et al 2001b; Green and Eklundh 2003). This consideration is discussed in Section 2.3.

2.1 Methods of Control

The roles of human in a HRS are application-specific (Sheridan 1992; Murphy and Rogers 2001). For example, the use of human's adaptive characteristics as a controller has a long history of providing a cost-effective method of increasing system reliability. The key question, over the last few decades, has been the role of human in the control of a system. Should he be an active, serial element in the control loop or should he be a supervisor monitoring the progress of the system (Curry and Ephrath 1976)? As a human is a necessary system element in the control loop, effective control method is important to determine how the human and robot interact to increase the system performance. HRI practitioners and researchers normally adopt certain models to guide the development of the system.

Their modelling approach can be described by certain metaphors that characterise the roles of humans and that of the robots in the system. All of these models are important, since each stresses a different aspect of HRI. An understanding of the nature of interactions of these models can lead to the identification and classification of different control methods. The roles and relationships of human and robot in the different types of HRS depicted in Table 1 are classified in Table 2.

In Table 2, between the extremes of master-slave relationship to that of a fully autonomous robot, there is a spectrum of control options involving humans as supervisors, partners and teachers of the robots. This is because, rather than wait for the results of research in achieving fully competent and autonomous intelligent systems, one way is to make use of the semi-autonomous control schemes (Sheridan 1992; Giralt et al. 1993) for humans to assist robots and to some extent robots to assist human (Bourhis and Agostini 1998; Fong 2001; Wasson and Gunderson 2001). The term "Semi-Autonomous Control" normally refers to an autonomous robot which can interact intelligently with a human, who might command, modify, or override its behaviour (Sheridan 1992; Giralt et al. 1993).

Classification of Roles and Relationships	Descriptions
Master-Slave (Type 1 and 2)	This describes the traditional teleoperation system (Sheridan 1992). The master-slave operation is the most basic form of control, where the human must always remain continuously in the control loop. The operating principle is simple; that is, human (master) has full control of the robot (slave), e.g. all the control decisions will depend on the human. When human stops, control stops.
Supervisor-Subordinate (Type 3 and 4)	Here, the robot does not simply mimic the human's movements as in the Master-Slave role. Instead, the worker robot has the capability to plan and execute all the necessary intermediate steps, taking into account all events and situations with minimum human intervention. On the other hand, the human as a supervisor divides a problem into a sequence of tasks, which the robot performs on its own (Sheridan 1992). If a problem occurs, the human supervisor is responsible for finding a solution and devising a new task plan.
Partner-Partner (Type 3-5)	Here, robot is viewed as the human's work partner and is able to work interactively with the human. Both the human and robot are able to take advantage of each other skills and to benefit from each other's advice and expertise (Bourhis and Agostini 1998; Fong 2001; Wasson and Gunderson 2001). As compared to the Supervisor-Subordinate, if a problem occurs, the robot may provide the necessary assistance to find a solution (Fong 2001).
Teacher-Learner (Type 3-6)	This assigns the human a primary role of teacher or demonstrator and assumes that the learning robot possesses sufficient intelligence to learn from him (Nicolescu and Mataric 2001). Once the robot is able to handle the task, it can replace the human completely or work together with the human depending on the context of the application.
Fully Autonomous (Type 6)	Here, the aim is to develop robotics system that has the capabilities to operate without any human intervention once the control is delegated to the robot (Giralt et al. 1993; Burgard 1998). This implies that the human can only monitor but not influence the robot operation. The only intervention is to stop the robot operation when a potentially serious error occurs.

Table 2. Different roles and relationships of human and robot

2.1.1 Semi-Autonomous Control

The solution for the concept of semi-autonomous control comes from two main stems (Murphy and Rogers 1996): the teleoperation concept (Sheridan 1992) and the autonomous robot concept (Giralt et al. 1993). According to Giralt et al. (1993), in the teleoperation concept, both human and machine interacts at the human operator station level. On the other hand, in the autonomous robot concept, the focus is to have on-board, in-built intelligence at machine level so that the robot can adapt its actions autonomously to the task conditions during HRI. Although the semi-autonomous control concept may emerge from the two mentioned stems, the basic objective remains the same. That is, in order to advance beyond simple human control of a robot there is a need to provide the robot basic competence and degree of autonomy (see Section 2.2). This leads to a reduction in the degree of supervision by the human (Sheridan 1992; Giralt et al. 1993).

Based on the roles and relationships shown in Table 2, Fig. 1 presents a hardware framework to illustrate the nature of the interactions between human and robot under different control modes in performing a task. The human cannot perform the task directly, but must perform the task via two main interaction loops. One loop defines the interaction between the human and the robot via an interface. The second loop defines the interaction between the robot and the task via its sensors and actuators. The "intermediary" that facilitates the interaction between these two loops is the control mode. Here, each control mode is viewed as a "task interaction mode" for human to interact with the robot in performing a task. Fig. 1(a) represents traditional master-slave manual control system (Type 1). Fig. 1 (b) represents indirect (i.e. with computer-aided) master-slave manual control system (Type 2). Fig. 1(f) represents autonomous control for fully autonomous robot (Type 6). Fig. 1(c) to 1(d) represents semi-autonomous control system (Type 3-5).

Semi-autonomous control can be further classified into parallel type, serial type or a combination of both parallel and serial types (Yoerger and Slotine 1987). In parallel type (Fig. 1(c)), both manual control and autonomous control operate at the same time. The parallel type is normally referred to as Shared Control, an approach to incorporate the strength of the human and robot by letting them control different aspects of the system simultaneously in situations that required teamwork (Arkin 1991; Papanikolopoulos and Khosla 1992; Sheridan 1992; Hirzinger 1993; Lee 1993; Krotkov et al. 1996; Bourhis and Agostini 1998; Fong et al. 2001a; Wasson and Gunderson 2001; Hoppenot and Colle 2002; Bruemmer et al. 2003). It is normally used in situations where the task is too difficult to be achieved by either the human (via manual control) or the robot (via autonomous control) alone.

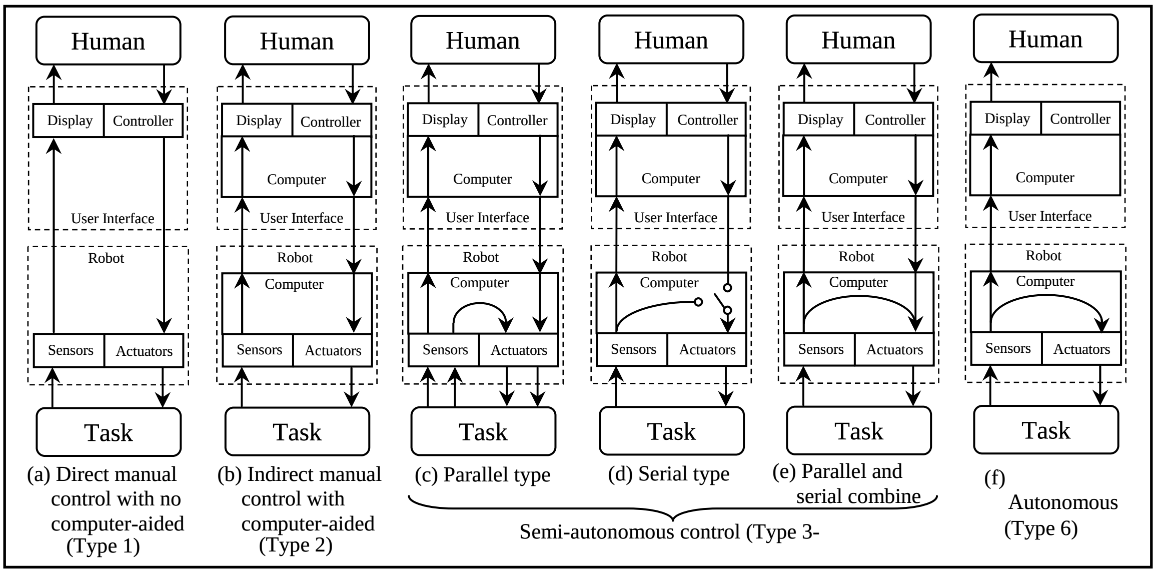
Shared control has been studied in different forms in both the domain of telemanipulation and teleoperation of mobile robot. The examples include position-compliance control (Hirzinger 1993; Lee 1993), vision-based perceptual guidance control (Papanikolopoulos and Khosla 1992; Hoppenot and Colle 2002), safeguarding control (Krotkov et al. 1996; Fong et al. 2001a; Wasson and Gunderson 2001) and behavioural control (Arkin 1991; Bourhis and Agostini 1998; Bruemmer et al. 2003).

In one way or another, all approaches have been based upon some form of coordination/fusion strategy with respect to the human inputs and the robot own assessment of the environmental task. As compared to manual control, shared control frees the human's attention from directly controlling nominal activities while allowing direct control during more perceptually intensive activities such as manipulation of parts (e.g. Hirzinger 1993; Lee 1993) and navigation in cluttered area (e.g. Bourhis and Agostini 1998; Bruemmer et al. 2003).

In serial type (Fig. 1(d)), either manual control or autonomous control can be selected as the operating mode at any one time. The serial type is normally referred to as Traded Control, a mutually exclusive approach for human and robot to exchange control on some basis (Papanikolopoulos and Khosla 1992; Sheridan 1992; Lee 1993; Kortenkamp 1997;

Bourhis and Agostini 1998; Bruemmer et al. 2003). The human and the robot can exchange control based on the demand of the task and the constraints of the environment due to goal derivations, addition/deletion of goals, modifications to the importance of goals/constraints, task completion, incompetence in performing the task and to veto dangerous commands/actions.

Basically, there are two perspectives on how control can be traded between human and robot in this context. In performing a navigation task (Bourhis and Agostini 1998; Bruemmer et al. 2003), the human may intervene and take the control from the robot (e.g. to give a new movement direction) if it moves in the wrong direction. On the other hand, the robot may override undesired commands (e.g. decelerates or stops) from the human, if the commands issue by the human may cause damage to itself. From this perspective, this control mode may allow both human and robot to "assist" each other in a partner-partner like manner.



(a) Direct manual control with no computer-aided (Type 1)

(b) Indirect manual control with computer-aided (Type 2)

(c) Parallel type

(d) Serial type

(e) Parallel and serial combine

Semi-autonomous control (Type 3-

(f) Autonomous (Type 6)

Figure 1. A spectrum of control modes ((Fig. (a), (b), (e) & (f) are adapted and modified from Sheridan (1992) and (Fig. (c) & (d)) are adapted and modified from Yasuyoshi et al. (1993))

In the combined configuration (Fig. 1(e)), both serial and parallel types interact to an extent, where the subtasks within each mode may also be shared and traded (Sheridan 1992). A classical example is the sharing and trading of control in the aircraft autopilot system (Billings 1997). During the cruise phase, in order to engage the autopilot system, the pilot trades the control over to the controller.

While the autopilot system holds the altitude, the pilot may adjust the heading, thereby sharing control at the same time. A classical example of the combine type is the Supervisory Control (SC) based on the Supervisor-Subordinate role by Sheridan (1992). Another recent example is the Collaborative Control (CC), an extension of SC based on the Partner-Partner model by Fong (2001) for the teleoperation of mobile robot. According to Fong (2001), the essential difference between CC and SC; is it can adjust its method of operation based on situational needs so as to enable "fine-grained sharing and trading of control". Specifically, in a situation where the robot does not know what to do or is performing poorly, it has the option to give control (e.g., for decision making) to the human in that situation. In other words, CC may enable work to be dynamically allocated to the robot or the human throughout the task performance. A summary of the different types of control discussed above is presented in Fig. 2.

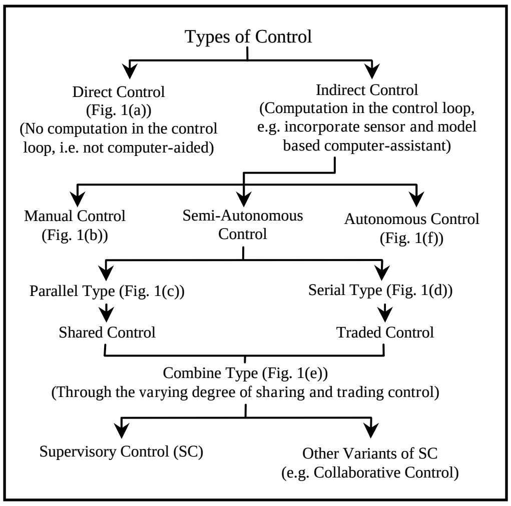

Types of Control
Direct Control (Fig. 1(a)) (No computation in the control loop, i.e. not computer-aided)
Indirect Control (Computation in the control loop, e.g. incorporate sensor and model based computer-assistant)
Manual Control (Fig. 1(b))
Semi-Autonomous Control
Autonomous Control (Fig. 1(f))
Parallel Type (Fig. 1(c))
Serial Type (Fig. 1(d))
Shared Control
Traded Control
Combine Type (Fig. 1(e)) (Through the varying degree of sharing and trading control)
Supervisory Control (SC)
Other Variants of SC (e.g. Collaborative Control)

Figure 2. A classification of different types of control in a Human-Robot System

2.1.2 Control Modes

To facilitate varying degree of sharing and trading of control in a HRS, an approach is to develop a control architecture that provide a fine range of control modes (i.e. from the continuum of manual control to autonomous control presented in Fig. 1) for the human to interact with the robot. The purpose of each control modes can be viewed as a strategy to realise particular operations (i.e. basic actions).

Adoption of a certain control strategy is required for adequate interaction and appropriate intervention. A control strategy can range from using abstract goal-oriented commands (i.e. high-level commands) to detail descriptions of the task. The choice of which control strategy to use is related to the type of communication format used (see Section 2.3.2), communication bandwidth available (see Section 2.3.3) and the complexity of the task. One reason for using abstract goal-oriented control strategy is to reduce the communication content in situations when the communication delay is high. Here, task is specified in a sufficiently high-level form (i.e. in terms of goals and constraints) where the robot performs the task on its own without constantly requesting guidance/assistance. Examples of high-level abstract goal-oriented commands are: follow the target, grasp the target, etc.

Clearly, to perform the task specified in this manner, the robot must have the required autonomy (see Section 2.2) to respond to unseen circumstances. In complex task, detailed descriptions of the task can be specified in a hierarchy manner based on the desired goal, e.g. by describing the robot direction, movement, traveling distance and so forth, in a stepwise manner.

Basically, most of the proposed control modes in the literature (e.g. Hirzinger 1993; Lee 1993; Kortenkamp et al. 1997; Bourhis and Agostini 1998; Bruemmer et al. 2003) have two important features: complementary and redundant. The control modes are complementary in order to let both the human and the robot contribute according to their expertise. The aim is to envisage a tighter cooperation, where the interactions are more mixed initiative to let both assist each other. On the other hand, the control modes are also redundant so as to provide more options for the human to develop strategies (i.e. via a sequences of control modes) to perform the task.

According to Callantine (1996), control modes have four basic characteristics: (1) Engagement Conditions – dictate when the mode will engage and encompass target values that must be set so the mode can attain and/or maintain them, and the modes that are currently in use; (2) Disengagement Conditions - that govern when the mode disengages. A mode may disengage when another mode is engaged, or when critical target value information no longer applies; (3) Operation Modifications – dictate the allowable modifications to operation that human or robot can make while the mode is engaged; (4) Control Properties – which include the specific set of parameters (e.g. speed, direction, etc.) that the mode controls, and the manner in which the mode controls them.

2.1.3 Control Mode Transitions

The characteristics of each control modes give rise to specific relationships between modes. Each control modes may have its own set of sub-modes, therefore the sub-modes of a given control modes can interact with the control modes of another. Hence, an important facet of control modes is mode transition. It determines when a particular control mode/sub-modes should be engaged or disengaged. According to Degani et al. (1995), a mode transition can result from three types of input: human initiated, robot initiated, or mixed initiated (i.e. from both human and robot).

An effective control mode transition will involve two important attributes, that is, monitoring and intervention.

Monitoring can be viewed as a precondition for intervention (Sheridan 1992). For example, once a task is delegated to the robot, the human must monitor the robot operation to obtain adequate feedback on its task performance so as to ensure that it is done properly. Adequate feedback can be achieved via observation to inspection, such as checking the robot agenda, reasoning, plan, etc. The observation can either be by direct viewing or mediated via a sensing device (see Section 2.3.1). If the robot encounters problems during execution, the human monitoring the situation will step in to update the commands or provide guidance to the robot. In cases where the errors cannot be recovered, the human may trade the control over, by stopping the operation and repairing the robot actions, e.g. via programming of new behaviours that are necessary to accomplish the task.

To classify the different levels of intervention, the three-level paradigm proposed by Rasmussen (1983), namely skill-based, rule-based and knowledge-based, is adopted. This paradigm is adopted because it is able to characterise both human and robot behaviours (Bourhis and Agostini 1998). For example, when a problem arises, the human or robot may simply use its sensory-motor actions (i.e. the skill-based behaviour) to react to the situation, or in known situation, standard operation/reaction procedure may be applied (i.e. the rule-based behaviour). On the other hand, if the situation is unknown to the human, he can use all his knowledge to evaluate the situation and make a decision from various goals (Sheridan 1992).

This can also be used to describe robot intervention behaviour. A good example is the application of remote operations where the robot situated at the remote environment is in a better position to give indication to the human if he executes the wrong commands (Bourhis and Agostini 1998; Fong et al. 2001b; Bruemmer et al. 2003). Another instance is the robot may trade the control over and execute autonomously in situation such as loss of communication.

Depending on the context of the situation, the intervention frequency can range from low to high. A problem in mode transition is the robot may not be able to keep up with the state of the world or of the task when the human intervenes and takes control over from the robot (i.e. during trading of control). This can make it difficult and dangerous for the robot to resume its operation once the task is delegated back to the robot by the human. This is because the robot's model of the world and of the task is inconsistent with the real state of the world (Kortenkamp et al. 1999). In addition, it is also difficult to know when control should be handed over to the robot and when it should be taken back (Sheridan 1992). To overcome this, the human and the robot must share some knowledge of the robot activities during task execution (Jackson et al. 1991). The human must understand the behaviours and the intention of the robot, if he wants to intervene to modify/change the mode (Bruemmer et al. 2003).

On the other hand, the robot must have the knowledge to interpret the human commands so as to respond to the control mode changes (see Section 2.2). In addition it must constantly update its knowledge-base so as to keep up with the real state of the world (Kortenkamp et al. 1999). This implies that it is important for both human/robot to develop a model of the interaction process based upon readily available interaction cues from each other so as to prevent mode confusion. Mode confusion (Bredereke and Lankenau 2002) arises when the mental model of the human does not match the model of the robot during HRI.

2.2 Robot Autonomy

To respond to the range of control modes and facilitate mode transitions, the robot must have the required autonomy to interact with the human. Here, the term autonomy is defined as "the ability of an agent (in this case, a robot) to act efficiently without any human's intervention" (Braynov and Hexmoor 2002). By stating that a robot is autonomous, it does not mean that the robot is thoroughly self-governing and capable of completing self-planning and self-control. However it can operate with some known (to the human) level of capability in the absence of human supervision/management for a defined period of time (Jackson et al. 1991).

Robot autonomy encompasses two basic attributes (Giralt et al. 1993): operating autonomy and decisional autonomy. Operating autonomy refers to the basic operational capability (i.e. the technological considerations) of a physical robot. For instance, to be "operational", a mobile robot must be equipped with the following basic components: Adequate sensors for navigation (e.g. range sensors for obstacles avoidance, detection, and location sensors to determine its own location), communication transceivers to interface with the human interface via a communication link, embedded computation and program storage for local control systems (e.g. to interpret commands from the human interface and translate these into signals for actuation).

Decisional autonomy refers to the level of intelligence imbued in a robot. This includes an internal representation of the world and of the task, and the capabilities to act reasonably in an unstructured/semi-structured environment. This encompasses the ability to reason about its own action, learn, and adapt to some extent on the basis of human feedback or from its own environment over a given period of time.

2.2.1 Robot Autonomy versus Human Control Involvement

Fig. 3 presents another view of describing the control modes in Fig. 1. The basic idea is to set up a discrete scale of robot autonomy, which enables the human to interact with the robot with different degrees of human control involvement. The horizontal axis represents the degree of robot autonomy, while the vertical axis corresponds to the degree of human control involvement.

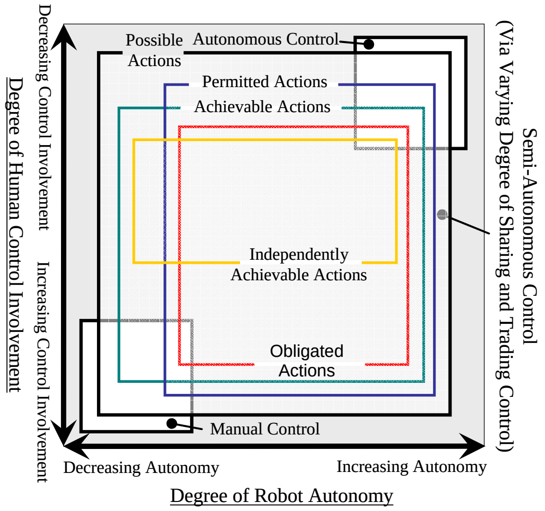


Figure 3. Control modes based on robot autonomy and human control involvement in accordance with varying nested ranges of action of robot

As shown in Fig. 3, the robot autonomy axis is inversely proportional to the human control involvement axis. Within these two axes, the manual control mode is situated at the bottom-left extreme, while the autonomous control mode is located at the top-right extreme. Between these two extremes is the continuum of semi-autonomous control. Within this continuum, varying degrees of sharing and trading control can be achieved based on varying nested ranges of action as proposed by Bradshaw et al. (2002a). They are: possible actions, independently achievable actions, achievable actions, permitted actions and obligated actions and are described in Table 3. Based on these five actions, constraints can be imposed so as to govern the robot autonomy within each level of control modes.

Ranges of Actions	Descriptions
Possible Actions	This refers to the theoretical maximum possible actions a robot can act with its given operating and decisional autonomy.
Independently Achievable Actions	This refers to a subset of possible actions that the robot could be expected to achieve independently with minimum human intervention.
Achievable Actions	This refers to a larger set of actions nested within the range of possible actions that could be achieved by the robot if it is able to work interactively with the human.
Permitted Actions	This refers to the actions nested within the range of possible actions that the robot is allowed to act (i.e. permitted by the human).
Obligated Actions	This refers to a subset of permitted actions that the robot is compelled to act.

Table 3. Degrees of autonomy based on varying nested ranges of action (adapted from Bradshaw et al. 2002a)

Another perspective of relating the degree of robot autonomy to human control is based on Sheridan's (1987) ten-level formulation of robot autonomy presented in Table 4. This formulation views the robot as a highly intelligent system that is capable of performing a whole task by itself in a given context. Here, the degree of robot autonomy is scaled accordingly based on human "decision" and "approval" when performing the task. Through this, a fine-grained presentation of a continuum of control between the robot and the human can be achieved.

Low ↑ Degree of Robot Autonomy ↓ High	1. Robot offers no assistance: the human perform the whole task	High ↑ Degree of Human Control ↓ Low
	2. Robot may assists by determining the multiple options of performing the task	
	3. Robot assists by narrowing down the options to a few, which human need not follow	
	4. Robot selects one action and the human may or may not approve	
	5. Robot selects action and implements it if the human approves	
	6. Robot informs and allows the human some time to veto task execution	
	7. Robot performs the task and necessarily informs the human what it did	
	8. Robot performs the task and informs the human what it did only if human explicitly requests	
	9. Robot performs the task and informs the human what it did, if it decides human should be informed	
	10. Robot does whole task autonomously	

Table 4. Ten-level formulation of robot autonomy (adapted from Sheridan 1987)

2.3. Human-Robot Communication

To ensure that the robot responds to the correct control mode when varying its own degree of autonomy, issues pertaining to Human-Robot Communication (HRC) is important. In Human-Human communication, humans communicate with each other easily through the same language. They can communicate effectively through electronic communication devices or face-to-face. However, in the case of HRC, it is not that straight forward, because the human cannot communicate with the robot directly. A well-defined communication channel is required to address the different modes of interactions between the human and the robot. Some of the basic considerations in HRC are: methods of communication, communication format, communication bandwidth and the purpose of communication as discussed in the following sections.

2.3.1. Methods of Communication

This relates to how information is transferred from the human to the robot (or vice versa). This issue is controversial because the current state of HRC encompasses a spectrum of methods, such as Personal Computer (PC) based control interfaces, Personal Digital Assistant (PDA) as interface devices (Fong et al. 2001b) and haptic interface which enables "drive-by-feel" (Fong et al. 2000) capability. In addition, methods such as speech and gesture (vision), that is analogous to human form of communication, are also widely used (Fong et al. 2000). The use of these methods is problem-specific or application-specific. However, regardless of the method used, effective communication exchange between the human and robot is paramount.

2.3.2 Communication Format

This pertains to the communication language used for information trading between the human and the robot. Zhai and Milgram (1992) proposed the notion of "continuous" and "discrete" languages as two different coding mechanisms to describe human-robot information trading.

According to Zhai and Milgram (1992), continuous language is used to represent information that is distributed continuously in quantitative or qualitative form, either along a spatial or a temporal dimension. In the context of robot communicating with human, examples include sending of raw sensors data, video images, etc. (i.e. perceived by the human).

In the context of human communicating with robot, examples include sending of continuous signal (e.g. via input devices such as joystick) to control the robot. On the other hand, discrete language is used to represent information which consists of separate or distinct elements.

Examples of discrete language are signs, symbols, written text, etc. used for communicating with the robot. As compared to continuous language, discrete language is normally used when the available information bandwidth is low or the communication delay is high. However, this implies that the robot must have sufficient autonomy (see Section 2.2) to perform the task.

A good example of using discrete language for HRC is through the use of dialog. The concept of using dialogue has recently received considerable research attention. Emerging from the research of mixed initiative artificial intelligent systems, it was subsequently adapted for HRC (e.g. Fong et al 2001b; Green and Eklundh 2003).

An example of dialogue adapted from (Green and Eklundh 2003) in defining a task during human intervention is as follows:

Human: Robot!
Robot: What is the task?
Human: Patrol Area A
Robot: Patrol Area A?
Human: Yes
Robot: Going to Area A

The idea of using dialogue is natural as it is very similar to human-human conversation. The purpose of confirming the human question (e.g. Patrol Area A?) is to ensure that the human has given the right command. If a wrong command is given, the human has a chance to correct his mistake. Using "confirmation" helps to prevent errors (i.e. giving wrong commands) and allows the robot to assist the human to learn from the mistake. Although this method is intuitive, it is difficult to decide how and when the robot should provide assistance or request for help. This issue is task specific and can only validate using human subject experiments.

2.3.3 Communication Bandwidth

This relates to the amount of HRC required to perform a given task. A good communication system is two-way (full duplex) with high data rate so that command data can be transferred from the human to the robot, and at the same time information of the robot can be conveyed back from the robot to the human.

The amount of communication can be quantified by the information quantity, measured in bits, and the information transfer bandwidth, measured in bits per second, of the messages that must be communicated between the human and the robot (Jackson et al. 1991). For instance, high communication bandwidth is normally required in manual control (such as teleoperation), because the human must control each movable function of the robot in real time. On the other hand, lower communication bandwidth is required in semi-autonomous control because continuous control of the robot is not required (see Section 2.1.1).

2.3.4 Purposes of Communication

This pertains to what type of information is shared and traded between human and robot during communication and what is the purpose of this information sharing and trading (Klingspor et al. 1997). In performing a task, the human must provide the robot with accurate information about the task to be performed.

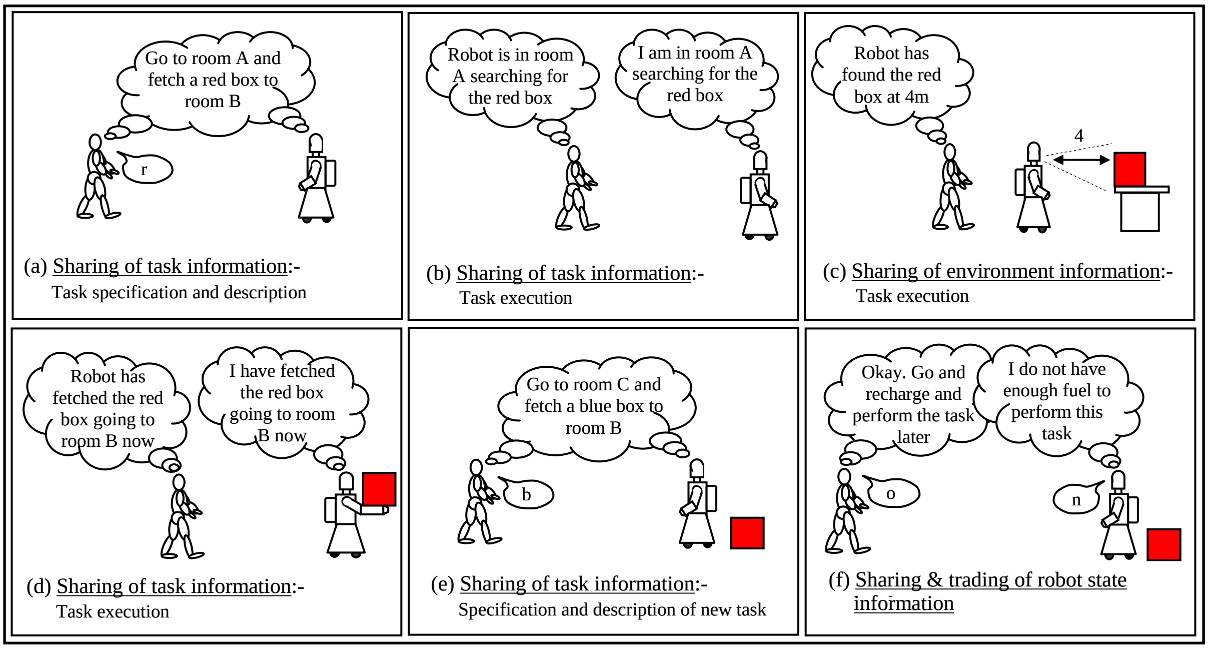
On the other hand, the robot should communicate to the human any information regarding its state and provide a feedback of the current status of the task to allow him to evaluate the robot task's successes and faults. In addition, it is important for the robot to convey any difficulty its encounters during the task (therefore needs human's assistance). A simple illustration of information sharing and trading between a human and a robot in a fetch-and-carry task is conveyed in Fig. 4.

The types of information presented in Fig. 4 are classified as follows (Scholtz 2002): task information, environment information and robot state information. In the context of human communicating with the robot, task information is the knowledge of the task as specified and described by the human to be performed by the robot (Fig. 4(a) & (e)). Task information is shared between the human and robot as follows: in Fig. 4(a) and (e), the human performs a communicative act 'r' (e.g. via any one of the communication method introduced in Section 2.3.1), addressed to the robot.

Through this, the following information is accessible to the human and the robot: 'r' means task specification (in this case the object to be handled, its location and destination), which are necessary for the task execution. By describing the task, the human provides the necessary instructions to the robot, about how to specify the task. Hence, the task information specified by 'r' is shared.

In the context of robot communicating with the human, task information is the knowledge of the robot with respect to the overall task defined by the human during task execution. This includes the robot's knowledge of its current location, its destination (Fig. 4(b)) and its next task execution decision (Fig. 4(d)).

Environment information consists of information in the robot's working environment (Fig. 4(c)). Examples of environment information are the objects (static or dynamic) in the environment and the robot's location relative to these objects. Robot state information is the information pertained to robot's status (e.g. speed, sensors status, health, etc.) and configurations (e.g. maximum sensing distance, available behaviours, etc.). In Fig. 4(b) – (d), information is shared between the human and robot via monitoring the execution of the tasks by the robot. Fig. 4(f) presents a scenario where information is shared and traded between the human and the robot.


Go to room A and fetch a red box to room B
r
Robot is in room A searching for the red box
I am in room A searching for the red box
Robot has found the red box at 4m
4
Robot has fetched the red box going to room B now
I have fetched the red box going to room B now
Go to room C and fetch a blue box to room B
b
Okay. Go and recharge and perform the task later
I do not have enough fuel to perform this task
o
n

(a) Sharing of task information:- Task specification and description

(b) Sharing of task information:- Task execution

(c) Sharing of environment information:- Task execution

(d) Sharing of task information:- Task execution

(e) Sharing of task information:- Specification and description of new task

(f) Sharing & trading of robot state information

Figure 4. An illustration of information sharing and trading between a human and a robot in a fetch-and-carry task

In this scenario, the robot takes the initiative to inform the human about its problem by performing a communicative act ‘n’, and the human responds to this communicative act by performing a communicative act ‘o’. Through this, the following information is exchanged between the robot and the human: ‘n’ means robot status (low fuel) and ‘o’ means “advises” (recharge and task specification). Hence, the meanings of ‘r’ (from the robot) and ‘o’ (from the human) is shared and traded. In fact, the robot may engage the human in communication at multiple task execution points to resolve differences in an entirely dialogue manner (see Section 2.3.2).

3. Towards Sharing and Trading in a Human-Robot System

It is proposed that a systematic approach to the design of a HRS can be based upon task allocation. That is “the assignment of various tasks either to humans or robots that are capable of doing those tasks” (Sheridan 1997). This perspective is based upon Fitts (1951) and is regarded by many as an essential component in systems engineering process (Sheridan 1997). In this quantitative approach, the attempt is made to identify which comparable capabilities are humans and machines “better at”, and subsequently analyse (e.g. “matching”) their best capabilities with aspects of the overall task at hand. This has come to be known as the “Fitts’ Men-are-better-at - Machines-are-better-at (MABA-MABA) List”. This list is often referred to as the first well-known basis for task allocation in the human factors literature (Hancock 1992; Sheridan 1997). Although this approach has gone though a sequence of different instantiations, e.g. published by Bekey (1970) and Meister (1982), the fundamental principle does not vary (Hancock 1992; Sheridan 1997). That is, the input for this approach is typically a list of abstract functions the HRS needs to achieve and the output is typically the same list categorised in terms of whether the human, robot, or some combination should implement the function (Hancock 1992; Sheridan 1997).

3.1 A Cooperative Human-Robot System

Although the MABA-MABA approach provides a formal and rational way for making allocation decisions, it has been criticised by many researchers (Hancock 1992; Sheridan 1997). The main concern is that there are large number of possible interactions between humans and robots for consideration, not simply just "human versus robots" (Bradshaw et al. 2002b; Hancock 1992; Jordan 1963; Sheridan 1997; Woods 2002). To develop a cooperative HRS, human and robot should be seen as the unit of concern rather than dichotomising them into separate unit (Jordan 1963; Hancock 1992; Sheridan 1997; Bradshaw et al. 2002b; Woods 2002). Jordan (1963) suggested that allocations of tasks between human and machines would only become useful if humans and robots were looked at as complementary, rather than comparable as in the MABA-MABA approach. He argued that this view is the key to optimise tasks allocation between human and robot. Sheridan (1989) shared the same view and stated that: "to cast the problem in terms of humans versus robots is simplistic, unproductive and self-defeating. We should be concerned with how they can cooperate". Bradshaw et al. (2002b) purport that the point is not to think so much about which tasks are best performed by humans and robots but rather how tasks can best be shared and traded by both humans and robots working together.

3.1.1 A Complementary View of Task Allocation

To provide a complementary view of how task can be allocated between human and robot, Woods (2002) proposed another perspective that does not concentrate on human shortcomings, called "Un-Fitts List". This is presented in Table 5 as summarised by Hoffman et al. (2002).

Robots			
Are constrained in that:		Need Human to:	
i	Sensitivity to context is low and is ontology-limited	i	Keep them aligned to context
ii	Sensitivity to change is low and recognition of anomaly is ontology-limited	ii	Keep them stable given the variability and change inherent in the world
iii	Adaptability to change is low and is ontology-limited	iii	Repair their ontologies
iv	They are not "aware" of the fact that the model of the world is itself in the world	iv	Keep the model aligned with the world
Humans			
Are not limited in that:		Yet they create robots to:	
v	Sensitivity to context is high and is knowledge- and attention-driven	v	Help them stay informed of ongoing events
vi	Sensitivity to change is high and is driven by the recognition of anomaly	vi	Help them align and repair their perceptions because they rely on mediated stimuli
vii	Adaptability to change is high and is goal driven	vii	Effect positive change following situation change
viii	They are aware of the fact that the model of the world is itself in the world	viii	Computationally instantiate their models of the world

Table 5. Woods' Un-Fitts List (adapted from Hoffman et al. 2002)

From the "Un-Fitts List", the important themes are the issue of sensitivity to limited ontologies and changing context, and how human can alleviate these deficiencies in robots; and also the emphasis on how robot can extend the capabilities of human and relieves the human load/task through appropriate task sharing (Bradshaw et al. 2002b). In other words, it looks into the possibilities of letting "human assist robot" (depicted in Table 5: i. to iv.) and "robot assist human" (depicted in Table 5: v. to viii.), which are important for understanding what can be shared and traded between human and robot in a HRS. From the "Un-Fitts List", the important themes are the issue of sensitivity to limited ontologies and changing context, and how human can alleviate these deficiencies in robots; and also the emphasis on how robot can extend the capabilities of human and relieves the human load/task through appropriate task sharing (Bradshaw et al. 2002b). In other words, it looks into the possibilities of letting "human assist robot" (depicted in Table 5: i. to iv.) and "robot assist human" (depicted in Table 5: v. to viii.), which are important for understanding what can be shared and traded between human and robot in a HRS.

3.1.1.1 Role of Ontology in Sharing and Trading

The "Un-Fitts List" shows that to allow human and robot share and trade effectively, they must adopt/share common ontologies. This implies that to establish what will be communicated during sharing and trading, they must share the same representation of the knowledge domain. There exist different kinds of ontology definitions in the literature, depending on the academic background of the researchers, e.g. from the field of philosophy, AI, knowledge engineering and so forth. In philosophy, ontology refers to the study of the kinds of things that exist. In AI, ontology refers to "the way in which a system conceives of the world external to itself, the internal representation of what is and what happens in the world" (Messina et al. 2001). In this field, it is used as basic construct in planning, learning, problem-solving, decision-making and communicating. A primary goal is to make knowledge sharable, by encoding domain knowledge using a standard vocabulary based on the ontology (Chandrasekaran et al. 1999).

A good reference that provides a vigorous analysis of the term "ontology" is from Guarino and Giaretta (1995). Here, the following definition for ontology is adopted: an explicit account or representation of some part of a conceptualisation (adapted from Guarino and Giaretta 1995). Generally, a conceptualisation is a world view/model corresponding to the working domain. In the context of sharing and trading between human and robot in a HRS, ontology is viewed virtually as the manifestation of a same perception while performing a task (e.g. same world view/model) that is agreed between the human and robot. Such "agreement" facilitates accurate and effective communication of task, environment and robot state information (see Section 2.3.4), which in turn facilitates the process of sharing and trading.

3.1.2 Task Sharing and Trading

An important observation from Section 3.1.1 is the trend of task allocation to task sharing and trading. This raises the next concern that is how to achieve cooperation via task sharing and trading. A good perspective of understanding this is from Sheridan (1992). He invoked the concepts of sharing and trading as distinct modes for task interaction between human and robot. These are further broken down into the sub-categories of extend, relieve, replace and backup and are conveyed in Fig. 5.

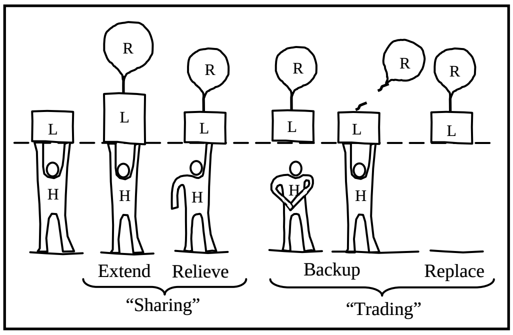


Figure 5. The notions of sharing and trading between human and robot. H the human, R the robot and L is the load or task (adapted and modified from Sheridan (1992))

Sheridan (1992) states: In sharing - "The robot can extend the human's capabilities beyond what he can achieve alone or it can partially relieve the human, making his job easier"; and in trading – "both the human and the robot can backup each other in cases where they fails, or the robot can replace him completely." These four simple interactive modes (extend, relieve, backup and replace) position the human and the robot in a number of situations where the division of task can be decided and deliberated. Hence, sharing involves extending the human capabilities and/or relieving the human, while trading involves letting both the human and the robot backup each other and/or letting the robot replace the human. The cooperation of human and robot can adopt different relative situations based on the above task interaction modes for them to share and trade control/autonomy (see Section 2.1.1 and 2.2.1); running from manual control (i.e. human replace robot) to autonomous control (i.e. robot replace human). Here, "control" and "autonomy" are placed within the context of a task representation.

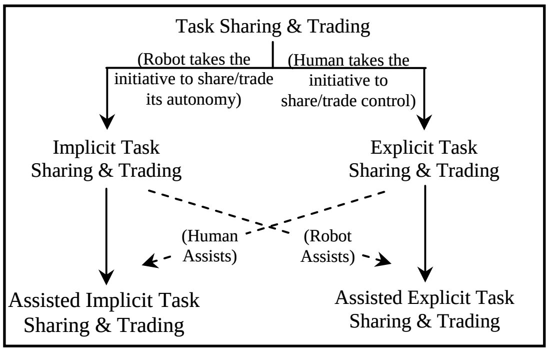

Task Sharing & Trading
(Robot takes the initiative to share/trade its autonomy)
(Human takes the initiative to share/trade control)
Implicit Task Sharing & Trading
Explicit Task Sharing & Trading
(Human Assists)
(Robot Assists)
Assisted Implicit Task Sharing & Trading
Assisted Explicit Task Sharing & Trading

Figure 6. A taxonomy of task sharing and trading

They are the basic elements that human and robot can share and trade with each other respectively to achieve task sharing and trading. If the robot takes the initiative to share/trade its autonomy, it is called an implicit task sharing and trading.

On the other hand, if the human takes the initiative to share/trade control, it is called an explicit task sharing and trading. However, there are also some instances where the human takes the initiative to share/trade control and the robot aid in selecting the desired control mode, in this case it is called assisted explicit task sharing and trading. If the roles are reversed, it is called assisted implicit task sharing and trading. The above discussion is summarised in Fig.6.

3.2 Defining Sharing and Trading in a Human-Robot System

Given Section 3.1, the basic **activities** within a HRS may consist of:

- Desired task as input task, IT
- Task allocated to the human, TH
- Task allocated to the robot, TR
- Output performance of both the human and robot, Op
- Task sharing and trading between human and robot, TS&T

These basic activities may be related as shown in Fig. 7.

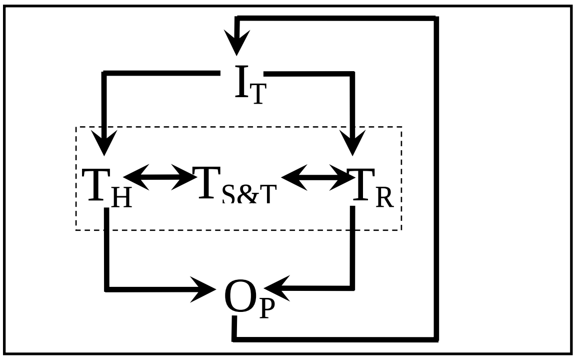


Figure 7. Activities within a Human-Robot System

In Fig. 7, it is suggested that there are three main paths to describe the activities within a HRS. The first path defines the input tasks (IT) allocated to either the human (TH) or the robot (TR). The second path represents the reallocation of task based on the output of the human-robot performance (OP). The reallocation can be based on the detection of 'off normal' or unexpected events. The third path, which is the focus of this paper, defines task sharing and trading (TS&T) between human and robot. That is, sharing and trading are based on current changing skills, intellectual capabilities and performance of both human and robot. This path is classified into sharing and trading of control, autonomy and information respectively to depict what can be shared and traded between the human and robot during TS&T. Here, it is defined that for human to perform TS&T with the robot, he must select the right control mode (see Section 2.1.2) to share and trade control with the robot. On the other hand, for robot to perform TS&T with the human, it must adjust to the right degree of autonomy (see Section 2.2.1) so as to respond to the selected control mode (i.e. sharing and trading its autonomy with the human). In both cases, to perform the appropriate actions (i.e. changes in control and robot autonomy), it invariably involves sharing of information (see Section 2.3.4). If the human and robot have different perceptions regarding the shared information, they must trade information to clarify any doubt before actual actions can be performed (see Section 2.3.4). In short, information sharing and trading is to find out what the other party is doing, what the intention of the other party might be and to resolve any conflict if it arises during task execution. To provide a clear roadmap, a classification of the above basic elements (i.e. control, autonomy and information) is presented in Table 6. They are classified in accordance to their associated attributes and features based on the discussion in Section 2.

Identifying what can be shared and traded above provides the basic construct for looking into the possible types of sharing and trading strategies to be implemented in a HRS. The basic approach adopted here is based on the perspective of how to let human and robot assist each other as depicted in Section 3.1.2. A good guide to envisage this perspective is the Un-Fitts List (see Section 3.1.1, Table 5). To understand how human and robot share and trade based on this perspective, it might be useful to understand the interactions or consequences of the human-robot team actions. In teamwork (Bradshaw et al. 2002b), where there is a collective responsibility, it is important to understand the team's collective actions. This then boils down to cooperation activities between team members. For human-robot team, the considerations are no longer just on robotic development but rather more complex interactive development in which both the human and robot exist as a cohesive team. To achieve such configuration, the role of the robot should be seen as an assistance or partner (see Section 2.1, Table 2). In the context of HRI, if the robot is merely envisaged as an autonomous entity (merely as a tool) capable of replacing a human, then the role of the human will only be to serve the need of the robot and to compensate its inadequacies. As compared to the latter case, the former is potentially much richer for the study of sharing and trading in HRI as it is aimed to develop "cooperative robot" to work closely with human, i.e. a paradigm of human assists robot – robot assists human.

Element	Attribute	Feature
Control	Mode (see Section 2.1.1)	manual-control … autonomous control
		complementary and redundant
		Characteristics: engagement conditions, disengagement conditions, operation modifications and control properties
	Strategy (see Section 2.1.2)	abstract description … detail description
	Transitions (see Section 2.1.3)	Human initiated, robot initiated, mixed initiated
	Monitoring (see Section 2.1.3)	observing … inspecting
	Intervention (see Section 2.1.3)	Purpose: commands … guidance … repair … stop
		Level: skills, rules-based and knowledge-based
		Frequency (situation-dependent): low … high
Autonomy	Type (see Section 2.2.2)	operating autonomy and decisional autonomy
	Variability (see Section 2.3.2)	fixed autonomy … adjustable autonomy
	Based on varying nested ranges of actions (see Table 3)	possible actions, independently achievable actions, achievable actions, permitted actions and obligated actions
	Degree	human maximum autonomy … robot maximum autonomy (e.g. Sheridan's ten-level formulation of autonomy, Table 4)
Information	Source (see Section 2.3.2)	continuous … discrete
		quantitative … qualitative
		Spatial … temporal
	Transfer (see Section 2.3.3)	Bandwidth: low … high
	Type (see Section 2.3.4)	Task information, environment information and robot state information

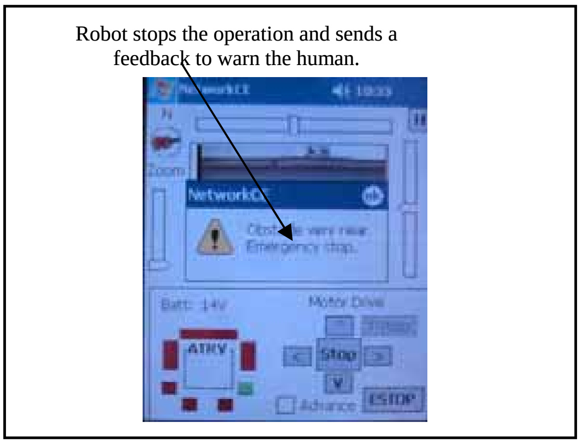
Table 6. A classification of the basic elements in $\mathbf{T}_{S\&T}$

3.2.1 A Paradigm of Human Assists Robot – Robot Assists Human

A basic consideration of this paradigm is that both the human and robot need to be aware of and understand one another's actions and intentions in order to assist each other. Here, we assume that both the human and robot share a common ontology (see Section 3.1.1.1) in HRC. To enable the robot to assist human, the robot needs to develop a model of the interaction process based upon readily available interaction cues from the human. This is to prevent any confusion during mode transition (see Section 2.1.3). Just as robots need to build a model to ensure effective interaction, it is also important for human to develop a mental model regarding the overall operation of a HRS (e.g. the operation procedures/process, robot capabilities, limitations, etc.), to operate the system smoothly.

A good guide in ensuring that the human is in effective command within a scope of responsibility is the principles from Billings (1997, pp. 39-48): the human must be involved in the interaction process, he must be informed of the ongoing events (to provide as much information as the human needs from the robot to operate the system optimally), he must be able to monitor the robot or alternatively, other automated processes (i.e. information concerning the status and activities of the whole system) must be able to track/know the intent of the robot in the system.

A good way to let the human know the intention of the robot is to ensure that, the feedback from the robot to the human indicates the "reason" for the invocation or initiation action during HRC (see Section 2.3.4, Fig. 4(f)). This implies that if the robot wants to override the human commands, the robot must provide clear indication for the human to know its intention to prevent any ambiguities. For example, during manual teleoperation, when the robot senses that it is in danger (e.g. colliding into an obstacle), the robot may stop the operation and send a feedback to warn the human in the form of a simple dialog as depicted in Fig. 8 (Ong et al. 2004).



Robot stops the operation and sends a feedback to warn the human.

Obstacle very near.
Emergency stop.

Figure 8. PDA dialog feedback (adapted from Ong et al. 2004)

3.2.2 Sharing and Trading Strategies

An intuitive approach of adopting the human assists robot – robot assists human paradigm for implementing sharing and trading strategies is based on the invocation of specific task events. It is possible to envisage a range of invocation events in accordance to the application tasks and invoke them based on the available information in the HRS. An advantage of this approach is that it directly addresses when sharing and trading occurs. From the extreme of initial task delegation to task completion, a spectrum of events can occur during task execution.

Within this spectrum, three types of events to invoke or initiate a TS&T process are distinguished. The first is termed goal deviations where the TS&T process would be invoked by human intervention. This highlights how human assists robot. The notion of goal here does not necessarily refer only to the goal of achieving a specific task, but also to the goal of attaining the overall task of the HRS. The word deviation refers to the departure from normal interactions between the robot and its task environment resulting in the robot being unable to achieve the goal. This also includes abnormalities arising during task execution. This may be due to either unforeseen changes in the working environment that cannot be managed by the robot or the robot itself; where an undesirable functional mapping from perception to action causes the robot to “misbehave” (e.g. due to sensing failures).

The second event is evolving situation in which the TS&T process would be invoked by the robot to veto human commands. This highlights how robot assists human. The types of robot veto actions can be loosely classified into prevention and automatic correction. Prevention implies that the robot will only impede the human actions but make no changes to it.

The human is responsible for correcting his own actions. An example is when the robot simply stops its operation in a dangerous situation and provides the necessary feedback to the human to rectify his commands. On the other hand, automatic correction encompasses prevention and rectification of human commands simultaneously. Depending on the task situation, the robot may or may not inform the human how to correct his actions.

For example, to prevent the human from driving into the side wall when teleoperating through a narrow corridor, the mobile robot maintains its orientation and constantly corrects the side distance with respect to the wall to align with it. In this case, the human may not be aware of this correction action and is able to drive seamlessly through the corridor. Based on Sheridan’s ten-level formulation of robot autonomy (see Section 2.2.1, Table 3), both prevention and automatic correction are positioned at level seven or higher. This is because it is the robot that judges whether the situation is safe or unsafe, as the human is unable to judge.

Finally, the third event is when both human and robot explicitly request assistance from each other. In such an event, the TS&T process between the two is mixed initiated, where each one strives to facilitate the individual activities in accordance to the task situation.

4. Discussion and Conclusion

To exemplify the concept of sharing and trading discussed in the preceding sections, a case study based on the application of security is presented to illustrate its working principles on the design and development of a HRS.

4.1 Application Case Study: An Automated Security HRS

Security has always been the fundamental issue in our present society to ensure the safety of our assets. In a typical security system, surveillance and intrusion are two essential elements. Normally, a range of different physical security devices and electronics surveillance systems are combined to automate surveillance and intrusion detection. Physical protection is provided by human security guards, security containers, locks, vaults and structural barriers such as multiple layers-fences, walls, and doors. Electronics surveillance is provided by interior and exterior intrusion detection sensors, access controls, closed circuit television (CCTV), alarm systems, lighting system and monitoring system. The key to a good physical security system is to have the above devices and systems tightly integrated and provide sufficient security controls and operating procedures to ensure safety of the assets. Although most of the security tasks, such as surveillance and intrusion detection, have been automated, human security guards are still required to perform physical security tasks such as patrolling and inspection in areas that cannot be covered by the security system. Using human security guards is not as efficient as most security tasks are very mundane most of the time hence automating them is highly desirable. For example, human security guards are required to be under constant alert while patrolling to look out for any intrusion or anomalous conditions such as fires. This tiring nature of the job is highly undesirable as it will cause fatigue and boredom. The current trend for job allocation in our present knowledge-based economy is to allocate high-level tasks such as unstructured decision-making to the humans while assigning automated machines to handle the low-level tasks that are repetitive and dangerous. This current trend is also applicable to the security system if it is automated. Therefore, one possible way to enhance the current security system is to automate the undesirable security tasks by replacing the human security guards with mobile robots (Gage and Hower 1994).

4.1.1 Task Allocation

The incorporation of mobile robots will change the operating principles of the conventional security system. Security tasks (i.e. IT) such as physical patrol and inspection in large premises originally performed by humans will be replaced by mobile robots (i.e. TR). From the control station, a human supervisor monitors, plans and supervises the operations of the mobile robots remotely (i.e. TH). A comparatively smaller number of human agents (i.e. human security guards) will render appropriate support to the robot agents at the remote site when necessary (i.e. TH). This scenario is reflected in Fig. 9; a testbed developed in our Robotics Research Centre (RRC) for studying HRI in a working environment that comprise of both humans and robots working together under human supervision (Ong et al. 2004). In accordance to the classification in Table 1, this testbed is a Type 3 HRS. The detail implementation of the testbed is discussed in Ong et al. (2004). For the purpose of this study, only the integration aspect between the human and robot is discussed.

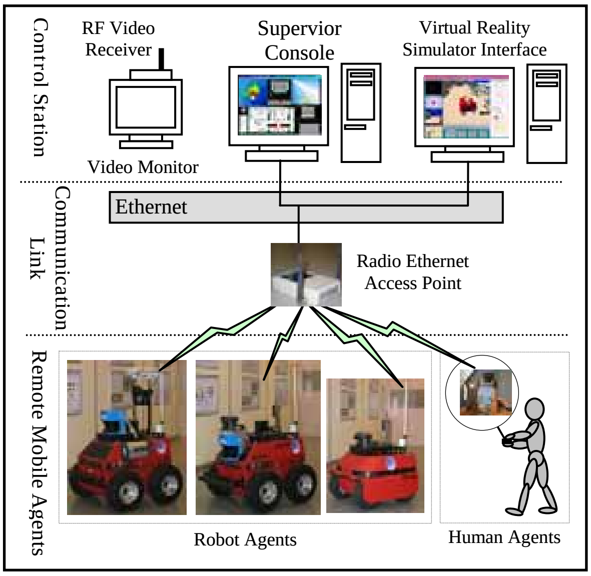


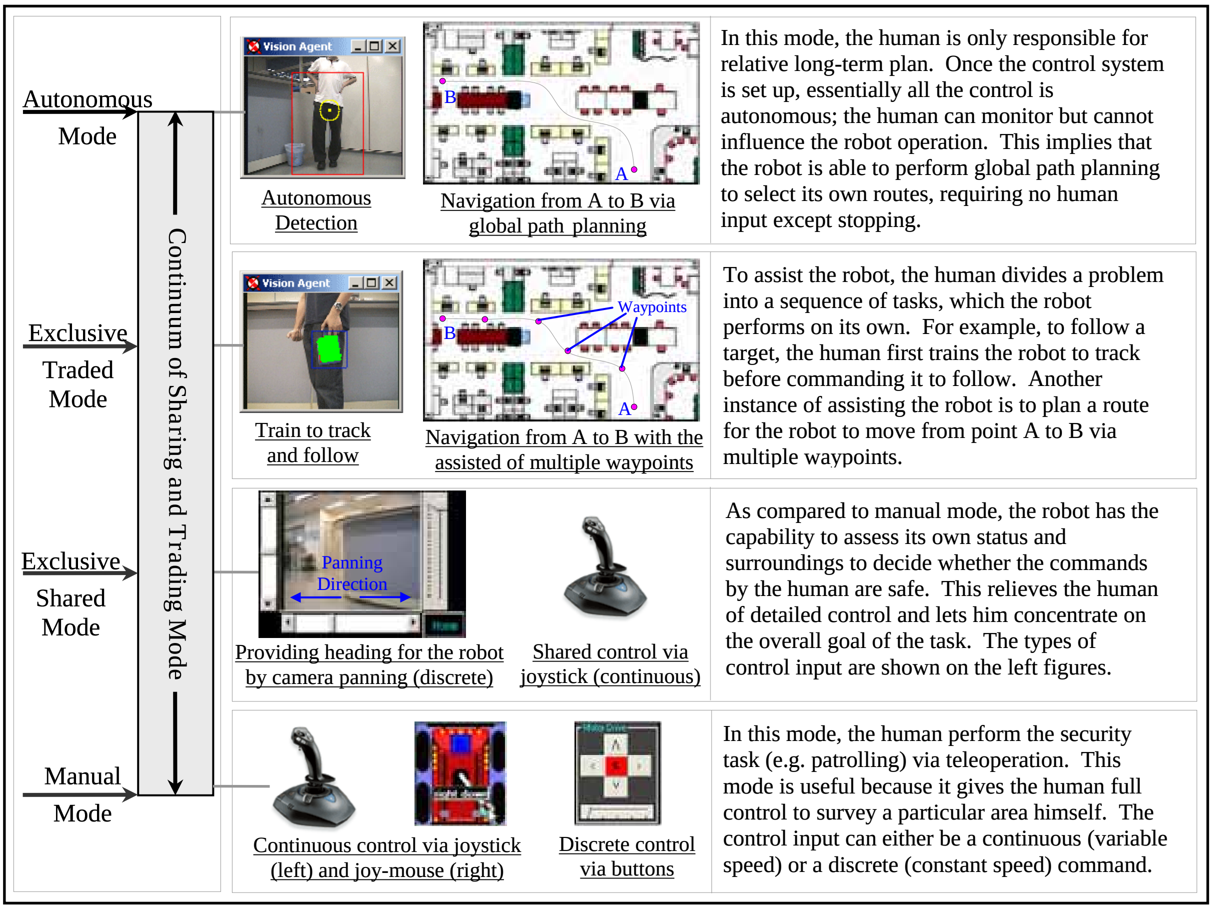
Figure 9. Block diagram of the HRI testbed (adapted from Ong et al. 2004)

4.1.2 Integration of Human and Robot

The process of integrating human and robot requires careful considerations from multiple perspectives. Possible considerations to this are the three basic requirements and issues pertaining to HRI as depicted in Section 2. They are method of control, robot autonomy and HRC that were discussed in Section 2.1, 2.2 and 2.3 respectively. Here, the primary approach of integrating human and robot is to consider how they shared and traded based on the paradigm of human assists robot – robot assists human as discussed in Section 3. Possible implementations of the sharing and trading strategies can be based on the guidelines and methods outlined in Section 3.1 and Section 3.2 respectively. The following section discusses these topics in more details.

4.1.1.1 TS&T between Human and Robot

The security task, e.g. "patrol area A", implies that the mobile robots must have the capabilities to perform the followings sub-tasks: navigation (including localisation), path planning, and intruder detection, to name a few. The main consideration is to achieve TS&T between human and robot. To support TS&T, the approach required is to develop a range of interaction modes from manual to autonomous for different situations (see Section 2.1). This is depicted in Fig. 10.



Autonomous Detection

Navigation from A to B via global path planning

In this mode, the human is only responsible for relative long-term plan. Once the control system is set up, essentially all the control is autonomous; the human can monitor but cannot influence the robot operation. This implies that the robot is able to perform global path planning to select its own routes, requiring no human input except stopping.

Train to track and follow

Navigation from A to B with the assisted of multiple waypoints

To assist the robot, the human divides a problem into a sequence of tasks, which the robot performs on its own. For example, to follow a target, the human first trains the robot to track before commanding it to follow. Another instance of assisting the robot is to plan a route for the robot to move from point A to B via multiple waypoints.

Providing heading for the robot by camera panning (discrete)

Shared control via joystick (continuous)

As compared to manual mode, the robot has the capability to assess its own status and surroundings to decide whether the commands by the human are safe. This relieves the human of detailed control and lets him concentrate on the overall goal of the task. The types of control input are shown on the left figures.

Continuous control via joystick (left) and joy-mouse (right)

Discrete control via buttons

In this mode, the human perform the security task (e.g. patrolling) via teleoperation. This mode is useful because it gives the human full control to survey a particular area himself. The control input can either be a continuous (variable speed) or a discrete (constant speed) command.

Figure 10. Examples of a range of interaction modes in the HRI testbed

As shown in Fig. 10, within the continuum of sharing and trading modes, two primary modes are defined. They are exclusive shared and traded modes to let human assist robot and robot assist human respectively. An important facet of these two modes is the ability of the robot to inform the human in situations where adherence to guidance would interfere with the pursuit of current goals, rather than blindly following the human's direction. Within these two modes, there exists a range of sub-modes using a combination of both arbitration and command fusion techniques for providing a finer gain of TS&T. How and when these sub-modes are triggered is based on the TS&T strategies outlined in Section 3.2.2.

4.2 Conclusion

Sharing and trading is important as it is an ubiquitous phenomenon in most robotics applications. As long as there are humans and robots working together within an application, there will always exists some form of sharing and trading among them. Therefore, this concept is adopted to address the issues of HRI in a Human-Robot System (HRS). Based on the discussion in the preceding sections, a framework is presented in Fig. 11 to conclude our approach of using the concept of sharing and trading for the design and development of a HRS. One of the major properties of our framework is to take into account the requirements of the human (Section 2.1), robot (Section 2.2), and the interactions between them (Section 2.3). This framework comprises of three phases: application requirements and analysis phase, human and robot integration phase, and the implementation and evaluation phase. The two main components in the application requirements and analysis phase, human and robot are discussed in Section 2.1 and 2.2 respectively, which are essential inputs to understand how human and robot share and trade. The essential inputs are the characteristics of human and robot. This encompasses the human and robot roles, responsibilities, robot functional requirements and their roles interactions in accordance to the task specifications. For the human and robot integration phase, the analysis of sharing and trading between human and robot are discussed in Section 2.3 and 3, which are in turn provides essential guidelines and methods for the design and development of a target HRS.

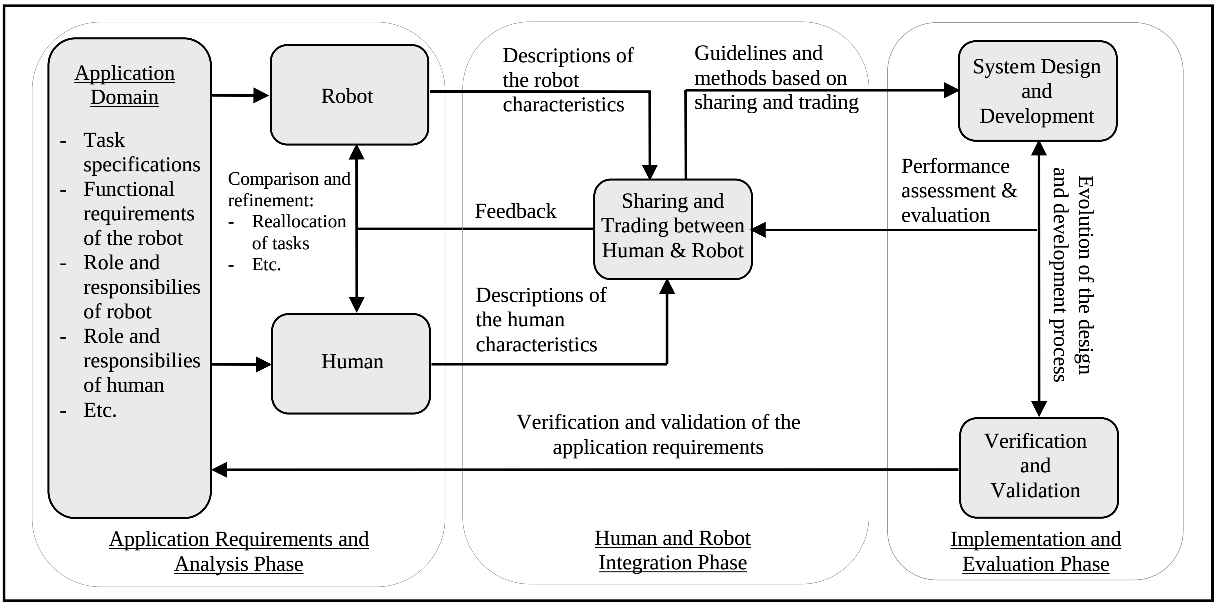

Application Domain
- Task specifications
- Functional requirements of the robot
- Role and responsibilities of robot
- Role and responsibilities of human
- Etc.
Robot
Human
Comparison and refinement:
- Reallocation of tasks
- Etc.
Descriptions of the robot characteristics
Guidelines and methods based on sharing and trading
Feedback
Sharing and Trading between Human & Robot
Descriptions of the human characteristics
System Design and Development
Performance assessment & evaluation
Evolution of the design and development process
Verification and Validation
Verification and validation of the application requirements
Application Requirements and Analysis Phase
Human and Robot Integration Phase
Implementation and Evaluation Phase

Figure 11. An application-driven framework for the design and development of a HRS

The reason of for outlining the framework in this way is to highlight the difficulties involved in this area. For example, to conduct research in this area, researchers need to deal with technical challenges such as achieving intelligent control, mobility and other special requirements of the robot while providing a seamless interaction between the human and the robot to enable useful communication exchanges in an effective and efficient way on a variety of levels. As discussed in Section 2, the recent advances in robotics, AI, and other disciplines have made robots more applicable to our current society thereby increasing the opportunities for humans and robots to work together in various ways. Hence, it is important that the design and development of a target HRS is followed by a contemporary in-depth understanding and knowledge of the related consequences and implications (i.e. scientific, technical, social and economic implications). In accordance with the approach and the conceptual framework outlined in this paper, the concept of sharing and trading may be able to shed some light on some of the related consequences and implications for addressing the fundamental issues pertaining to HRI in a holistic manner.

5. References

Agnes, M. E. (Ed.) (2003). Webster's New World College Dictionary, 4th Edition, Webster's New World, ISBN 0028631188.

Arkin, R. C. (1991). Reactive Control as a Substrate for Telerobotic Systems, IEEE AES Systems Magazine, Vol. 6, No. 6, pp. 24–31, ISSN 0885-8985, NY.

Bekey, G. A. (1970). The Human Operator in Control Systems. In: Systems Psychology, K. B. De Greene (Ed.), pp. 248-277, McGraw-Hill, ISBN 0-07-0162387, NY.

Billings, C. E. (1997). Aviation Automation: The Search for a Human-Centered Approach, Lawrence Erlbaum Associates, ISBN 0805821260, Mahwah.

Bourhis, G. & Agostini, Y. (1998). The Vahm Robotised Wheelchair: System Architecture and Human-Machine Interaction. Journal of Intelligent and Robotic systems, Vol. 22, No. 1, (May), pp. 39-50, ISSN 0921-0296.

Bradshaw, J. M.; Boy, G.; Durfee, E.; Gruninger, M.; Hexmoor, H.; Suri, N.; Tambe, M.; Uschold, M. & Vitek, J. (Eds.). (2002a). Software Agents for the Warfighter, ITAC Consortium Report, AAAI Press/MIT Press, Cambridge.

Bradshaw, J. M., Sierhuis, M., Acquisti, A., Feltovich, P., Hoffman, R., Jeffers, R., Prescott, D., Suri, N., Uszok, A. and Hoof, R. V. (2002b). Adjustable Autonomy and Human-Agent Teamwork in Practice: An Interim Report on Space Applications, In: Agent Autonomy, H. Hexmoor, C. Cristiano & R. Falcone, (Eds.), pp. 191-220, Kluwer, ISBN 1-4020-7402-6, Boston.

Braynov, S. & Hexmoor, H. (2002). Quantifying Relative Autonomy in Multiagent Interaction, In: Agent Autonomy, H. Hexmoor, C. Cristiano & R. Falcone, (Eds.), pp. 43-56, Kluwer, ISBN 1-4020-7402-6, Boston.

Bredereke, J. and Lankenau, A. (2002). A Rigorous View of Mode Confusion, Proceedings of SafeComp, S. Anderson, S. Bologna, M. Felici (Eds.), pp. 19-31, ISSN 0302-9743, Catania, Italy, Sept. 10-13, Springer Verlag.

Bruemmer, D. J.; Marble, J. L.; Dudenhoeffer, D. D.; Anderson, M. O.; & McKay, M. D. (2003). Mixed-Initiative Control for Remote Characterisation of Hazardous Environments, Proceedings of the IEEE 36th Annual Hawaii International Conference on System Sciences, pp. 127-135, ISBN 0-7695-1874-5, Waikoloa Village, Hawaii, Jan. 06-09.

Burgard W.; Cremers, A. B.; Fox D.; Hahnel D.; Lakemeyer, G.; Schulz, D.; Steiner, W. & Thrun, S. (1998). The Interactive Museum Tour-Guide Robot, Proceedings of the 15th National Conference on Artificial Intelligence, ISBN 0-262-51098-7, Madison, Wisconsin, July 26-30, AAAI Press.

Callantine, T. (1996). Tracking Operator Activities in Complex Systems, Ph.D. Dissertation, Industrial and Systems Engineering, Georgia Institute of Technology, US.

Casper, J. & Murphy, R. R. (2003). Human-Robot Interactions during the Robot-Assisted Urban Search and Rescue Response at the World Trade Center, IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics, Vol. 33, No. 3, pp. 367-385, (June), ISSN 1083-4419.

Chandrasekaran, B.; Josephson J. R. & Benjamins, V. R. (1999). What are Ontologies and Why Do We Need Them?, IEEE Intelligent Systems, Vol. 14, No. 1, (Jan/Feb), pp. 20-26, ISSN 1541-1672.

Cohen, P. R. (2000). Learning Concepts by Interaction, Technical Report 00-52, University of Massachusetts Computer Science Department.

Curry, R. E. & Ephrath, A. R. (1976). Monitoring and Control of Unreliable Systems, In: Monitoring Behaviour and Supervisory Control, T. B. Sheridan and G. Johannsen (Eds.), Vol. 1, pp. 193-203, Plenum Press, ISBN 030632881X, NY.

Degani, A.; Mitchell, C. M., & Chappell, A. R. (1995). Task models to guide analysis: Use of the operator function model to represent mode transitions, Proceedings of Eighth International Symposium on Aviation Psychology, R. S. Jensen (Ed.), Columbus, OH.

Fitts, P. M. (1951). Human Engineering for an Effective Air Navigation and Traffic Control System, National Research Council, Washington, D.C.

Fong, T. W.; Conti, F.; Grange, S. & Baur, C. (2000). Novel Interfaces for Remote Driving: Gesture, Haptic and PDA, SPIE Telemanipulator and Telepresence Technologies VII, ISBN 0-8194-3860-X, Boston, MA, November.

Fong, T. W. (2001). Collaborative Control: A Robot-Centric Model for Vehicle Teleoperation, Ph.D. Dissertation, Robotics Institute, Carnegie Mellon University, US.

Fong, T.; Thorpe, C. & Bauer, C. (2001a). A Safeguarded Teleoperation Controller, Proceedings of the IEEE International Conference on Advanced Robotics, ISBN 963-7154-05-1, Budapest, Hungary.

Fong, T.; Thorpe, C. & Bauer, C. (2001b). Collaboration, Dialogue, and Human-robot Interaction, Proceedings of the 10th International Symposium of Robotics Research, Lorne, Victoria, Australia, November 9-12.

Gage, D. W. & Hower S. P (1994). The MDARS Multiple Robot Host Architecture, Proceedings of AUVS-94, 21st Annual Technical Symposium and Exhibition of the Association for Unmanned Vehicle Systems, pp 133-136, Detroit MI, May 23-25

Gage, W. D. (1995). UGV History 101: A Brief History of Unmanned Ground Vehicle (UGV) development efforts, Unmanned Systems Magazine Vol. 13, No. 3, pp 9-16.

Giralt, G.; Chatila, R. & Alami, R. (1993). Remote Intervention, Robot Autonomy, and Teleprogramming: Generic Concepts and Real World Application Cases, Proceedings of the IEEE/RSJ International Conference on Intelligence on Intelligent Robots and Systems, pp. 27-34, ISBN 0780308247, Yokohama, Japan, July 26-30.

Green, A. & Eklundh, K. S. (2003). Designing for Learnability in Human-Robot Communication, IEEE Transactions on Industrial Electronics, Vol. 50, No. 4, (August), ISSN 0278-0046.

Guarino, N. & Giaretta, P. (1995). Ontologies and Knowledge Bases – Towards a Terminological Clarificiation, In: Towards Very Large Knowledge Bases – Knowledge Building and Knowledge Sharing, N. J. I. Mars (Ed.), pp. 25-32., IOS Press, ISBN 9051992173, Amsterdam.

Hancock, P. A. (1992). On the Future of Hybrid Human-Machine Systems. In: Verification and Validation of Complex Systems: Human Factors Issues, J. A. Wise, V. D. Hopkin and P. Stager (Eds.), NATO ASI Series F, Vol. 110, pp. 61-85, Springer-Verlag, ISBN 3540565744, Berlin.

Haegele, M.; Neugebauer J. & Schraft, R. D. (2001). From Robots to Robot Assistants, Proceedings of the 32nd International Symposium on Robotics, pp. 404-409, ISBN 89-88366-04-2, Seoul, Korea, April 19-21.

Hirzinger, G. (1993). Multisensory Shared Autonomy and Tele-Sensor Programming – Key Issues in Space Robotics, Robotics and Autonomous Systems, Vol. 11, pp. 141-162, ISSN 0921-8890.

Hoffman, R.; Feltovich, P.; Ford, K. M.; Woods, D. D.; Klein, G., & Feltovich, A. (2002). A Rose by Any Other Name… Would probably be Given an Acronym, IEEE Intelligent Systems, pp. 72-80, ISSN 1541-1672.

Hoppenot, P. & Colle, E. (2002). Mobile Robot Command by Man-Machine Co-Operation – Application to Disabled and Elderly People Assistance, Journal of Intelligent and Robotic Systems, Vol. 34, pp. 235-252, ISSN 0921-0296.

Jackson, E.; Williams, O. & Buchan, K. (1991). Achieving Robot Autonomy, Proceedings of the 3rd Conference on Military Robotic Applications, DND Canada, September.

Jordan, N. (1963). Allocation of Function between Man and Machines in Automated System, Journal of Applied Psychology, Vol. 47, pp. 161-165, ISSN 0021-9010.

Klingspor, V.; Demiris, J. & Kaiser, M. (1997). Human-Robot Communication and Machine Learning, Applied Artificial Intelligence, Vol. 11, pp.719-746, ISSN 1087-6545.

Kortenkamp, D.; Bonasso, R. P.; Ryan, D. & Schreckenghost, D. (1997). Traded Control with Autonomous Robots as Mixed Initiative Interaction, Proceedings of the *AAAI Spring Symposium on Mixed Initiative Interaction*.

Kortenkamp, D.; Burridge, R.; Bonasso, R. P.; Schrenkenghoist, D. & Hudson, M., B. (1999). An Intelligent Software Architecture for Semi-autonomous Robot Control, Proceedings of the *3rd International Conference on Autonomous Agents Workshop on Autonomy Control Software*.

Krotkov, E.; Simmons, R.; Cozman, F. & Koenig. S. (1996). Safeguarded Teleoperation for Lunar Rovers: From Human Factors to Field Trials, IEEE Planetary Rover Technology and Systems Workshop, Minneapolis, MN.

Lee, S. (1993). Intelligent Sensing and Control for Advanced Teleoperation, IEEE Control System Magazine, Vol. 13 (3), pp. 19-28, ISSN 0272-1708.

Meister, D. (1982). Behavioral Analysis and Measurement Methods, Wiley, ASIN 0471896403, NY.

Messina, E.; Meystel, A. & Reeker, L. 2001. Measuring Performance and Intelligence of Intelligent systems, Proceedings of the Performance Metrics for Intelligence System (PerMIS) Workshop, Mexico City, Mexico, Sept. 4.

Murphy, R. R. & Rogers, E. (1996). Cooperative Assistance for Remote Robot Supervision, Presence, Special Issue, Vol. 5., No. 2, pp. 224-240.

Murphy, R. R. & Rogers, E. (2001). Human-Robot Interaction, Final Report for Defense Advanced Research Projects Agency (DARPA)/National Science Foundation (NSF) Study on Human-Robot Interaction, Sept. 29-30.

Nicolescu, M. N. & Mataric, M. J. (2001). Learning and Interacting in Human-Robot Domain, IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans, Vol. 31, No. 5, pp.419-430, ISSN 1083-4427.

Ong, K. W.; Seet, G.; Sim, S. K.; Teoh, W.; Lim, K. H.; Yow, A. N. & Low, S. C. (2004). A Testbed for Human-Robot Interactions, To be Appeared in the Proceedings of ASME DETC'04 28th Biennial Mechanisms and Robotics Conference, Salt Lake City, Utah, Sept. 28 to Oct. 2.

Papanikolopoulos, N. P. & Khosla, P. K. (1992). Shared and Traded Telerobotics Visual Control, Proceedings of the IEEE Conference on Robotics and Automation, pp. 878-885.

Pedersen, L.; Kortenkamp, D.; Wettergreen, D. & Nourbakhsh, I. (2003). A Survey of Space Robotics, Proceedings of the 7th International Symposium on Artificial Intelligent, Robotics and Automation in Space, Nara, Japan, May 19-23.

Roman, H. T. (1993). Robotic Applications in PSE&G's Nuclear and Fossil Power Plants, IEEE Transactions on Energy Conversion, Vol. 8, No. 3, pp. 584-592, ISSN 0885-8969.

Scholtz, J. (2002). Evaluation Methods for Human-System Performance of Intelligent Systems, Proceedings of the Performance Metrics for Intelligence System (PerMIS) Workshop, E. R. Messina and A. M. Meystel (Eds.), Aug. 13-15.

Sheridan, T. B. (1987). Supervisory Control, In: Handbook of Human Factors/Ergonomics, G. Salvevely (Ed.), Wiley, ISBN 0471116, NY.

Sheridan, T. B. (1989). Telerobotics, Automatica, Vol. 25, No. 4, pp. 487-507, ISSN 0005-1098.

Sheridan, T. B. (1992). Telerobotics, Automation, and Human Supervisory Control, MIT Press, ISBN: 0262193167, Cambridge.

Sheridan, T. B. (1997). Task Analysis, Task Allocation and Supervisory Control, In: Handbook of Human-Computer Interaction (2nd, Completely Revised Edition), M. Helander, T. K. Landauer, P. Prabhu (Ed.), pp. 87-105, Elsevier, ISBN 0444818262, Amsterdam.

Thieme, T. (2002). At the Controls: Columbia Presbyterian Medical Center, New York, Popular Science.

Wasson, G. S. & Gunderson, J. P. (2001). Variable Autonomy in a Shared Control Pedestrain Mobility Aid for Elderly, IJCAI Workshop on Autonomy, Delegation and Control.

Yoerger, D. R. & Slotine J.E. (1987). Supervisory Control Architecture for Underwater Teleoperation, Proceedings of the IEEE International Conference on Robotics and Automation, pp. 2068-2073.

Yokokohji, Y.; Ogawa, A.; Hasunuma, H. & Yoshikawa, T. (1993). Operation Modes for Cooperating With Autonomous Functions, Proceedings of IEEE International Conference on Intelligent Teleoperation Systems Robotics and Automation, Vol. 3, pp. 510 – 515, May 2-6.

Zhai, S. & Milgram, P. (1992). Human Robot Synergism and Virtual Telerobotics Control, *Proceedings of the 25th Annual Conference of the Human Factors Association of Canada*.

-VII-

Service Robotics

A Robotic System for Volcano Exploration

Daniele Caltabiano & Giovanni Muscato

1. Introduction

Robovolc: "A Robot for Volcano Exploration" is the name of a project funded by the European Commission, whose activities started in 2000. The main purpose of the ROBOVOLC project has been the development and trial of an automatic robotic system to explore and perform measurements in a volcanic environment. The robot, shown in Fig.1, has been realized to minimize the risk for volcanologists who are involved in work close to volcanic vents during eruptive phenomena. The robot is capable of moving on very rough and unstructured terrain thanks to its six independently actuated wheels and to its articulated chassis; sampling lava rocks and gas using its SCARA manipulator; collecting measures of the explored environment using the onboard video camera, the IR camera, the still camera and the radar Doppler. Details on the projects are reported and updated on the web pages: http://www.robovolc.diees.unict.it
The project has required both robotic and volcanology expertises. The involved partners have a recognised know-how in Europe as regards Service robotics both for academic (Università degli Studi di Catania, Italy and University of Leeds, UK) and industrial aspects (BAE Systems, UK and Robosoft, France). The two research institutes involved (Istituto Nazionale di Geofisica e Vulcanologia, Italy and Institute de Physique du Globe de Paris, France) are among the most important research institutes on volcanology in Europe.

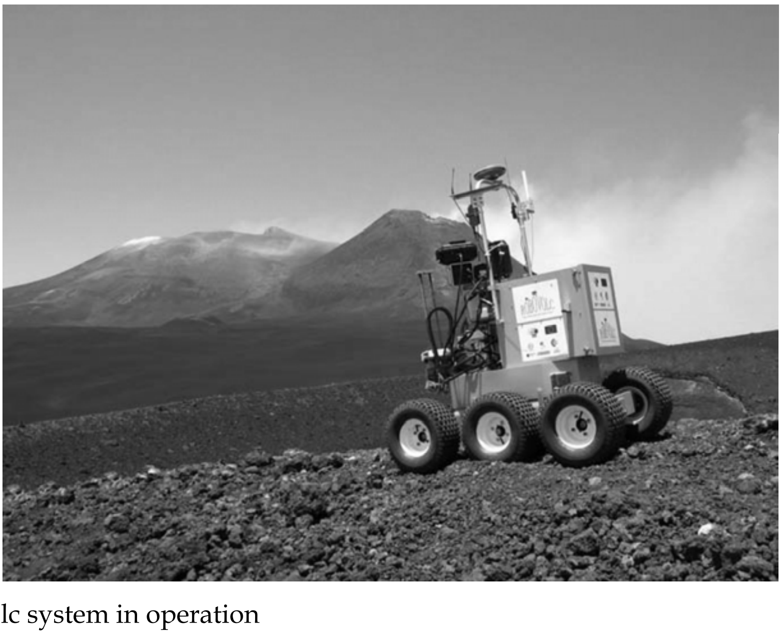


Figure 1. The Robovolc system in operation

The project required the complementary expertises that these partners together offered and that can not be found in a single country. The problem of volcano monitoring and forecasting is a large problem and is common to more than one European country which emphasises the need for pan-European co-operation in the RTD.

The realized robot is a useful tool to help volcanologists increasing their knowledge on volcanic activities especially during dangerous active phases. In this way the safety of people employed in collecting such a kind of measurement will considerably improve. Historically many scientists studying eruptions from unsafe places suffered serious injuries. Due to both the unpredictable timing and to the magnitude of volcanic phenomena, several volcanologists still die surveying eruptions.

The improvement in the working conditions for volcanologists that are directly involved in monitoring dangerous eruptive activity will enhance the systematic study of these phenomena for which many measurement data are not yet available. Another important aspect to be considered concerns the improvement in the anticipatory capability of the volcanic activity by way of continuous updating during eruptive phenomena, also when they become very dangerous for volcanologists. This could improve volcanic risk assessment contributing to an integrated risk management system for obtaining an almost real-time early warning, useful to Civil Protection authorities to inform and protect citizen from dangerous volcanic eruption consequences. This will lead to huge savings in potential losses caused by damage to buildings, land, equipment, livestock and injury to humans. From this point of view the Robovolc robot will contribute to achieve a very important social objective of the Community: the improving of the safety, and in general the quality of life, of the people living around active volcanoes located in the European countries. In volcanology the measurement and sampling processes in the proximity of active eruptive vents are fundamental. The most important kinds of measurement are: magmatic gas geochemistry, physical modelling of magma degassing, and stability assessment of the craters and domes:

Magmatic gas geochemistry: due to the rapid mixing between the gas released by the magma and atmosphere it is quite difficult to make accurate measurements of the quantity of some gas species produced by volcanoes that are abundant also in the atmosphere. In particular the CO2 released during eruptions could contribute significantly to the global warming of the planet. Accurate measurement of this gas during eruptions of basaltic magma, in which it is more abundant, will help to better discriminate natural and human activity contributions of the CO2 increase in the atmosphere.

Physical modelling of magma degassing: dynamics of the gas bubbles that rise up in the magma and disrupt at the surface drives all eruptions. Its modelling depends on the geophysical data collected close to the disrupting surface where the bubbles burst. This process is very frequent in active craters of the basaltic volcanoes where explosive activity is produced. Unfortunately it observation and measurement is often prevented by the funnel shaped geometry of the volcanic vents, so a very close approach with specific instrumentation (stereo cameras, Doppler-radar, etc.) is necessary to collect these data.

Stability assessment of the craters and domes: active volcanic crater and dome structures are subject to a rapid growth during an eruption and often collapse under their own weight and due to endogenous forces. Dome collapses produce very dangerous pyroclastic flows and surges. Crater collapses block the erupting vent and can produce large explosions due to gas overpressures inside. The measurement of the instability of craters and domes is hence very useful to forecast dangerous eruptive phenomena, however due to the unpredictability of these collapses fieldwork it is not possible without a robot.

The purpose of the project ROBOVOLC was then to build a vehicle capable to approach an active volcanic vent and to perform several kind of measurement keeping the operators at a safe distance. ROBOVOLC can be considered as a system constituted by two main parts: the rover and the measurement systems. The development of the rover involved a careful analysis of the requirements and of the environmental conditions. Volcanic terrain is one of the worst outdoor environments that can be found for a robot: rocks, steep slopes, very fine ash sand, snow, high temperatures close to the vents and sometimes, due to the high altitudes, very low temperatures. Moreover operations often must be carried out without a direct view of the system by the operators. The measurement system also required a great research effort in developing suitable instruments. Most of the operations to be performed by the robot were previously performed directly by a human operator on-site and consequently most of commercial instruments are not automatic.

In section 2 of this chapter some previous projects concerning robots used for volcano exploration are reported. In the next sections a description of the different phases of the project will be shown. In particular in section 3 some preliminary activities to understand the requirements and the specifications of the system are presented. Then a description of the designed and built sub-systems is reported. In Section 5 some details on the localisation sub-system are presented and finally in Section 4 some considerations on the field trials and results obtained are given.

2. Previous Projects on Robots for Volcanoes

In known, published literature, there is only one example of a robot specifically developed for volcano exploration, *Dante II*. Dante II is a multi-legged frame walking robot designed by NASA and Carnegie Mellon University to investigate live volcanoes and test robotic technology (Bares & Wettergreen,1999). The robot is a frame-walker with eight pantographic legs arranged in two groups of four, on inner and outer frames. A tension-controlled tether was connected to Dante II, to maintain stability and to allow rappelling on steep slopes. In 1994 Dante II underwent a trial exploration of Mount Spurr volcano in Alaska. For more than five days the robot explored alone in the volcano crater using a combination of supervised autonomous control, and tele-operated, control. The robot travelled one-quarter of its 165-m descent autonomously, relying only on on-board sensors and computers to plan and execute its motion. The terrain was very rough including crossing 1-m boulders on ash-covered slopes, navigating areas of deep snow, ditches and rubble. The robot measured the gas composition of several large fumaroles vents. However while climbing out of the crater, Dante II lost stability and fell on its side thus ending its mission. The Dante II/Mt. Spurr expedition was considered a success because of the amount of data and experience that was accumulated. Dante II was successful in retrieving data from a very harsh environment such as might be expected on other planets. This trial gave NASA valuable experience in determining what improvement considerations would be needed for future robotic missions. There are instead several examples of robot that have been designed for planetary exploration and that have been tested on volcanic sites (Guccione et al., 2000). In fact there are many similarities between volcanic terrain and many planetary sites. It is important to observe that not one of these robots has been totally developed in an EC country. An important example that can be cited is the *Marsokhod Planetary Rover*. The Marsokhod rover is an all terrain vehicle developed by the Mobile Vehicle Engineering Institute (VNIITransmash) in Russia for planetary exploration (Kemurdjian et al., 1992). The chassis (100cm wide, 150cm long, 35kg unloaded mass) consists of three pairs of independently driven titanium wheels joined together by a three degree of freedom passively articulated frame. This design enables the rover to conform passively to very rugged terrain. The amplifiers, motors and batteries are mounted inside the wheels to provide a very low centre-of-gravity. The robot can travel at speeds up to 12 cm/sec and can traverse obstacles up to 30 cm high and slopes up to 45°. The duration of operation with batteries is approximately 6 hours. The Marsokhod robot, originally designed for Mars exploration, has been extensively tested in volcanic environments such as in Kamchatka, Russia (1993), Amboy crater in California (1994) and Kilauea Volcano in Hawaii (1995). Kilauea Volcano was selected primarily for its great diversity of geological features similar to those expected on Mars and the Moon.

Finally, as an example of flying vehicles tested on volcanic environments, a *Yamaha RMAX helicopter* has been involved in a project for the surveillance of the Mt. Usu Volcano in the Hokkaido region of Japan. For this purpose a special version of the unmanned helicopter RMAX has been developed. Due to the large distances of operation an autonomous flight system has been developed and the cruise autonomy of the helicopter increased to 4km. In April 2000 the helicopter, equipped with four CCD cameras, was successful in performing several surveillance missions observing the hazards caused by volcanic sediment and debris flow (Yamaha, 2000).

3. Preliminary Activities

The lessons learnt from these previous machines and the advances made in robotic have been directly applied to the ROBOVOLC robot. The major innovations are in the mechanical structure and materials (lightweight, dust proof, heat and impact resistant), locomotion systems (intelligent control, robust traction for the harsh and unstructured environment), guidance (environmental mapping, intelligent path-planning, autonomous decision making) and sensors (integration of a variety of sensors for robust localization and environment reconstruction, an effective user interface). Several preliminary technical visits to volcanic sites (Etna, Stromboli and Vulcano) were organised and detailed discussions were carried out with the volcanologists, in order to investigate the requirements of the system. It was immediately apparent that the extremely rough and difficult volcanic terrain required the development of specific solutions.

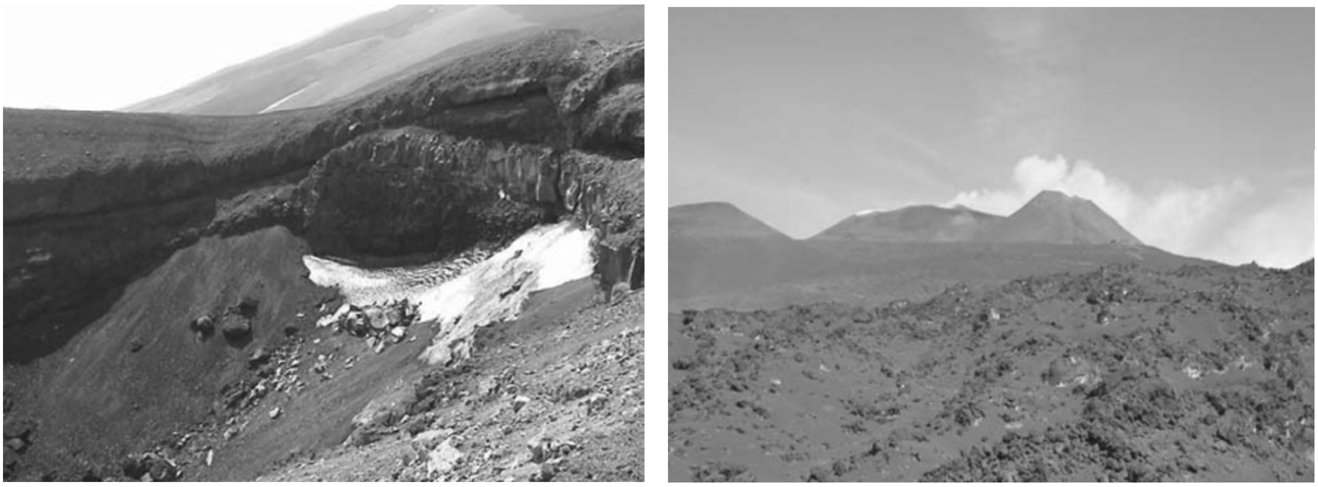


Figure 2. Examples of terrain found in volcanic environment

Ground surface change, varying from rough lava to rocky surfaces or sandy slopes, occur typically in a short distance. Then it was decided to concentrate our attention on specific kinds of missions that could be accomplished by the robot on specific types of terrain. A system capable to be useful in all of the possible situations exceeds current robotic capabilities.

Some prototypes of robots were also built in this phase of the project, also to explore the capabilities of different locomotion architectures. For example the WHEELEG robot (Fig. 3a) is a hybrid locomotion robot composed by two front pneumatic legs and two rear wheels (Guccione & Muscato, 2003), (Lacagnina et al., 2003). The M6 robot (Fig. 3b) is a six independently actuated wheels robot with a very articulated chassis (Lacagnina et al., 2002). The P6W robot (Fig. 4a) is a 1:4 scale prototype of the Robovolc rover that is used also to test the traction capability of Robovolc in laboratory, with a high degree of repeatability (Caltabiano & Muscato, 2002), (Caltabiano et al., 2004a).

In order to measure volcanic terrain features a four wheel probe was built. This cart, shown in Fig. 4b, was manually driven into quiescent craters, to simulate possible paths taken by a robot inside the craters. The probe carried the following instrumentation: a GPS, encoders in both wheels, a video camera, temperature, humidity and pressure sensors (Azad et al., 2001).

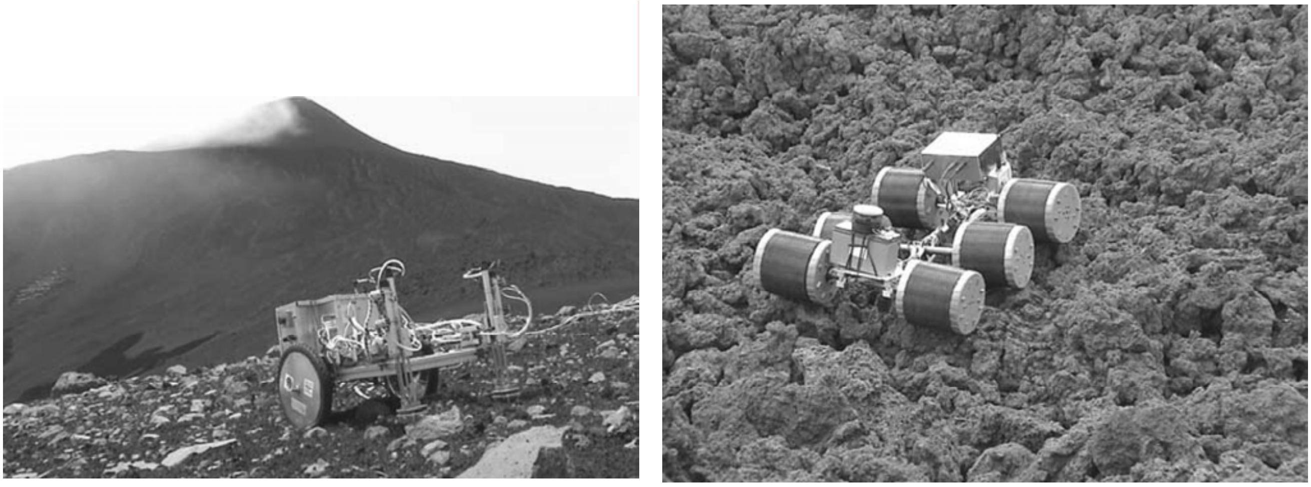


Figure 3. The Wheeleg and the M6

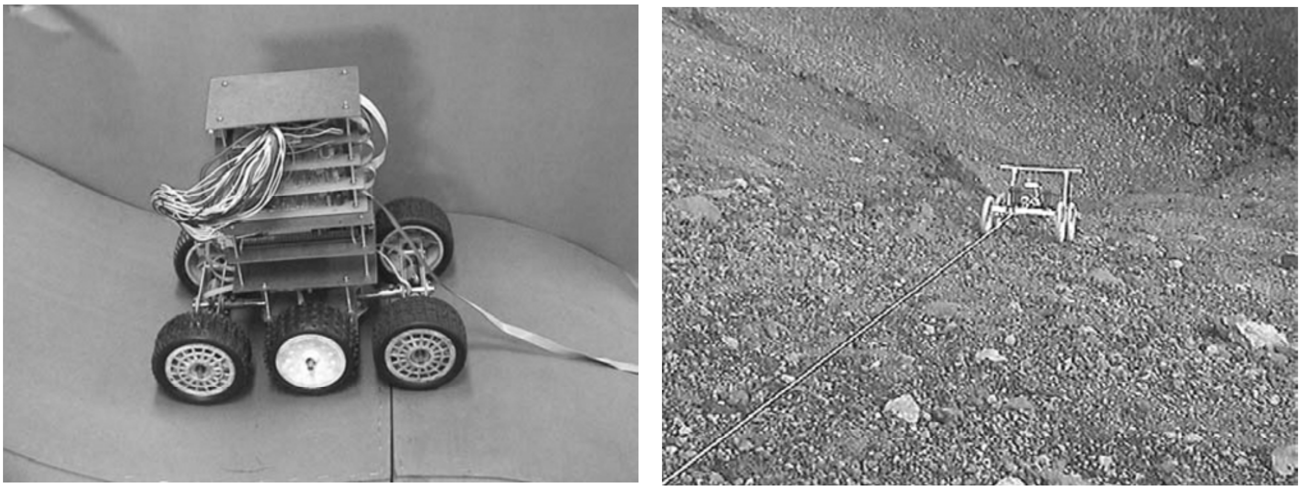


Figure 4. The P6W and the four wheel probe

Following a careful examination of all the measured data and several meetings and discussions performed jointly by the volcanologists and the robotic technology providers, the requirements and the specifications of the Robovolc systems were stated. A typical mission was divided into different phases:

- Transportation of all the equipment from the laboratory into a safe place close to the volcanic vent named base station;
- Settling of the instrumentation on the robot and of the control console;
- Teleoperation of the robot to reach the site of interest;
- Measurements of volcanic parameters;
- Teleoperation of the robot the way back to the base station;
- Transportation of all the equipment into the laboratory.

The specification of the system required a modular robot with each module of limited dimensions and weight, in order to be transported by no more than 5 people in case of problems. The maximum slope was limited to 30° to guarantee a good stability margin and to avoid the use of cables that could limit the range of operations of the system. The distance required to reach the sites of interest from a safe place was estimated to be less than 2km, with half of the path in rough terrain. Once the site is reached the phase of measurement could last several hours, with the robot to be put in a low consumption mode.

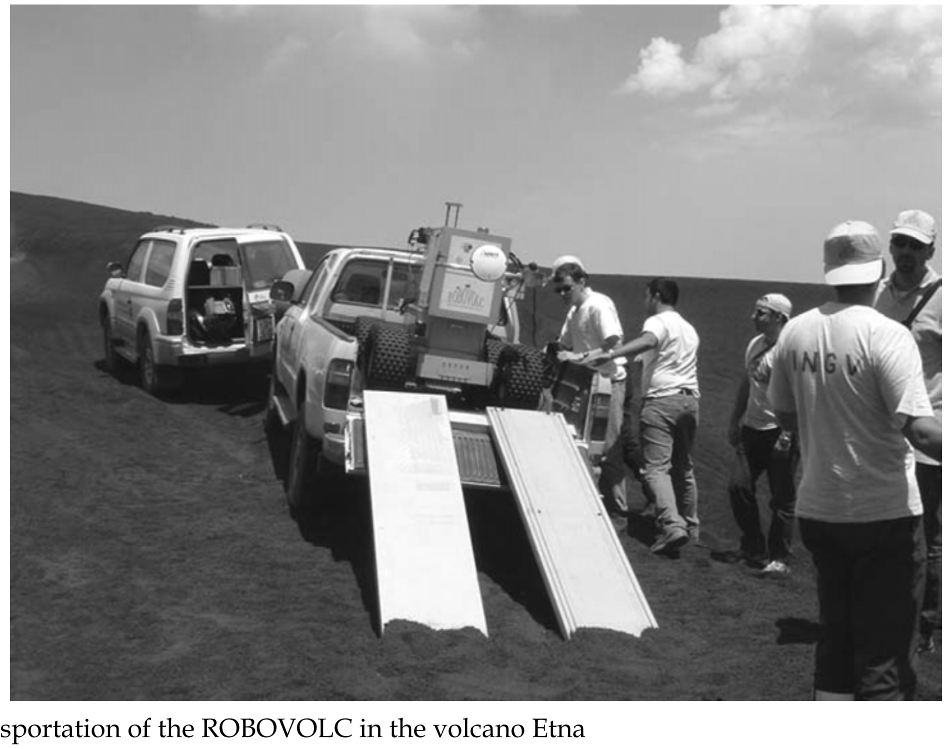


Figure 5. Transportation of the ROBOVOLC in the volcano Etna

4. The ROBOVOLC System

A picture of the final ROBOVOLC system is reported in Fig. 6. The system can be divided in several different subsystems, as shown in the diagram of Fig. 7, that will be described in the following subsections.

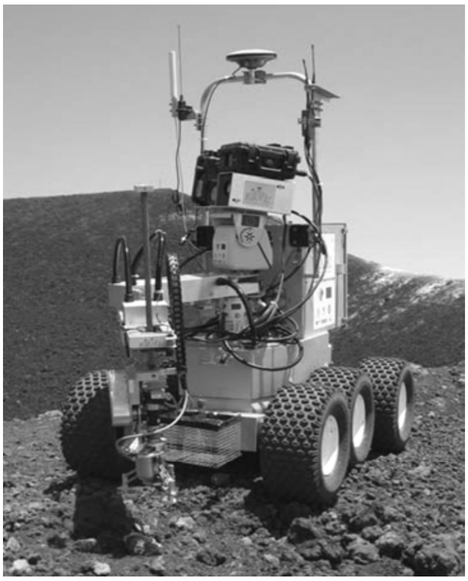


Figure 6. The Robovolc system

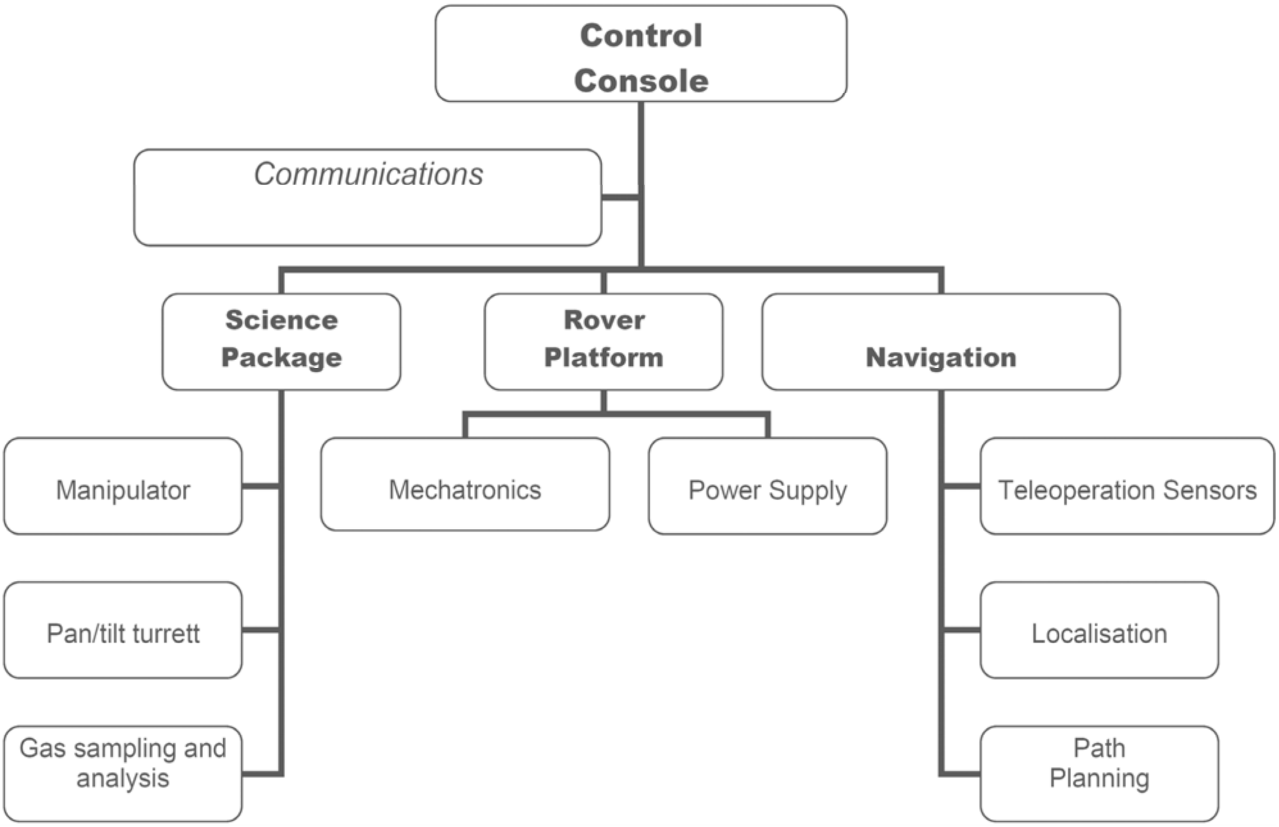


Figure 7. Block diagram of the Robovolc main sub-systems

4.1 The Rover Platform

The platform adopted is a six-wheeled system with an articulated chassis. The system has been equipped with semi-active joints connecting the front and rear axes to the body.
These joints consist of two springs which have controllable stiffness. Tests over volcanic surfaces have revealed that this system outperforms some of the requirements coupled with the quality of mechanical robustness. Six independent DC motors actuate the wheels. Three different types of wheels have been adopted to cope with sandy terrain, rocky

terrain or mixed type terrain. In Fig. 8 some drawing explaining the kinematic capabilities of this platform are reported.

The platform contains the power supply unit (PSU) constituted of 4 sealed lead acid batteries coupled to form two 24V units. The first unit is used to power the platform and the second is used to power the science package and the manipulator. These batteries are mounted in the lower part of the chassis in order to guarantee a low center of gravity and consequently increase the rover stability.

The computer infrastructure is based on the utilisation of three PC104 based computers located in the rover. The first PC is dedicated to the *Motion control* of the rover and to the *Low-speed communication* management; the second PC is for the *Manipulator control* while the third manages the *Science package*, the *Video system control* and the *Localization system*. The first two PC use a Linux operating system, while the third runs on MS Windows 2000 OS. The distribution of the Robovolc systems across three computers was chosen to increase the reliability of the individual components. This is particularly true for rover motion control, with one PC dedicated to motion control; this reduces the likelihood that a failure in one of the other systems can result in the rover being unable to move. The three PC are connected into a LAN and through a high power bridge to a Wireless LAN to be interfaced to the remote Control Console.

The communications between the software modules is implemented through an RPC (Remote Procedure Call) protocol.

The motion controllers used for the control of the DC motors of the platform are RoboSoft RMPC-555. These motor controllers have been used for both Rover and Manipulator motion control and are interfaced to the PCs through respectively a serial port and a CAN BUS (Caltabiano & Muscato,2002),(Caltabiano et al., 2004a).

A traction control algorithm has been implemented to guarantee a suitable distribution of the torque among the wheels, guaranteeing at the same time low power consumption.

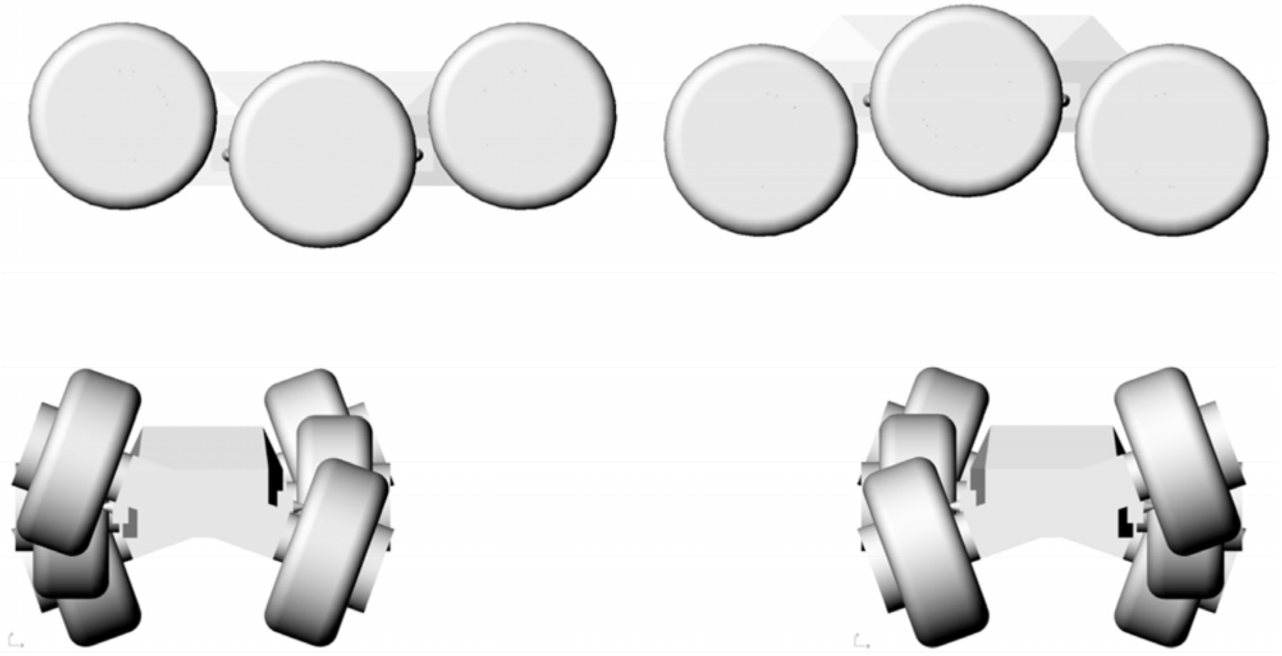


Figure 8. Articulation capabilities of the platform

4.2 The Science Package

The science package is formed by the *pan tilt turret*, the *manipulator* and the *gas sampling system*. All the electronic modules needed to control these equipments are installed within a box mounted on the rover. The pan-tilt turret can be oriented by the user in order to point the different devices to the region of interest (Fig. 9). The devices installed on this turret are a digital video-camera recorder, a high resolution still image camera, a video-camera, an infrared camera for thermal measurement, and a radar Doppler for lava and gas-jet speed measurement. All these devices have been installed within weather and shock proof sealed containers.

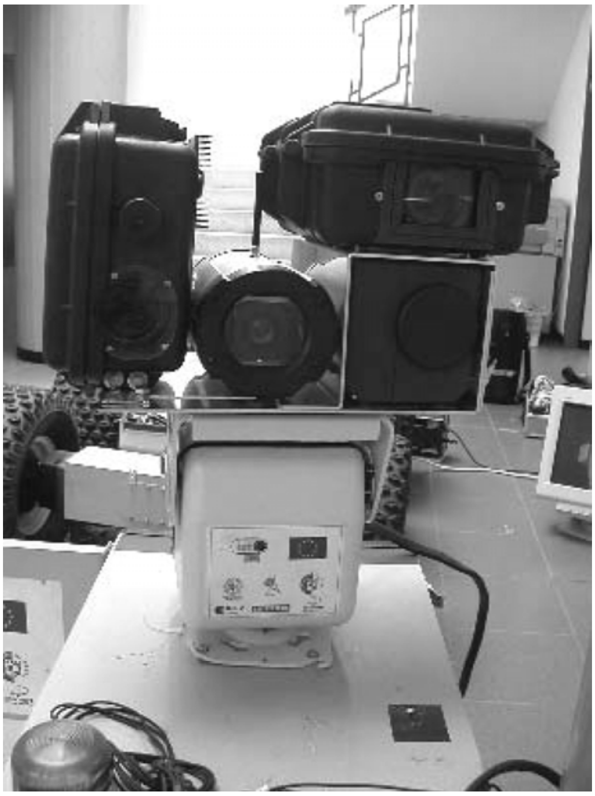


Figure 9. Detail of the pan-tilt turret

The manipulator, shown in operation in Fig. 10, has been designed specifically for the Robovolc system. This is of SCARA type with 5 degrees of freedoms plus a gripper, all actuated by DC motors. The manipulator is adopted to collect samples of rocks, by using a three finger gripper, or to drop and pick instruments into the field or to collect gases in the proximity of fumaroles. This gripper has a force sensor to measure and control the grasping strength.

Gas sampling is the most important measurement to be carried out by the Robovolc system, since the analysis of gas ejected from a volcano is one of the main indicator of its internal activity. The main problems with this operation are: sampling gas that reaches temperatures above 600°C; the presence of extremely corrosive acid components; and the avoidance of gas mixing with the surrounding atmosphere resulting in corrupted samples. A system has been specifically designed for the Robovolc system composed of a titanium probe with a thermal control system, and a gas collection and analysis system. The main problem was to design all the systems to be teleoperated, since most of the equipment was originally designed to be carried on site by human operators. This fact required also the adoption of a series of additional video-cameras to monitor all the sequence of operations.

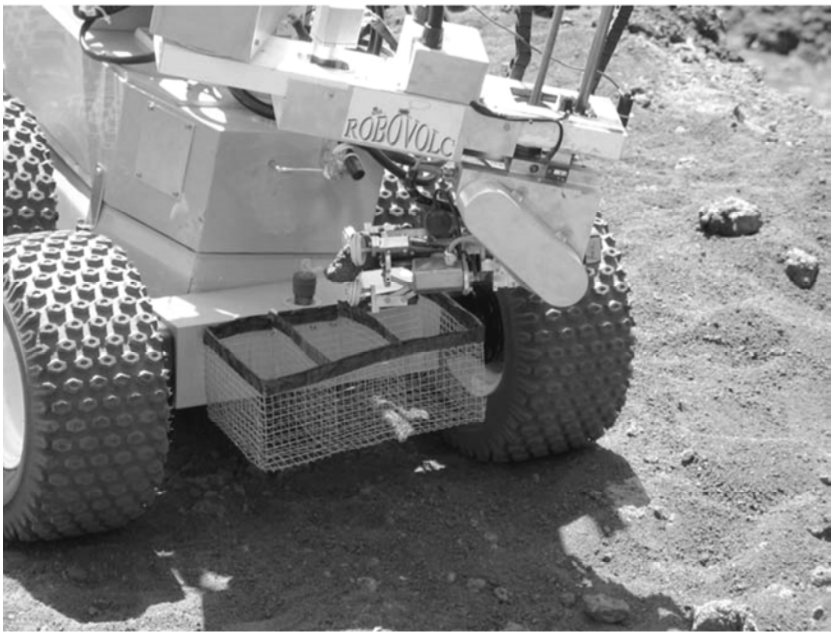


Figure 10. The SCARA manipulator picking a rock

4.3 The Control Console

The Control console had to satisfy two main requirements: to be user friendly, in order to allow the volcanologists to operate the system, and to be rapidly and easily installable on site (Fig. 11). The control console is mainly composed by two LCD monitors and two laptop PCs. The first monitor is connected to a camera on the pan-tilt turret of the science package, while the image of the second monitor can be selected from all the other cameras and the IR-camera. The first PC is to manage the teleoperations of the rover and of the arm, while the second allows operating the science package instruments. In this way a first operator can drive the robot, while a second can, at the same time, conduct the scientific measurements. The control of the robot is accomplished by two joysticks and touch-screen/mouse inputs. Ergonomic considerations have been taken in account in the design of the interface since a mission can be several hours in duration, so the employment of graphics and customisable GUI (graphic user interface) was intended to reduce the eye-strain involved in understanding the various information displayed in the screens (Sim et al., 2004).

4.4 Communication System

The communication system is composed mainly by two subsystems, used for transmitting respectively binary and video data. The first subsystem is based on a redundancy apparatus using both a standard high-power wireless LAN interface, for most of the data exchanged at a high data rate (10Mbps) and a pair of low speed radio modems (100kbps), for critical tele-operation purpose when the wireless communications fails. In fact the wireless LAN interface does not work properly when the robot is out of the line of sight; in this case the radio modems restore the communication with the control console.

The video links are guaranteed by two pairs of video receiver and transmitter, the first pair is used to transmit the video signal of the frontal video camera mounted on the pan/tilt turret while two 8 channels switches are used to connect all the other video signals to the other video link. The video source can be selected by the operator from the control console.

Another communication subsystem is the radio-link that connect the base GPS with the rover GPS for the transmission of the differential signals. Also this link works properly only if the antennas are in the line of sight, when the signal is lost the rover GPS act as a single GPS, hence with a lower accuracy.

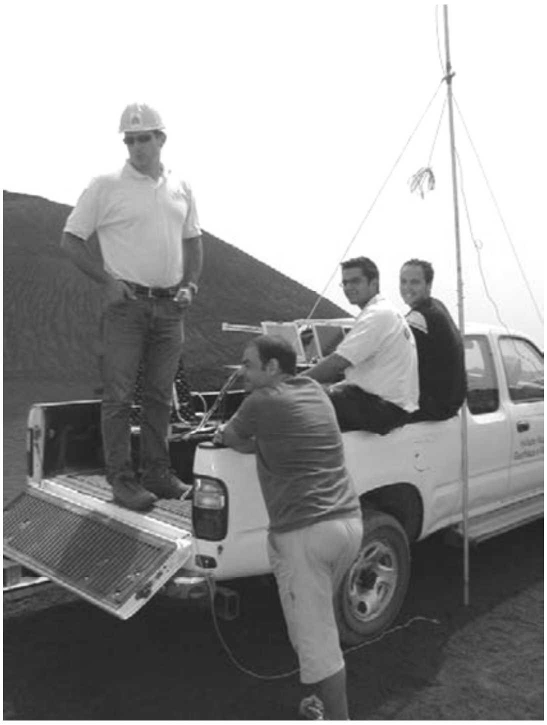


Figure 11. The Control Console installed on a pick-up vehicle

4.5 Navigation System

Most of the operations of Robovolc are handled by a human operator, so that the system is tele-operated, however since the connection link cannot be guaranteed for all the situations and in order to increase system reliability and to decrease the effort of the operators in long duration mission, a limited amount of autonomy was included in the system. For teleoperations five fixed and two mobile video cameras are adopted. Four of the fixed cameras view each side of the robot, while the fifth is in front of the robot to help the teleoperations of the arm. Another camera is mounted on the manipulator wrist and is used to help pick and place operations, while a last camera with zoom capabilities is on the pan-tilt turret and is adopted to examine far and close terrains in front of the robot. The user if needed can switch the monitor to the IR-camera view, thus allowing to avoid the robot to move on very hot surfaces.

As regard autonomous navigation this is based on a hierarchical structure that models the cognitive process employed in human navigation. This is implemented into four layers: *long range planning* to define the map based way-point map of the robot; the *short range path planning* that manages the navigation of the robot between the waypoints; *the instant path planning* that decides the direction of the robot on the basis of the short range layer inputs and on the terrain data. Finally the motion control layer translates these commands into control commands for the motion control boards.

In order to guarantee a precise positioning of the robot a reliable localisation system was needed. Some details of this localisation system are described in the next section.

5. Localisation System

Localization is a major task in mobile robotic for outdoor environment. The function of the *localisation system* is to determine the exact location of the robot both for navigation purposes and also to allow the volcanologists to reconstruct the terrain morphology. A block diagram is reported in Fig. 12. The main sensor adopted for localisation is a DGPS (Ashtec model Z-Xtreme) based on a Real Time Kinematic Differential GPS (RTKDGPS) with an optimum position accuracy lower than one centimetre. However the precision of the DGPS in some situation can be drastically reduced. This can be caused by the loss of communication with the differential correction signal or to the interference of satellite signals reflected by nearby rocks. In these cases the accuracy of the DGPS can drop to several meters.

In order to avoid such problems several data fusion algorithms have been designed to improve the accuracy of localisation data from the DGPS by merging it with inputs from other sensors. The sensors inputs include odometry data from the wheels encoders, rate of turn from gyroscopes and the attitude of the robot measured by an inclinometer (Longo et al., 2002).

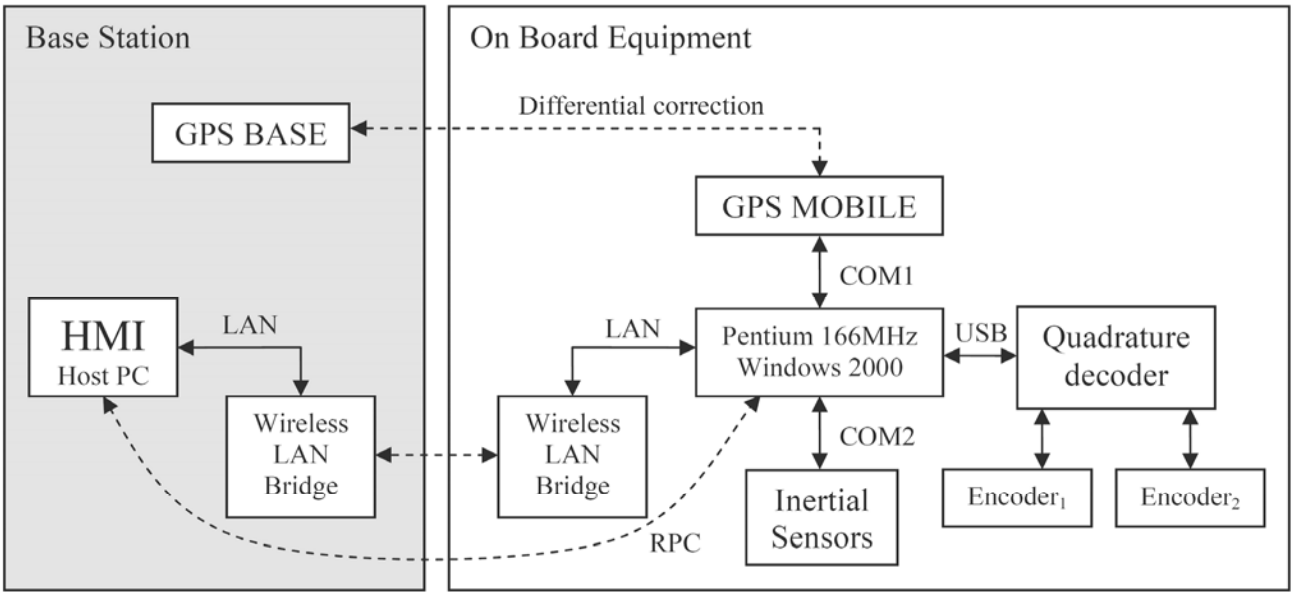


Figure 12. Block diagram of the Localisation system

5.1 Self Calibrating Extended Kalman Filter

This section presents one of the localization algorithms, called "Self Calibrating Extended Kalman Filter" (EKFSC), which has been developed for Robovolc. The EKFSC algorithm uses an EKF to fuse DGPS and encoder data in order to calculate the robot position and to estimate the parameters of the robot model (Caltabiano et al., 2004b).

Odometry is based on the assumption that measures of wheels movements can be translated into measures of the vehicle motion using the kinematic model of the robot. This assumption is only of limited value, because there are several reasons bringing inaccuracies in the odometry. The main causes of error sources in odometry are: wheels slipping, uncertainty in the odometry parameters (wheels radii and wheelbase), misalignment of the wheels and finite encoder resolution and sampling frequency.

The error sources related with the encoders, most of the time, can be neglected due to the excellent resolution of the encoders and to the high sampling frequency. If the robot is well assembled, the misalignment of the wheels is also negligible compared to the other sources of error. The errors caused by the wheels slipping cannot be estimated by the encoders and hence can not be compensated. On the other hand the error caused by the uncertainty on the odometry parameters could be reduced if a good calibration is performed (Borenstein and Feng, 1996).

Calibration is the problem of estimating the parameters of the robot model from sensors measures. This estimation is sometimes very difficult for various reasons: the pressure of tires can change during the time; furthermore the effective wheelbase depends on the terrain where the robot is moving; In our specific application, different kind of tires can be mounted on Robovolc, depending on the environment where the robot has to move (rocky or sandy terrain), hence affecting the measures of the wheels diameter and wheelbase; therefore the calibration of the robot parameters is periodically required.

In practical robotic applications, the odometry parameters are identified in a dedicated phase and then they are no more calibrated. Most of the researches, in fact, use an absolute sensor like a Laser Scanner or a GPS to compensate the accumulated errors caused by dead reckoning (Martinelli et al., 2003), (Foxlin, 2002). Unfortunately, using wrong parameters, the robot can get lost very quickly, if for some reason the measure from an absolute sensor is not available. For example a big rock or a building can reduce the number of visible satellites of a GPS.

The EKFSC is a solution to the Simultaneous Localization and Auto-Calibration (SLAC) problem using an Augmented Kalman Filter (AKF): the state vector has been augmented with the odometry parameters; moreover an odometric filter model is used in the *Predict phase* of the EKF, while the *Update phase* uses the GPS data asynchronously.

The EKFSC localization algorithm relies on an Extended Kalman Filter where the predictive phase, calculated using the odometry measures, computes at high frequency the expected position and orientation of the robot and the update phase corrects these estimations at a lower frequency using the GPS measures. The odometry parameters are considered as state variables in the EKF hence they are estimated together with the localization variables. The details of this algorithms are reported in the paper (Caltabiano et al., 2004b), where a comparison with the UMBmark calibration procedure is also reported (Borenstein and Feng, 1996).

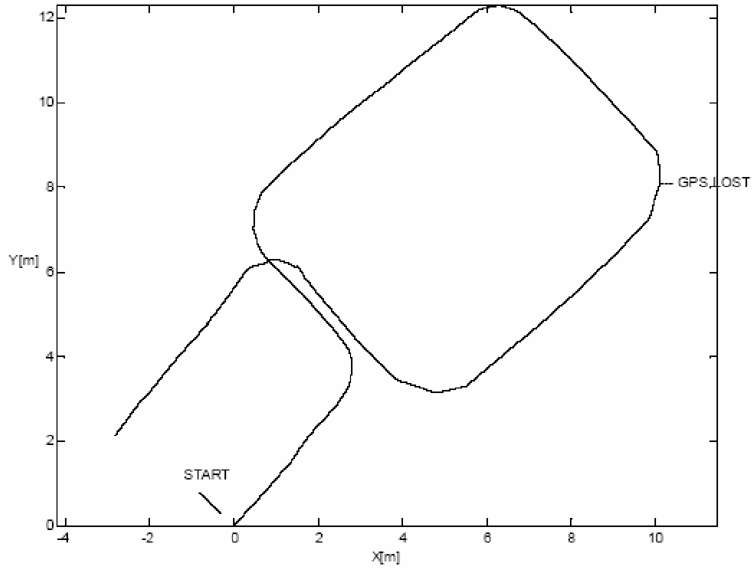


Figure 13. Robot trajectory adopted for testing the EKFSC localization algorithm

In order to test the presented algorithm several trials have been realized with Robovolc. The odometry parameters of Robovolc, measured directly on the robot are: Wheels radius: *R1*=0.21m, *R2*=0.21m, Wheelbase: *L*=0.82m, while the EKF algorithms are initialized with the wrong parameters: *R1(0)*=0.20m, *R2(0)*=0.20m, *L(0)*=0.69m. The robot moves along the trajectory shown in Fig. 13. During this trial, DGPS data have been really accurate, so the above trajectory has been reconstructed using only this sensor. A GPS failure has been simulated at time t=90s and is indicated in the image as *GPS LOST*, hence in the rest of the trajectory the GPS measures are not available for the localization algorithm.

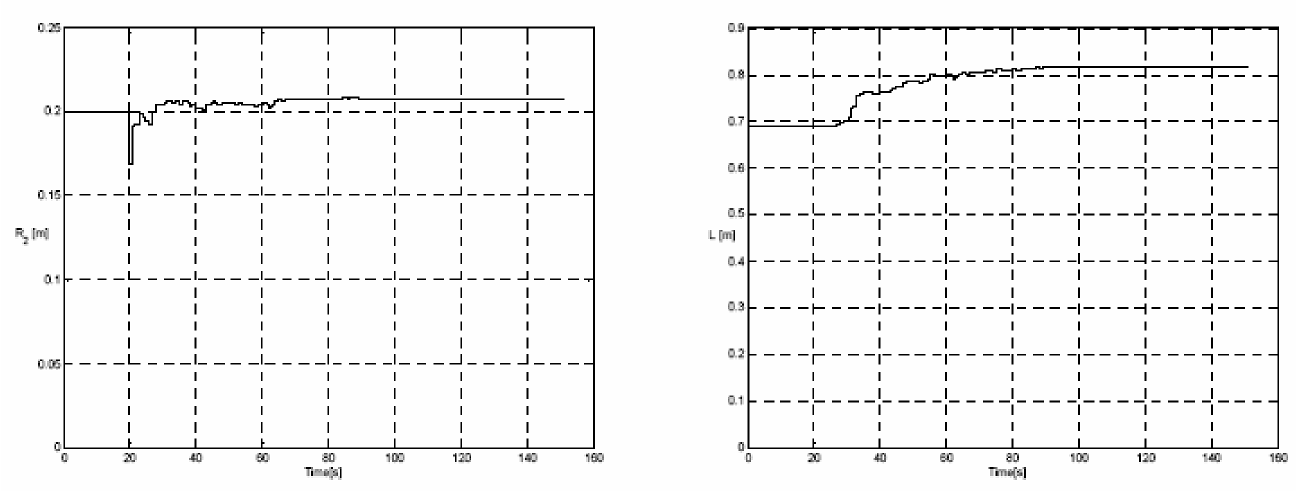


Figure 14. Estimation of *R1* and of *L*

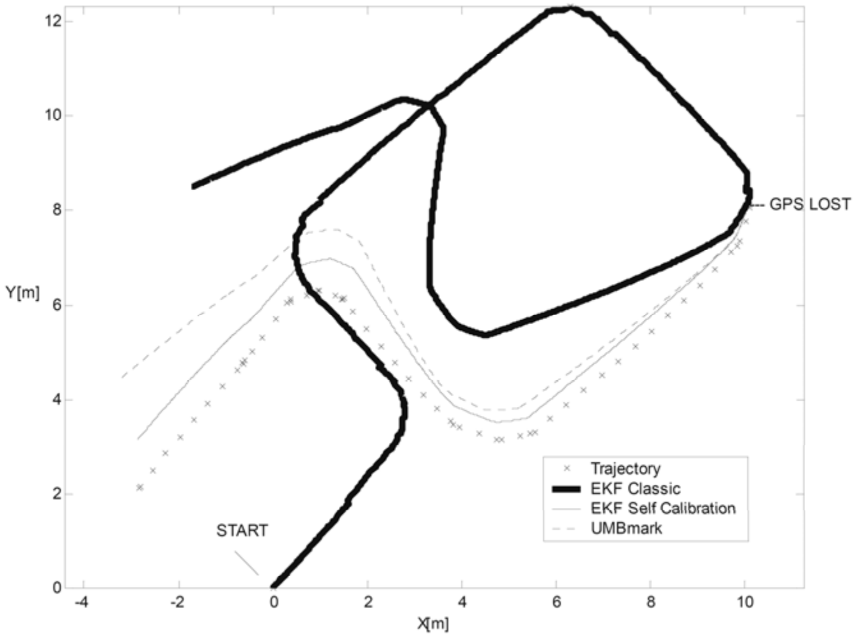


Figure 15. Comparison between EKFSC and EKFClassic and an EKF calibrated via the UMBmark procedure

When the robot was moving along the trajectory, the odometry parameters have been estimated as shown in Fig. 8 and 9. Fig. 14 shows the estimation of *R1* and of *L*. The estimated final value are *R1(90)*=0.209m and *L(90)*=0.816m. Similar results have been obtained for *R2*, the estimated final value is *R2(90)*=0.208m.

Fig. 15 shows the comparison between the EKFSC the EKFClassic and an EKF calibrated via the UMBmark procedure in the reconstruction of the test trajectory

The EKFClassic brings to really poor results when the GPS is not available and the parameters are not calibrated. On the contrary the trajectory reconstructed, using the EKFSC algorithm, is very close to the real one and is much better than those obtained by using the UMBmark calibration.

5.2 Orientation Estimator

Another localisation algorithm that was implemented and tested on the Robovolc platform is a modified Kalman filter that includes an orientation estimator. Magnetic compasses are the most classical absolute orientation sensors. However, the precision that can be obtained is usually low (0.1°) and they can be strongly affected by external disturbances. When an extra magnetic field is present (i.e. motors, electrical equipment or magnetic rocks), the precision of the compass can be very poor. The rocks in volcano environments are often magnetic hence compass provides very unreliable measures. Moreover, no estimation of the standard deviation of the measurements is usually given by the sensor. For all these reasons using a compass on volcano environments is not recommended.

This problem is particularly important when the robot is moving following a rotation. In this case due to the skid-steering system, the precision of the odometry to compute the orientation is very low and a classical EKF converge to the true orientation very slowly. The absolute orientation can be obtained, instead, helping the EKF by using two consecutive GPS measures. The adopted localization algorithm is a classical Extended Kalman Filter where the predictive phase, calculated using the odometry measures, computes at high frequency the expected position and orientation of the robot and the update phase corrects these estimations at a lower frequency using the GPS measures (Fig. 16). Moreover the measurement vector contains both the position and the orientation of the robot; the latter is obtained through an algorithm using the information contained in two consecutive GPS measures (Caltabiano & Muscato, 2003).

In order to verify the speed-up of the EKF3 algorithm, it has been compared with the classical EKF2 algorithm.

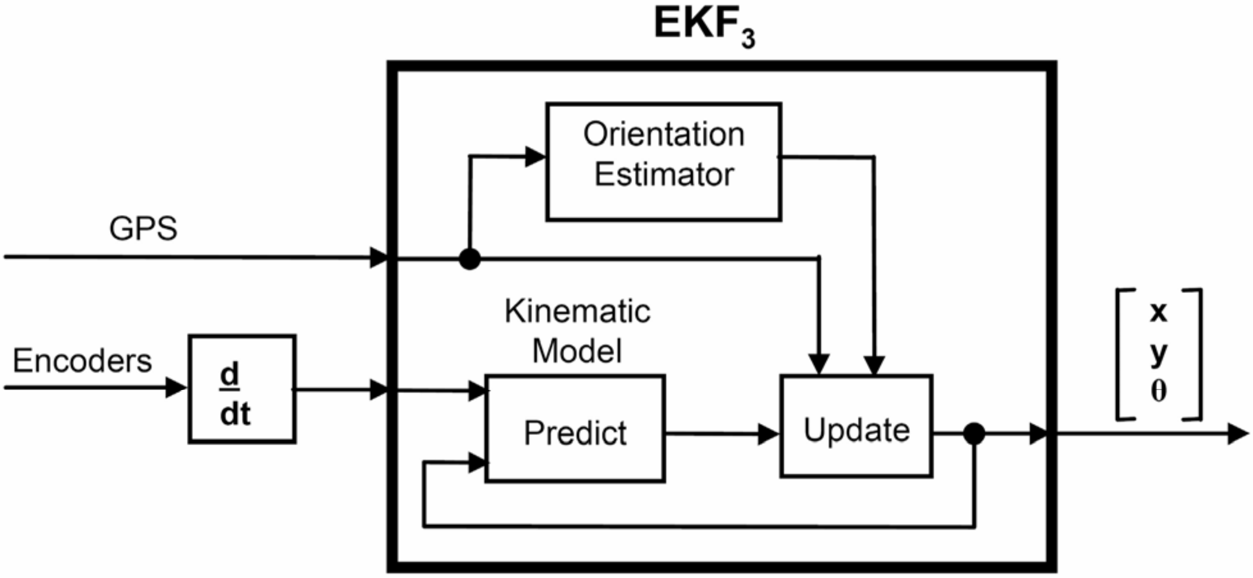


Figure 16. Scheme of the EKF3 localisation algorithm

It has been assumed that: the robot moves toward North-West with an average speed of 0.81 m/s; the GPS has a sampling frequency of 1 Hz; the Encoders are sampled at 20 Hz; the initial orientation of the EKF algorithm is the East. The EKF2 algorithm output is given in Fig. 17a, where the square boxes correspond to the GPS measures (sampled at 1Hz) connected with a solid line, the small points are the EKF output (sampled at 20Hz) while its standard deviation is represented with circles.

In order to obtain clear images, the standard deviation circles are plotted at half rate (10Hz) and for diameters below half a meter. Fig. 17b refers to the EKF3 algorithm. As it can be observed from these results the transient passed from 5 seconds (4 m of path) obtained with the classical EKF2 algorithm to 1 second (70 cm) obtained with the EKF3. In our application 4m of path with an uncertain direction could represent the difference between falling inside a crater and following the right trajectory.

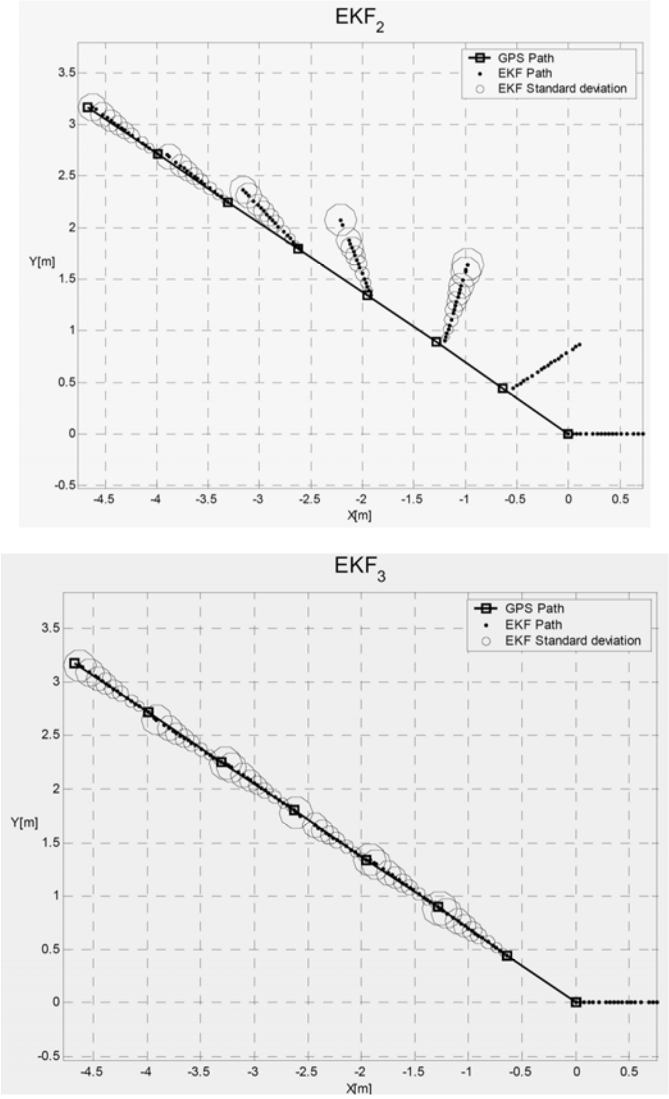


Figure 17. Transient of the EKF2 and EKF3 algorithms of the Robovolc tests

6. Trials on Mt. Etna Volcano and Final Considerations

Several tests of the Robovolc platform have been performed on the Mt. Etna Volcano in Italy where different kinds of terrain can be explored due to different types of eruptions over the years. Some pictures of the platform and of the explored environment are reported in this section. Each of the Robovolc subsystems has been tested initially separated and then altogether in several virtual missions.

A preliminary test campaign on Etna volcano has been performed in September 2002 and has been discussed accurately in (Muscato et al., 2003).

In summer 2003, the rover has been successfully tele-operated to move inside one of the still hot craters of the eruption Dec 2002-Jan 2003. Traction tests have been performed successfully on rocky terrain (Fig. 18).

The rover demonstrated very good traction capabilities on rough surfaces. The robot easily coped with rocky obstacles of more than 40 cm in diameter and with 30 cm ground fissures. Traction tests with different tyres type on sandy environment have been performed successfully too, on the base of the *Laghetto Crater* (January 2002 eruption) (Fig. 19) also in surface with slopes of 30°.

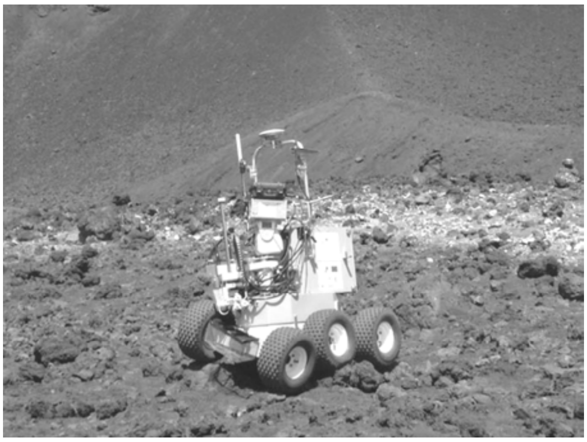


Figure 18. Operations on rocky terrain

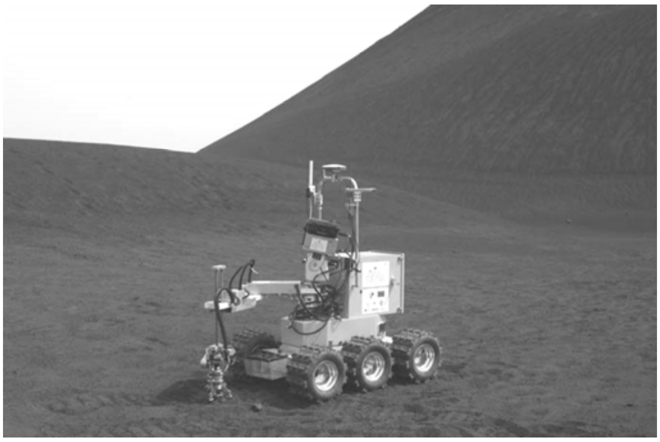


Figure 19. Operations on sandy terrain

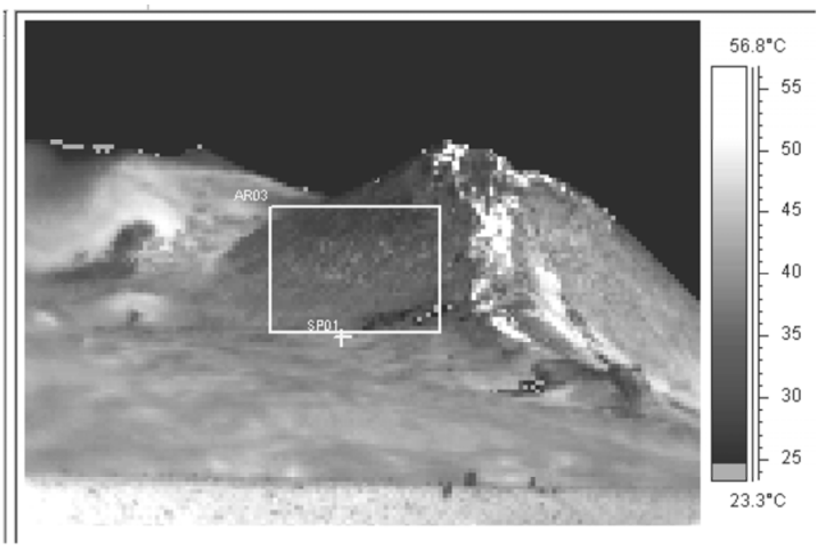


Figure 20. The Etna SE Crater viewed from the IR camera on board the robot. Clearer zones indicates higher temperatures

During the tele-operation of the robot, the entire science package has been tested: the pan-tilt turret has been oriented properly to take some pictures with the IR camera. One of the images of S.E. crater taken by the robot using the IR camera is shown in Fig. 20. Several videos have been recorded with the onboard video-camera and some navigation tests have

been performed to validate also the localization system. The manipulator has been tested in order to pick sample of rocks found in the environment.
Each trial required a great logistic effort in order to carry the robot on an altitude of about 3000m, to set-up the base station and to manage the uncountable problems encountered in all the testing phase. Fig. 21 shows the participants of this trial near the Robovolc platform.

Figure 21. Some participants to one Robovolc test campaign

7. Conclusion

The main innovation of this project is the capability to take measurements during volcanic eruptions by the development of a specific mobile robotic system for this purpose.
This robot will have a relevant impact on the mitigation of the volcanic risk both in general, since it contributes to improve knowledge about volcanic phenomena, and in particular because it is now integrated in the volcanic surveillance system to be used when the approach to active vents becomes too dangerous for human live, but information is vital for a correct forecast of dangerous eruptions. For instance, during the volcano unrest that precede a large eruption when the gas emission, thermal variation and ground deformation inside a crater or caldera, related to the ascent of new magma, give a direct information for forecasting of the approaching eruptive event; or during long-lived volcanic eruptions like large dome inflation the evaluation of the lava dome stability will significantly improve the forecasting of dome failures and pyroclastic-flow forming eruptions. The robot activity will contribute to an integrated risk management system by means of the updating, using GIS, of areas potentially threatened by catastrophic eruptions with geographical information about the volcanic products dispersion and volcanic hazard assessments to obtain an almost real-time early warning and to inform Civil Protection authorities about dangerous volcanic eruptions approaching.
The missions demonstrated the usefulness of a mobile robot in contributing to science research for volcanology. the system is now a new tool own by Istituto Nazionale di Geofisica e Vulcanologia in Sicily who are responsible for surveillance of Mt. Etna and the other Sicilian volcanoes in the Aeolian islands. The Robovolc system is ready to be used in the next active volcanic phases.

The expertise acquired in the development of the project and during the trial allowed the group to greatly improve the system and its reliability. On field trials have been fundamental to understand the real requirement of a system that has to operate in such hard environment and that could not be discovered by simple laboratory test.

The authors acknowledge the work performed by all the partners of the Robovolc project, more than 50 different persons were involved in the various steps of the project and their effort was fundamental for the success of the operations. Particular thank goes to each group leader M.Coltelli (INGV), P. Briole (IPGP), T. White (BAESYSTEMS), A. Semerano (Robosoft) and Prof. G.S. Virk (University of Leeds).

8. References

Azad, A.K.M.; Virk, G.S.; Qi, M; Muscato, G.; Guccione, S.; Nunnari, G.; White, T.; Glazebrook, C.; Semerano, A.; Ghrissi, M.; Briole, P. ; Faucher, C.; Coltelli, M.; Puglisi, G.; Cioni, R. & Pompilio, M. (2001). ROBOVOLC: Remote Inspection for Volcanoes, *Workshop on Autonomous artificial systems exploring hostile environments, International NAISO Congress on Information Science Innovations (ISI'2001)*, Dubai, March 18th 2001.

Bares, J.E. & Wettergreen, D.S. (1999). Dante II: Technical Description, Results and Lessons Learned, *The International Journal of Robotics Research*, pp. 621-649, July 1999.

Borenstein, J. & Feng, L. (1996). Measurement and correction of systematic odometry errors in mobile robots, *IEEE Transactions on Robotics and Automation*, Vol. 12, Issue: 6 , Dec. 1996 pp.: 869 -880.

Caltabiano, D & Muscato, G. (2002). A Comparison between different traction methods for a field robot, *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2002*, Lausanne Switzerland, Sep.30-Oct.4 2002.

Caltabiano, D & Muscato, G. (2003). A New Localization Algorithm for Mobile Robots in Outdoor Environments, *Proceedings of the 6h International Conference on Climbing and Walking Robots (CLAWAR 2003)*, 17-19 September 2003 Catania, Italy, pp 925-932.

Caltabiano, D.; Ciancitto, D. & Muscato, G. (2004a). Experimental Results on a Traction Control Algorithm for Mobile Robots in Volcano Environment, *Proceedings of the 2004 IEEE International Conference on Robotics & Automation*, New Orleans, LA, April 2004, pp. 4375-4380.

Caltabiano, D.; Muscato, G. & Russo, F. (2004b). Localization and Self Calibration of a Robot for Volcano Exploration, *Proceedings of the 2004 IEEE International Conference on Robotics & Automation*, New Orleans, LA, April 2004, pp. 586- 591.

Foxlin, E.M. (2002). Generalized Architecture for Simultaneous Localization, Auto-Calibration, and Map-building, *Proceedings of the 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems*, EPFL, Switzerland, October 2002.

Guccione, S; Muscato, G.; Nunnari, G.; Virk, G.S.; Azad, A.K.M.; Semerano, A.; Ghrissi, M.; White, T.; Glazebrook, C. (2000). Robots for Volcanos: The State of the Art, *Proceedings of the 3rd International Conference on Climbing and Walking Robots (CLAWAR 2000)*, pp.777-788, Madrid (Spain), 2-4 October 2000.

Guccione, S. & Muscato, G. (2003), Control strategies computing architectures and experimental results of the hybrid robot Wheeleg, *IEEE Robotics and Automation Magazine*, Vol.10 No. 2, 2003. (IEEE Piscataway, USA).

Kemurdjian, A.; Gromov, V.; Mishkinyuk, V.; Kucherenko, V. & Sologub, P. (1992). Small Marsokhod Configuration, *Proceedings of the IEEE International Conference on Robotics and Automation*, Nice, France 1992.

Lacagnina, M.; Muscato, G.; Guccione, S. & Sinatra, R. (2002). Modelling and simulation of multibody mobile robot for volcanic environment explorations, *IEEE/RSJ International Conference on Intelligent Robots and System, IROS 2002*, Lausanne Switzerland, Sep.30-Oct.4 2002

Lacagnina, M.; Muscato, G. & Sinatra, R. (2003). Kinematics, Dynamics and Control of a Hybrid Robot Wheeleg, *Journal Robotics and Autonomous Systems*, (Elsevier, Oxford, UK),Vol 45, pp. 161-180, 2003.

Longo, D; Muscato, G. & Sacco, V. (2002). Localization using fuzzy and Kalman filtering data fusion, *Proceedings of the 5th International Conference on Climbing and Walking Robots (CLAWAR 2002)*, 25-27 September 2002 Paris, France.

Martinelli, A.; Tomatis, N.; Tapus, A. & Siegwart, R. (2003). Simultaneous Localization and Odometry Calibration for Mobile Robot, *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Las Vegas, Nevada, October 2003.

Muscato G.; Caltabiano, D.; Guccione, S.; Longo, D.; Coltelli, M.; Pecora, E.; Cristaldi, A.; Virk, G.S. ; Sim, P.; Sacco, V.; Briole, P.; Semerano, A. & White, T. (2003) ROBOVOLC: A Robot for Volcano Exploration Result of first test campaign, *Industrial Robots*, Vol. 30, N.3, May 2003, pp.231-242, ISSN 0143- 991X.

Sim, P.; Sacco, V. & Virk, G.S. (2004). The user interface for the Robovolc exploration robot, *Industrial Robots*, Vol. 31, N.2, pp.189-200, ISSN 0143-991X.

YAMAHA (2000) Helicopter news: http://www.yamaha-motor.co.jp/news/2000-04-24/sky-e.html

A Simulator for Helping in Design of a New Active Catheter Dedicated to Coloscopy

Georges Dumont & Christofer Kuehl

1. Introduction

In this paper*, we focus on a dynamical simulator dedicated to the global design, by means of virtual prototyping methods, of an active micro-catheter for coloscopy. These devices are fully in the scope of surgeons demand for near future, because they enable the operating gesture to be assisted (Cohn, M., Crawford, L., Wendlandt, J. & Sastry, S. S. 1995) and thus, minimise one patient hurt during intervention. Prototypes of such systems are proposed for more than ten years (Lim, G., Minami, K., Yamamoto, K., Sugihara, M., Uchiyama, M. & Esashi, M. 1996), but only few works are dedicated to the computer aided design process of these devices (Dumont, G., Chapelle, F. & Bidaud, P. 2001) (Ikuta, K., Iritani, K. & Fukuyama, J. 2001). Classic flexible endoscopes are generally constituted of articulated links, actuated by four cables connected to the head of the endoscope. The motion is achieved by means of knurls that are pulled by the surgeon, so the gesture is not assisted. A new generation of active endoscopes was developed by Olympus (Takehana, S., Ueda, Y., Gotanda, M., Sakurai, T. & Adachi, H. 1990). In this device, the head orientation is controlled through flexion modules activated by SMA actuators. However, this system does not automatically adapt its curvature to the environments. Developments of new prototypes, aiming at improving the navigation of endoscopes in the inspected ducts, are in progress. Prototypes constituted of trays interconnected by SMA threads have been proposed in which the piloting the length of the threads controls the orientation between two consecutive trays. One, for example, has been proposed in (Montesi, M.C., Martini, B., Pellegrinetti, A., Dario, P., Lencioni, L., Montano, A. 1995). Its diameter is 8mm and only the head is controlled. Another has been proposed in (Park, K. & Esashi, M. 1999). This prototype is of diameter 1.7mm and is dedicated to interventions in ureter. It is presented in Fig. 1.

We propose here a brief description of our real prototype and outline the interest of designing a training simulator. As the foreseen catheter will have to crawl its way inside complex environments as human ducts, we have chosen a modular stiff poly-articulated structure presented in Fig. 2 and designed in our laboratories (Szewczyk, Sars, Bidaud & Dumont, 2000) and (Kühl, Dumont, Mognol, Gouleau & Furet, 2002).

The chosen mechanism consists in a succession of modules related to each other by pin joints, which are alternatively oriented at 90°. This configuration allows a 3D motion of the structure. The endoscope head should contain a device allowing obtaining a multi-

* This work was partially supported by the Centre National de la Recherche Scientifique, France (CNRS/SPI).

directional vision of the observed space. This will be achieved by using a polymer gel actuated prism.

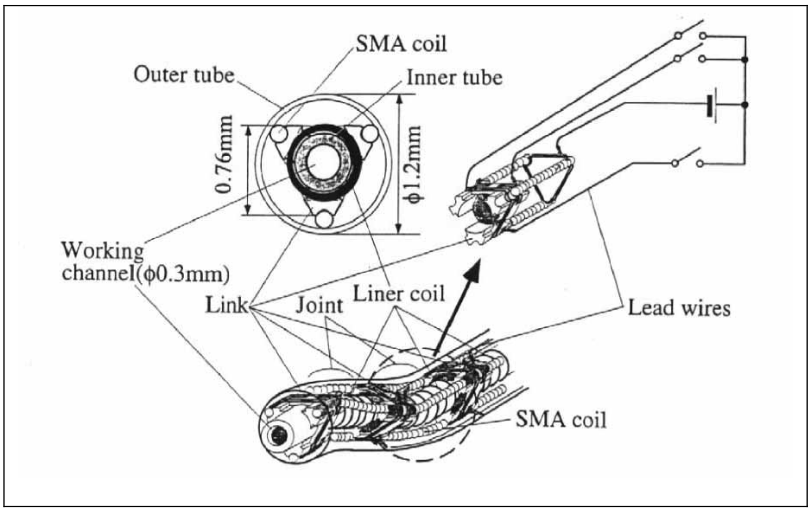


Figure 1. Multilink active catheter (Park & Esashi, 1999)

As medical device, a flexible polymer sheath should protect the whole system. Each joint is provided with two antagonist electrically commanded SMA spring-like actuators allowing the control of the relative orientation of consecutive links. An integrated circuit controlling the electrical power supplied to the SMA realizes the command. The SMA actuators have been chosen because they ensure a good compromise between their mass and the forces that they can deliver.

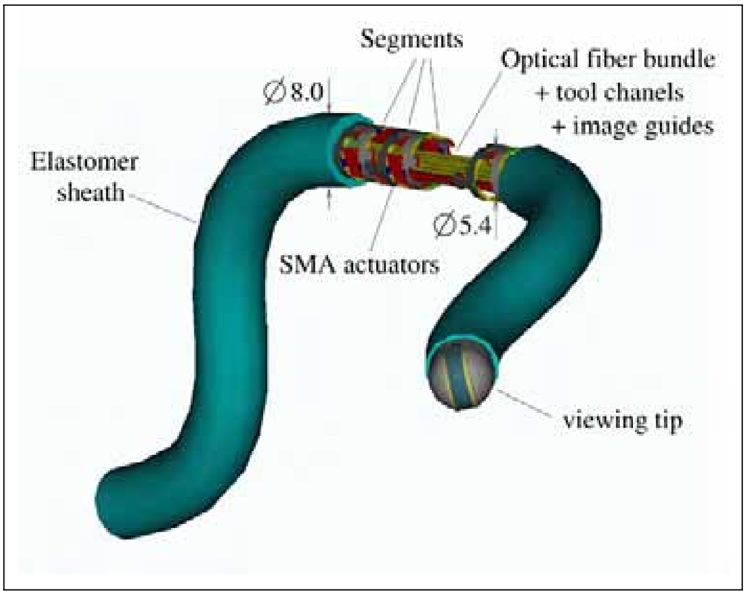


Figure 2. Schematic view of the prototype

Fig. 3 presents our simulator. The objective for this simulator is to realize virtual operations in a virtual patient body, modelled by a reconstruction of the digestive tract. In our case, the model of this digestive track is obtained by Magnetic Resonance Imaging (MRI). The surgeon feelings will be maximal by using graphic and haptic interfaces provided by the virtual reality techniques. The computational model used for the simulation is the model of the proposed real prototype.

The interests for a design and training simulator are triple:

- Teaching to young surgeons: the training on simulator is obviously more accessible, less expensive and less risky and should allow to experiment pathologic cases;
- Preoperative training: the simulator will allow the surgeon to virtually repeat the operation on the patient, before the real operation and thus minimize the risks;
- Virtual prototyping (Dumont, G., Kühl, C. & Bidaud, P. 2002): furthermore, the simulator allows to virtually test the tool quality with respect to a given operation task. By using optimization techniques based on genetic algorithms, we can test which is the best candidate to achieve this task, and propose an adapted endoscope.

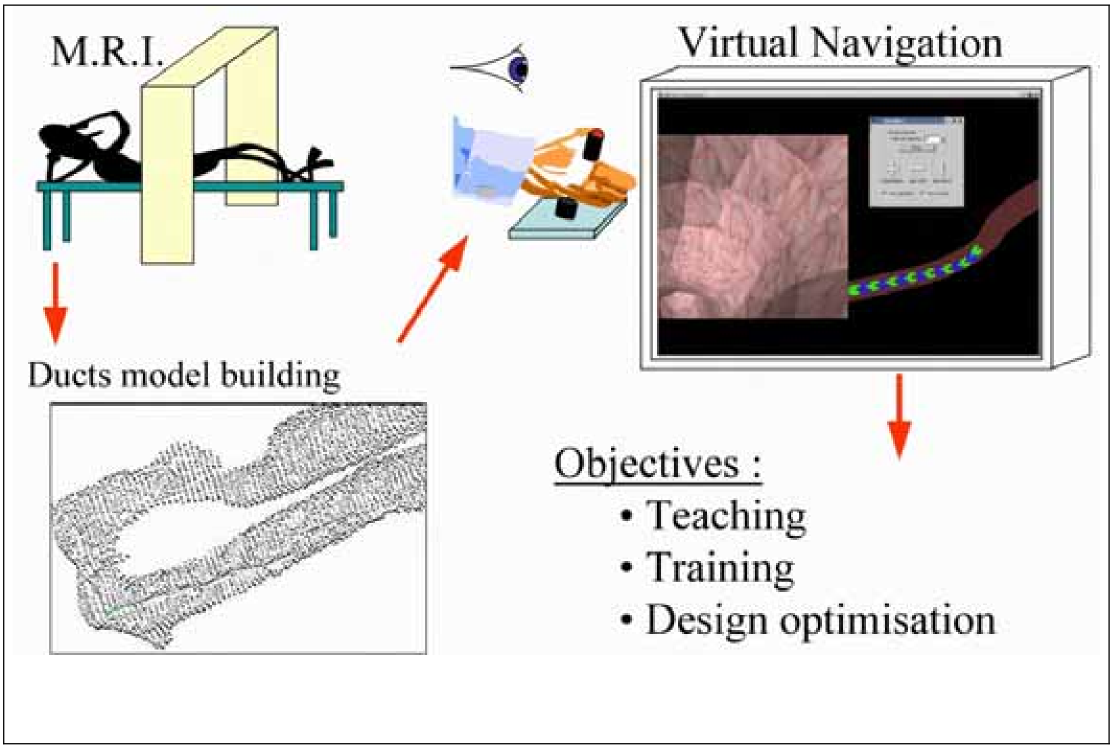


Figure 3. Proposed Simulator

In the following, we will focus on the simulator components. In order to deal with endoscope toward ducts interaction, we present the "devoxelisation" process, which leads to an interpolated distance cartography of the ducts. This database construction is used to minimize the computation time for treatment of the interactions. The contact detection algorithm, allowing determining these interactions forces between the endoscope model and the duct model is then presented. Let us notice that an important experimental work should be lead to identify the parameters of this contact model. A result will illustrate the simulation process. To minimize one patient hurt during operation, we use the control capability of the device. We will address it by a multi-agent control strategy for controlling the catheter conformation to the ducts and compare this method to the uncontrolled case. It is shown that the magnitude of contact forces decreases.

The third proposed use of the simulator is the design optimization of the prototype. To address this purpose, the simulator is coupled to genetic based algorithms. This allows showing the feasibility of an optimization process for geometric conditions, where the design is optimized with respect to accessibility conditions. We propose also an optimization process for dynamic conditions, where a dynamical simulation of the system evaluates the design during the realization of an inspection task in the environment.

2. Description of the Simulator

A mechanical model of the proposed real prototype allows us to better understand its capabilities. The model is a kinematical open chain described by a Denavit and Hartenberg representation presented in Fig. 4.

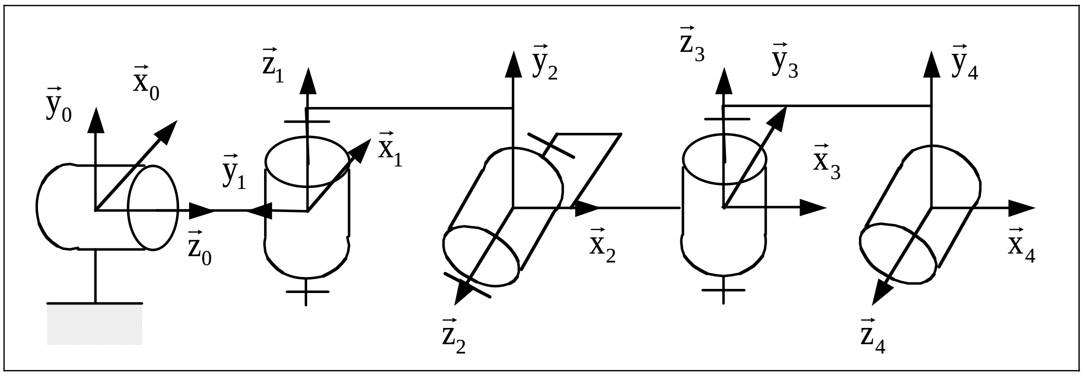


Figure 4. Denavit-Hartenberg representation of the endoscope

Let $(q_i) = (z_1, \theta_1, \theta_2, ..., \theta_n)$ denote the set of parameters describing the endoscope configuration. The Lagrange formulae allows to write the motion equation for such a model:

$$\frac{d}{dt}\frac{\partial K_E}{\partial \dot{q}_i}(q,\dot{q},t) - \frac{\partial K_E}{\partial q_i}(q,\dot{q},t) = Q_i - \frac{\partial E_p}{\partial q_i} - \frac{\partial E_d}{\partial \dot{q}_i} \tag{1}$$

In this equation, K_E is the kinetic energy of the system. As in our application, the movements are slow and as the mass of the device is small, we neglect this inertia effect in the model.

In this equation, Q_i are the external forces due to contact interactions and E_p denotes the potential energy due to gravity E_{grav} and elasticity $E_{pe} = \frac{1}{2}K_s(\theta_i - \theta_{i0})^2$ of the polymer sheath. The dissipative term at each pin joint is written as $\frac{\partial E_d}{\partial \dot{\theta}_i} = A\dot{\theta}_i$.

The obtained equations (Kuehl, C. 2003) are written as:

$$\begin{bmatrix} 0 & & & & & -L_z^T \\ & 0 & & & & -L_{\theta_1}^T \\ & & A & & & \cdots \\ & & & \cdots & & \cdots \\ & & & & A & -L_{\theta_n}^T \\ L_z^T & L_{\theta_1}^T & \cdots & \cdots & L_{\theta_n}^T & 0 \end{bmatrix} \begin{bmatrix} \dot{z} \\ \dot{\theta}_1 \\ \cdots \\ \cdots \\ \dot{\theta}_n \\ \lambda \end{bmatrix} = \begin{bmatrix} \frac{\partial E_{grav}}{\partial z} \\ \frac{\partial E_{grav}}{\partial \theta_1} \\ \cdots \\ \cdots \\ \frac{\partial E_{grav}}{\partial \theta_n} \\ E \end{bmatrix} + \begin{bmatrix} 0 & & & & & \\ & 0 & & & & \\ & & -K_s & & & \\ & & & \cdots & & \\ & & & & -K_s & \\ & & & & & 0 \end{bmatrix} \begin{bmatrix} z \\ \theta_1 \\ \cdots \\ \cdots \\ \theta_n \\ 0 \end{bmatrix} + \begin{bmatrix} F_1 \\ C_1 \\ \cdots \\ \cdots \\ C_n \\ 0 \end{bmatrix} \tag{2}$$

They may be rewritten in a more compact equation as:

$$A \bullet \dot{q} = G(q,t) + K_s \bullet (q - q_0) + F(q, \dot{q}, t) \tag{3}$$

In this expression, G represents the gravity effect, K_s represents the elasticity effect of the SMA actuators in uncontrolled configuration and F represents the forces exerted by the SMA actuators in the controlled configuration and the contact forces. The terms related to the gravity effect and to the external forces are not constant; we need to evaluate them at each time step.

These equations constitute a first order differential equation system. The choice of the integration method for solving this system is important to ensure the convergence of the algorithm. As our models are constituted of long open chains, we have to use a precise and stable method. Thus, as the number of links may increase, the time step will have to be reduced. Furthermore, the use of this computational model into an optimization process, will lead the algorithm to compute model of variable length. For all these reasons, we are using a Runge-Kutta algorithm (of 4th order) with adaptive time-step as proposed in (Press, 1992).

The management of the simulator is implemented on the OpenMASK† platform (Margery, D., Arnaldi, B., Chauffaut, A., Donikian, S. & Duval, T. 2002) developed at IRISA. The main objective of OpenMASK is to propose a modular simulation to be executed on various material configurations. The platform manages the synchronization and the exchanges of data between the co-operative processes insuring a "dilated real time" in its standard version. It is developed by using the specificities of the object-oriented programming. The synoptic of its organization for our application is presented in Fig. 5.

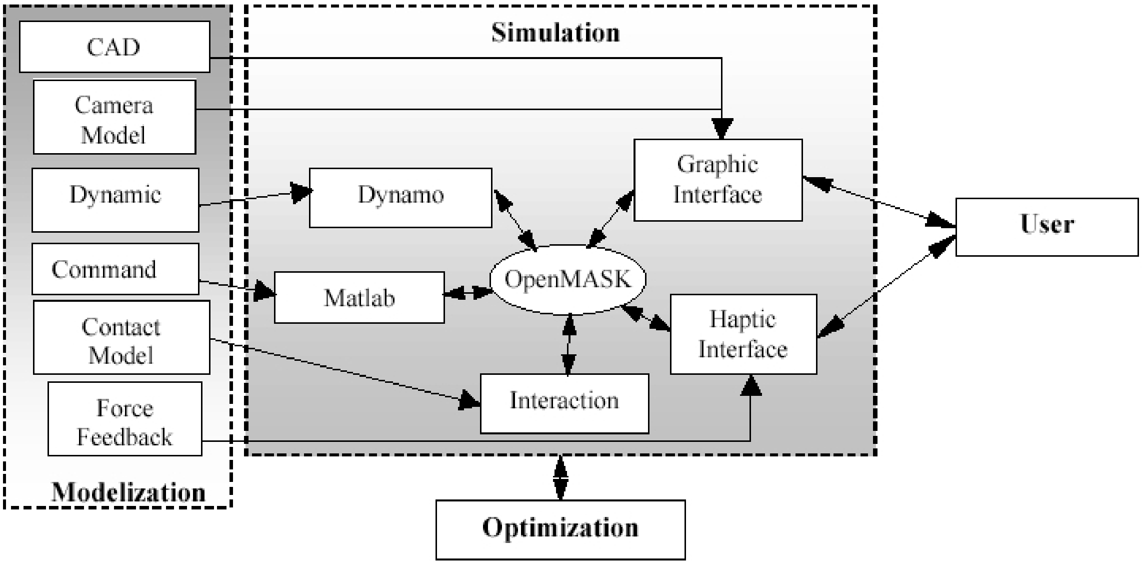


Figure 5. OpenMASK: Modular Animation and Simulation Kit

Various models (Fung, Y. C. 1984) are proposed to describe the behaviour of human organs or tissues, including non-linear behaviour and relaxation. A first approach, because

† OpenMASK, which stands for Modular Animation and Simulation Kit, is delivered as OpenSource Software: http://www.openmask.org

the interaction occurs between a stiff body imposing the motion (the links of the endoscope on one hand) and a soft body (human tissues on the other hand), is to compute the reaction force by a compliance method. This method is applicable with respect to the dynamical model and furthermore is a rapid computation method, which is in the scope of our objective to realize "real-time" simulations.

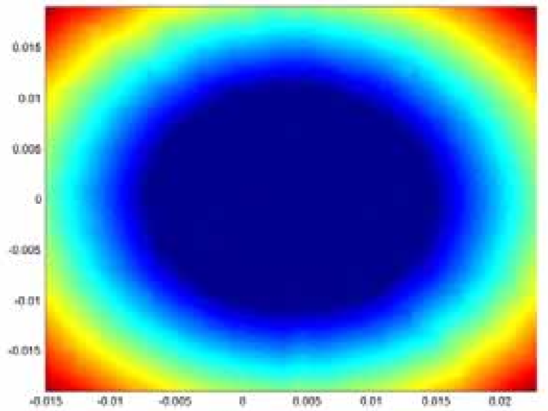


Figure 6. Pre-processed Medical database (one slice)

The proposed method for interaction treatment consists in three steps:

- As geometrical detection is obviously time consuming, the first stage consists in a pre-processing method to build, from the rough MRI (Magnetic Resonance Imaging) medical database, an interpolated distance cartography in the space which is presented, for a slice of the duct, in Fig. 6;
- The second stage of the reaction force computation is ensured by geometric collision detection. The links of the endoscope model are represented by interaction points for which collision is tested by calculating the distances from each interaction point to each "voxel" of the human duct model. This determines the distance from the point on the devices to the duct.

In case of collision, a classical compliance effort is applied to the link, which is proportional to the penetration depth and to the penetration velocity. Let dist be the penetration depth computed during the second stage, $\vec{n}$ be the inward normal to the duct model, k be the global elasticity module of the duct and f be the global viscosity of the duct. The expression of the interaction effort is then: $\vec{F} = -k \bullet \text{dist} \bullet \vec{n} - f \bullet (\vec{v} \bullet \vec{n}) \bullet \vec{n}$. This model, which parameters need to be identified by experimental measurement, seems to be a good one with respect to the stiffness difference between the rigid catheter and the soft human ducts. A simulation result is presented in Fig. 7.

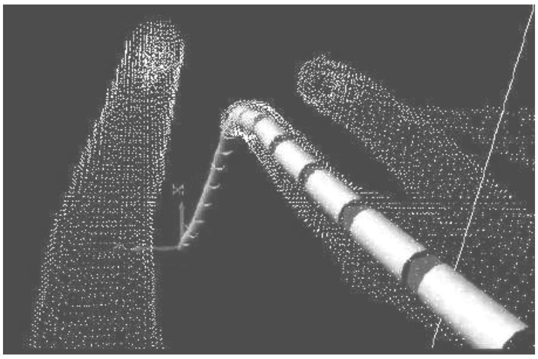


Figure 7. Insertion into a medical database

3. Endoscope Command

The ability to pilot the orientation of the links allows an easier introduction in the virtual duct and constitutes the major interest of using active devices. So, we have developed a module defining a controller, which commands the mechanical model. The controller is based on the geometrical description of the connections between two neighbouring links and on the behavioural characteristics of the SMA actuators, for which a model as been proposed in (Troisfontaine, N. & Bidaud, P. 1998). The behaviour of the controller for a 15 links catheter and for an orientation consign of 10°, illustrated in Fig. 8, shows a response time of 0.4s and a good precision. The stability is good for the links located near to the head. The links located near to the basis shows oscillations, which are due to the important inertia supported.

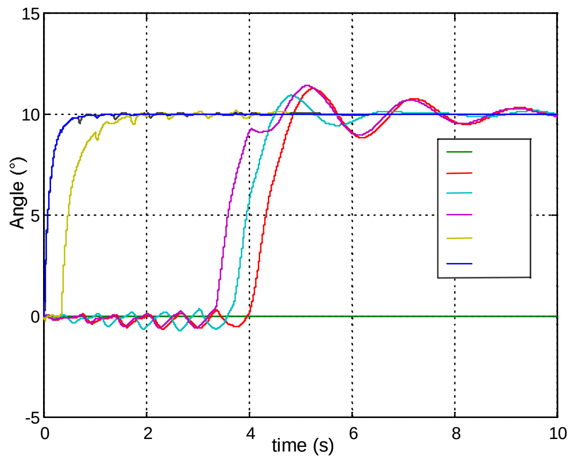


Figure 8. Response time

To take benefit of the active property of the catheter, a command strategy is proposed. Its objective is to automatically adapt the endoscope curvature to the shape of the inspected ducts. In the real world, this will allow to minimise the contact between the endoscopic device and the inspected channel, in order to minimise wound during the operation. In Fig. 9, we present a simulation with a pushed device without any joint control: the applied forces onto the duct, represented by the red vectors, are very important.

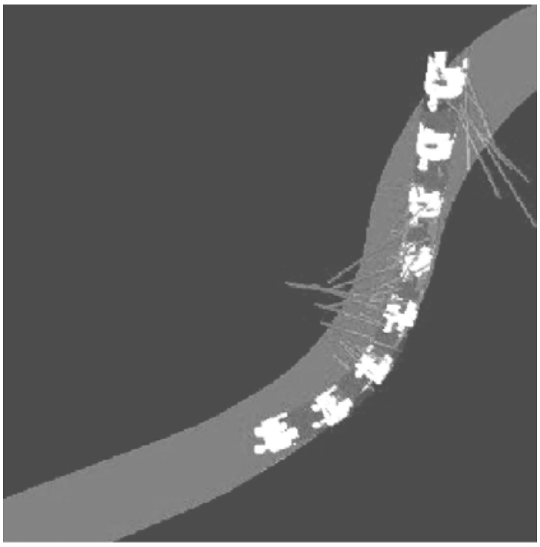


Figure 9. Simulation without joint control

The used command strategy is based on a multi agent approach. It consists in splitting the steering mechanism into independent sub-systems, which constitutes the agents (Duhaut, D. 1993). This distributed modular solution is independent from the structure length and should minimise the amount of information exchanged between the agents. The principle, presented in Fig. 10 for a two dimensional model, is quite simple. If an effort is detected on one of the links, which, in real world, will be achieved by piezoelectric film sensors, an orientation consign is applied to the preceding joint so as to decrease this interaction effort, and the opposite consign is affected to the next joint. The effect is, on one hand, to minimise the contact effort and, on the other hand, to ensure an unchanged orientation of the endoscope head. Numerical tests on stability show that the agent should be constituted of at least two consecutive links in the bi-dimensional case. The three-dimensional extension consists, because of the relative orientations of the pin joints, in agents constituted of four links.

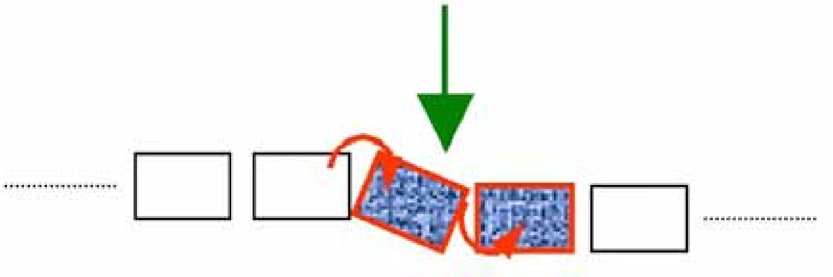


Figure 10. Multi-agent approach (2D principle)

An endoscope piloted with this strategy is proposed, for a three-dimensional model, in Fig. 11. This method, based on the contact detection, seems to be adapted to human ducts inspections and strongly limits the interaction efforts.

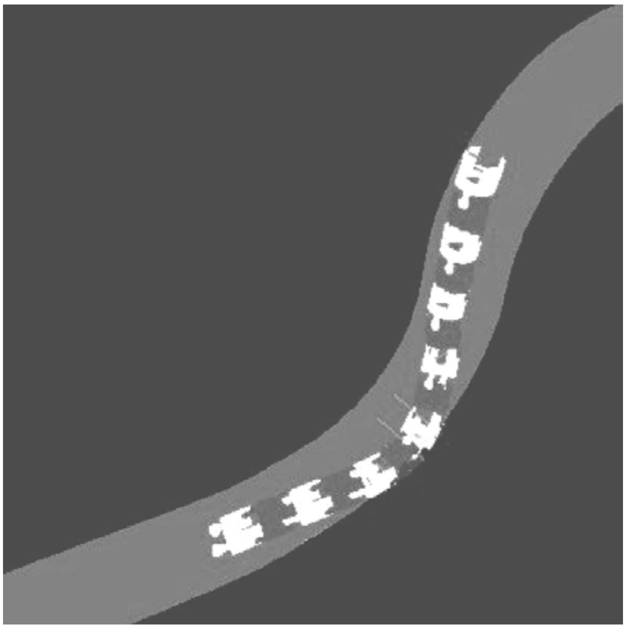


Figure 11. Result with a multi-agent control

4. Optimization by Genetic Algorithms

The techniques of virtual prototyping (Dumont, G. & Kühl, C. 2005) aim at improving the quality of the developed prototypes. In order to test the product functionalities, which could not easily be tested on real prototypes, we use the above described simulation process. The quality of the model, with respect to the task to accomplish (for example: a navigation task) can be measured by an objective function (Endo, K. & Maeno, T. 2001). This function could take into account the developed energy to accomplish the task, which represents the comfort of the surgeon during the operation, and the magnitude of the developed efforts during contact phases, which represents the security of this operation. In order to minimise the objective function, which leads to a better device, we use genetic algorithms (Wall, M. 1996) that have a good solution space exploration capability and lead to acceptable design solutions. Genetic algorithms imitate the natural process of species evolution (Darwin, C. 1859): an individual survives only if it can adapt to the surrounding environment. When it reproduces, its genes are transmitted to its descent. Among these genes, some mutates naturally.

The whole involved process is presented in Fig. 12 and the algorithm is defined as follow:

- Initialisation of the population, this step is performed only once;
- The evaluation of each candidate is done by performing a simulation in order to evaluate the objective, or cost, function;
- The best candidates are selected, by using partially unpredictable selection. This allow to keep mainly the candidates with a good performance, with respect to the specified task, but furthermore to have less good candidates which could lead to good descent;
- Among the so obtained population, the crossover is performed in order to generate a new population being intended in replacing the first one;
- In order not to impoverish the genetic content of the population a non determinist mutation is performed;
- The obtained population constitutes the initialisation of the next algorithm step;
- Convergence criterion may be defined as the number of generation, which ensures that the algorithm stops but does not ensure a strict convergence. Nevertheless, this test shows good results. Another criterion may be based on the average evolution of the population quality between to steps, insuring controlled convergence with inconvenience of possibly being endless.

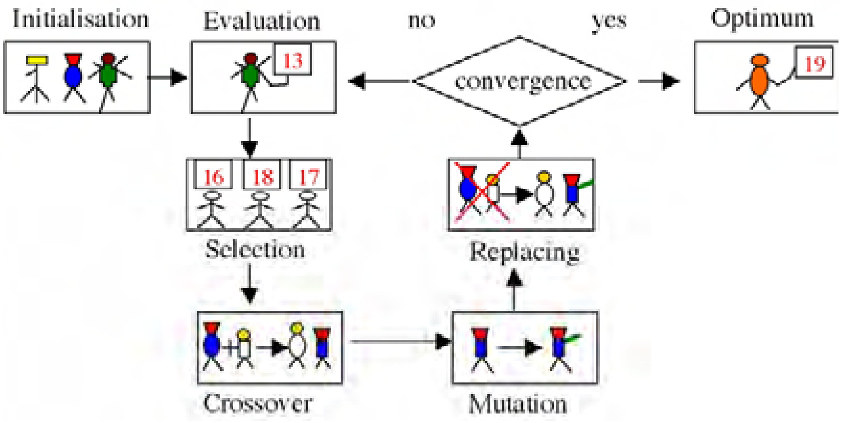


Figure 12. Genetic algorithms

As usual with optimization processes, we have to define optimization variables: for the catheter case, with the modular proposed structure, we can choose the length of each link, the number of used links. This could lead to conceive the best prototype for one given inspection task. As the computation time, involved by this method, is obviously a punishing factor, leading to limit the investigation domain, we have chosen to define various complexity optimization levels:

- The first level consists in a purely geometrical analysis: at this step, for which a result into a 3D-duct model is presented in Fig. 13, we use an objective function built on the length and the number of links.

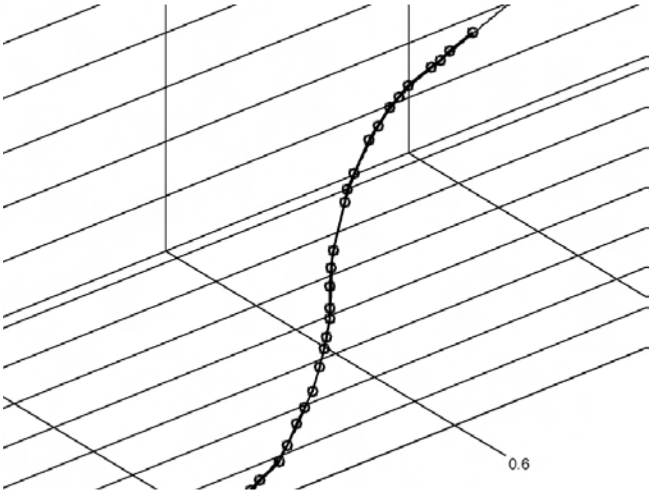


Figure 13. Result for 3D geometrical optimisation

During the progression, the endoscope should interact at least with the patient tissues, so the average line of the channel to inspect defines the trajectory to follow. For this given trajectory, the algorithm aims at defining a prototype, which presents a good following capability, with a minimum number of links. Thus the objective function is defined by:

$$f = \mathrm{good} \bullet \mathrm{i} \bullet \exp^{-\mathrm{k} \bullet \mathrm{dist}} \tag{4}$$

In this equation, i represents the index of the highest trajectory point reached by the endoscope head. It is saturated with the point index corresponding to the desired position. dist corresponds to the sum of the gaps from endoscope to trajectory during progress. good is a function, represented in Fig. 14, increasing the endoscope quality when it has a small number of links in order to decrease the complexity of the structure: $\mathrm{good} = \frac{2}{\pi}(\pi - \arctan(\mathrm{k} \bullet (\mathrm{n}_{\mathrm{artic}} - \mathrm{n}_{\mathrm{max}})))$.

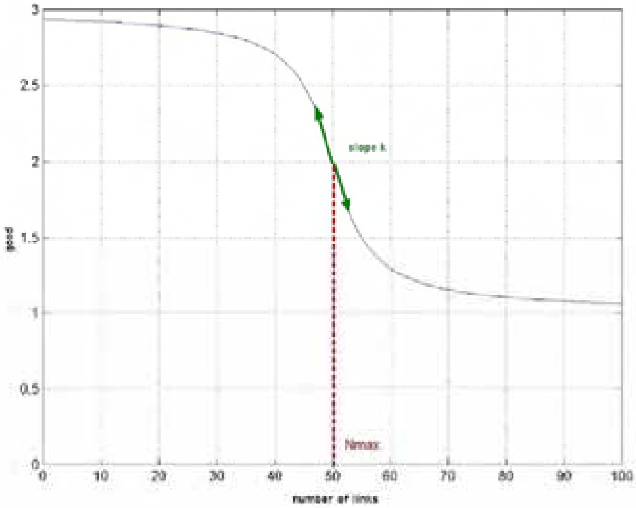


Figure 14. Improvement function

- The second level is a degraded mechanical optimisation: at this stage, the good candidates, issued from the first step, present a good conformation to the trajectory. The purpose is here to determine the actuators parameters insuring a good follow-up of trajectory. It is called a degraded mechanical simulation in the sense that the mechanical model of the device is used but no interaction with the duct is taken into account. This is the purpose of the third level;
- The third level should lie on a complete mechanical optimization: at this step, we dispose of models, which have the ability to follow correctly the predefined trajectory, avoiding the contact. The complete simulation is initialized with the lengths obtained during the first stage and with the mechanical and actuators parameters resulting from the second one. The interaction model is then introduced, and the objective function is the one related to the task to accomplish. It should lead to the best endoscope for this task and furthermore could take into account the mobility of the inspected duct.

In the presented optimization result (Fig. 13), each generation is composed of 40 individuals, and the computation is carried out on 80 generations. This number is chosen as convergence criterion for the optimization process. The optimization variable is the length of each link, which can be chosen between 4 to 18mm, by step of 2mm (coded on 3 bits). The number of used links is variable. The computation time, using an 800 MHz Pentium III with 256 Mo of RAM is around 3 minutes. The optimal solution presented is a 31 segments endoscope and the maximum distance from links to the trajectory is 5 mm. This optimization stage gives satisfactory results: indeed, the procedure does not find the trivial solution which is the endoscope made up of the segments of the smallest size, which solution is the closest to the trajectory but which results in an endoscope having a great number of segments.

5. Conclusion and Future Works

The simulation with multi-agent controller, presented in Fig. 11, proves the validity of this approach for minimizing contacts into the ducts, furthermore it can be extended to the case where the ducts are not rigid but elastically, or visco-elastically deformable, as in reality (Kühl, C. 2003) (Kühl, C & Dumont, G. 2005). A sigmoid untwisting, that is a classical examination in coloscopy, has been reproduced for testing the simulator. The main part of this examination is done when the endoscopic device is withdrawed. The medical practitioner try to reach the ileocolic valve: this part is very delicate because the practitioner has to align the endoscope with the colon in order to continue the progression. Furthermore, the colic angles are very difficult to cross. The final point is the crossing of the sigmoid colon. The technique consists in rolling up the endoscopic device into the sigmoid. Then the practitioner unrolls the buckle by rotating the endoscopic device and by withdrawing it. The sigmoid is untwisted and the rest of the inspection is more easily done. An extract of the simulation is presented in Fig. 15, shows that this operation can be reproduced thanks to the developed simulator.

We have developed a simulator allowing to compute the progression of a poly-articulated endoscope, based on an under development real prototype, actuated by SMA actuators with identified behaviour model. This simulator is based on a mechanical description of the device and on the interacting environment, specific to the considered patient, through a medical MRI acquisition. As mentioned in the introduction, an experimental work is to be done in order to identify the contact model parameters.

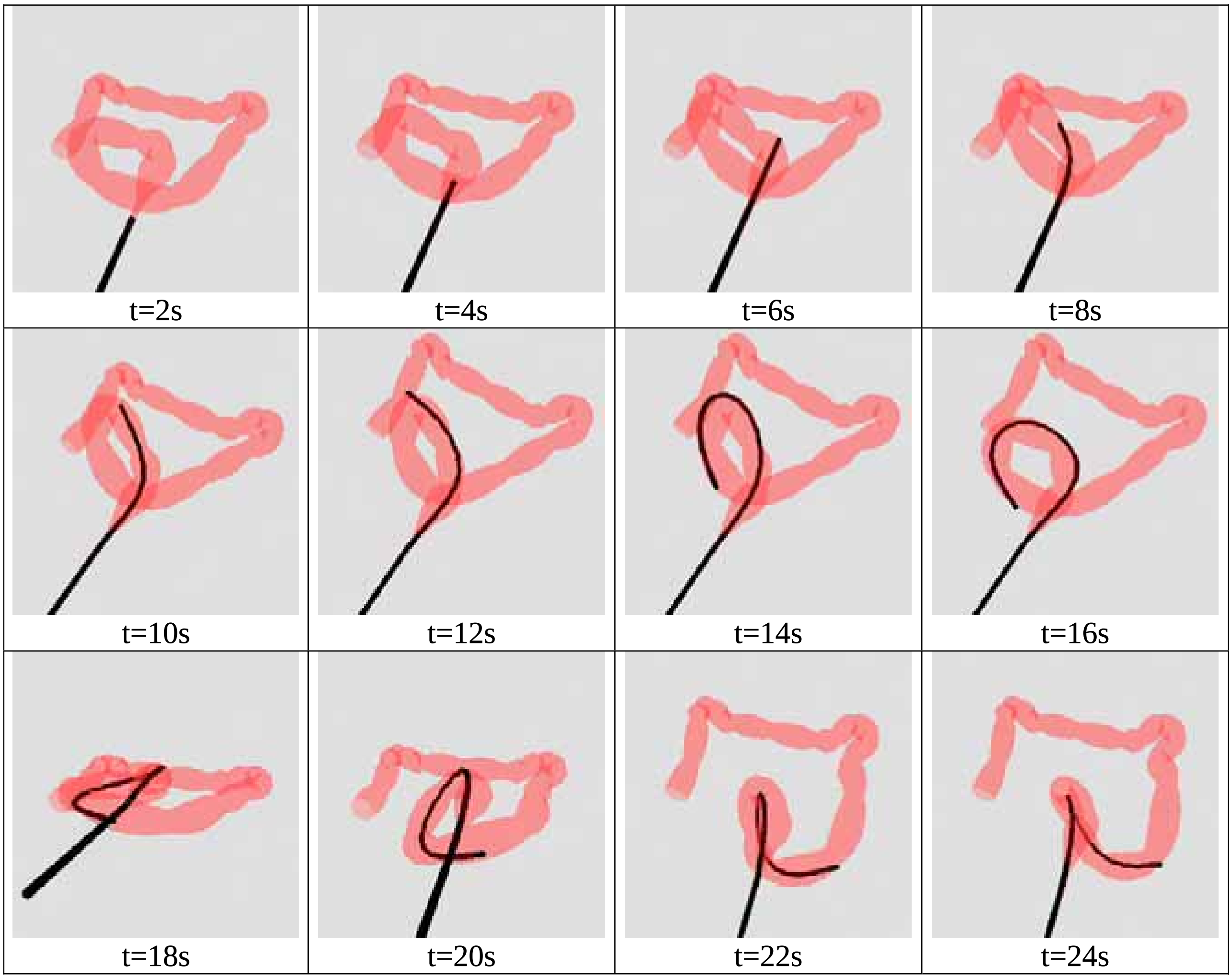

t=2s
t=4s
t=6s
t=8s
t=10s
t=12s
t=14s
t=16s
t=18s
t=20s
t=22s
t=24s

Figure 15. Sigmoid untwisting

Different control strategies for the actuators have been proposed, which results have been presented and which give good results to minimize the contact between the catheter model and the patient database.

In order to improve the accuracy of the considered prototypes, the use of the simulator seems to be a good direction especially when coupled to optimization process. We have proposed different optimization approaches by genetic algorithms and developed some results. Geometrical optimization gives cheering results: it has been shown to predetermine endoscope segments lengths. The second stage of degraded mechanical optimization should allow identifying mechanical parameters adapted for the actuators choice. This optimization procedure will be very similar to the geometrical optimization one, last generation of which will be used to initialize this second stage. Springs and damps will be added to the already calculated lengths. They will be evaluated with criteria taking into account the displacements, the eigen frequencies and the damping in order to obtain satisfactory within the minimum of generation steps, since here time of test for each individual will be considerably increased. Then, they will be generalized to the whole system parameters, as segments diameter or thrusts value, in order to be able to propose the most powerful structure to the surgeon.

The lake of interfacing capabilities, especially in the haptic domain, needs to be examined. It is expected that such devices should improve the control over the catheter in order to define "real" tasks to accomplish in real time. This would also allow to feel the contact between the catheter and the ducts in order to produce a training simulator.

6. References

Cohn, M., Crawford, L., Wendlandt, J. & Sastry, S. S. (1995). Surgical applications of milli-robots. Journal of robotic systems, Vol 12, No 6, pp. 401-416.

Darwin, C. (1859). The origin of species.

Duhaut, D. (1993). Using a multi-agent approach to solve the inverse kinematics, Proceedings of IROS 1993 (Intelligent Robot and System Conference), IROS, pp. 2002-2007.

Dumont, G., Chapelle, F. & Bidaud, P. (2001). Toward virtual prototyping of active endoscopes, Proceedings of ISR2001 (International Symposium on Robotics), IFR (International Federation of Robotics), Seoul, Korea, pp. 821-826. 19-21 april 2001.

Dumont, G., Kühl, C. & Bidaud, P. (2002). Simulating and optimizing active endoscope prototypes, Proceedings of ISR2002 (International Symposium on Robotics), IFR (International Federation of Robotics), Stockholm, Sweden. 7-11 oct 2002.

Dumont, G & Kühl, C., (2005). Finite element simulation for design optimization of shape memory alloy spring actuators. Engineering Computations : International Journal For Computer Aided Engineering And Software (accepted for publication).

Endo, K. & Maeno, T. (2001). Simultaneous generation of morphology of body and neural system of multi-linked locomotive robot using evolutionary computation, Proceedings of ISR2001 (International Symposium on Robotics), IFR (International Federation of Robotics), Seoul, Korea, pp. 499-504. 19-21 april 2001.

Fung, Y. C. (1984), Biomechanics, Mechanical Properties of Living Tissues, Springer Verlag.

Ikuta, K., Iritani, K. & Fukuyama, J. (2001). Mobile virtual endoscope system with haptic and visual information for non-invasive inspection training, Proceedings of ICRA 2001 (International conference on robotics and automation), IEEE, Seoul, Korea, pp. 2037-2044.

Kühl, C & Dumont, G. (2005). Coloscopy simulation : toward endoscope improvement. Journal of Computer Methods In Biomechanics And Biomedical Engineering. Special issue « Computer-aided medicine and surgery » (to appear 2005).

Kühl, C. (2003). Prototypage virtuel d'endoscopes à actionneurs distribués, PhD thesis, École normale supérieure de Cachan.

Kühl, C., Dumont, G., Mognol, P., Gouleau, S. & Furet, B. (2002). Active catheter prototyping : From virtual to real, Proceedings of 4th International Conference On Integrated Design And Manufacturing In Mechanical Engineering, Primeca, IFMA, Clermont-Ferrand, France. 14-16 may 2002.

Lim, G., Minami, K., Yamamoto, K., Sugihara, M., Uchiyama, M. & Esashi, M. (1996). Multi-link active catheter with snake-like motion, Robotica, Vol. 14.

Margery, D., Arnaldi, B., Chauffaut, A., Donikian, S. & Duval, T. (2002). Openmask: Multi-threaded or modular animation and simulation kernel or kit : a general introduction, Proceedings of VRIC, pp. 101-110. http://www.openmask.org

Montesi, M.C., Martini, B., Pellegrinetti, A., Dario, P., Lencioni, L., Montano, A. (1995) An SMA-based flexible active endoscope for minimal invasive surgery, Journal of micromechanic and microengineering, Vol. 5, pp. 180-182, 1995

Park, K. & Esashi, M. (1999). An active catheter with integrated circuit for communication and control, In Technical Digest of the Twelfth IEEE International Conference On Micro Electro Mechanical Systems (MEMS '99), IEEE, pp. 400-405.

Press W. H., Teukolsky S. A., Vetterling W. T., Flannery B. P. (1992). Numerical recipes in C, The art of scientific computing, Second edition, Cambridge university press, 1992

Szewczyk, J., Sars, V. D., Bidaud, P. & Dumont, G. (2000). An active tubular polyarticulated micro-system for flexible endoscope, Proceedings of ISER2000 (7th International Symposium on Experimental Robotics), Hawaii. 10-13 december 2000.

Takehana, S., Ueda, Y., Gotanda, M., Sakurai, T. & Adachi, H. (1990). Apparatus for bending an insertion section of an endoscope using a SMA, United States Patent 05/06/1990 US4930494.

Troisfontaine, N. & Bidaud, P. (1998). Position and force control of sma micro-actuators, In : International Advanced Robotics Programm (IARP), pp. 110-126.

Wall, M. (1996). Galib: a C++ library of genetic algorithm components, Technical report, Massachusetts Institute of Technology, Mechanical Engineering Department.

Development of a Range of Robot and Automation Prototypes for Service Applications

Bing Lam Luk, Alexandar Djordjevic, Shiu Kit Tso & King Pui Liu

1. Introduction

It has been suggested that the robotics and automation research will move more and more towards the service sector, rather than the more structured factory jobs as in the past (Engelberger 1989), (Schraft 1994). A number of service robots have been developed worldwide in recent years, including an increasing number for the healthcare sector. Examples include the HANDY-1 robotic arm for feeding the severely disabled (Whittaker 1992), (Jackson 1993), Meal Assistance Robot System (Ishii *et al.* 1991), MANUS wheelchair mounted robotic arm (Kwee 1998), KARES rehabilitation robotic system (Song *et al.* 1998), Care-O-bot mobile home care system (Schraft *et al.* 1998), THR hip replacement surgical robotic system (Paul *et al.* 1992), CERO robot for assisting partly motion-impaired people to transport light objects in an office environment (Huttenrauch *et al*. 2002), and WorkPartner for interactive work with humans outdoors (Ylonen *et al*. 2002).

While the above examples provide direct services to the end users, the authors in City University of Hong Kong (CityU) have focused their attention on developing facilities to support maintenance or auxiliary-aid services such as cleaning, delivery, and inspection in vast but difficult to reach/manage places. Although these are less glamorous or prominent chores, they are nevertheless essential routine services often associated with a substantial tedium and lack of stimulating challenges. Additionally, these services may also include safety risks, as in the case of washing windows on high-rise buildings, moving through a maze of dilapidated ventilation ducts, or delivering to, or cleaning, a hospital ward accommodating infectious or infection-sensitive patients.

Such safety risks or convenience of service provided further emphasise the benefits of automating these monotonous tasks. CityU robotic systems reviewed in this text have been arranged approximately in the order of diminishing risk to operators from the targeted activities. Some originally developed supporting technology has also been included.

2. "Cleanbot" Climbing Robots for Remote Maintenance and Cleaning

Traditional manual methods of inspection and maintenance of tall buildings normally require the construction of temporary scaffoldings or permanent gondola systems for workers to stand on in mid-air or at high altitude. Inevitably, this increases the cost and slows down the operations. Additionally, the workers are at risk of falling from lifethreatening heights. In order to automate these operations and reduce the overall hazards involved, climbing service robots are needed. Because it is unlikely that any single robot configuration could suit all different types of modern high-rise buildings that tend to be uniquely shaped one-off designs, three types of climbing service robots have been developed at CityU, "Cleanbot I", II and III (Tso *et al.*, 2000). While all three rely on vacuum grippers for climbing, they belong to quite different mechanical designs.

The largest and heaviest of the three robots is Cleanbot I shown in Figure 2.1. It is designed mainly for cleaning large glass-wall surfaces with window frames protruding up to 40 mm. It is pneumatically actuated and has four DOFs: three linear and one rotational about the axis perpendicular to the wall. Weighing about 30kg and measuring 1.3×1.2×0.4-deep m, it can move on glass-walls through a sequence of steps (up to 0.4 m long) by swapping its vertical and horizontal frames (Tso *et al.* 2000). The glass-cleaning unit consists of two special cleaning squeegees attached at both ends of the horizontal cylinder. Cleaning liquid is supplied to the squeegees to clean the glass. A sucking system is attached to the squeegees to collect a high proportion of the sewage.

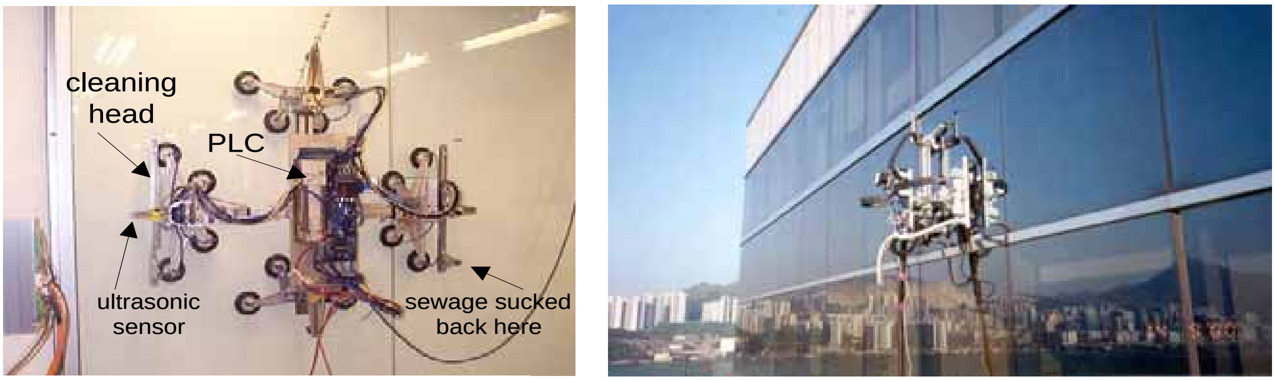


Figure 2.1 "Cleanbot I" Climbing Robot

Two main rodless cylinders are controlled by two sets of solenoid valves in parallel, one of type noted for large air-flow capacity and the other for high-speed response. The motion speed is thus not compromised for precision. The main cylinders are also equipped with pneumatic locking units for more agile and precise stopping. Two additional short cylinders are used for the robot rotation (in 1.6 ° increments) to align it with the window frame. The position sensors on the main cylinders are rotary encoders with rack and pinion mounting. One ultrasonic sensor is located at each end of both main cylinders. Their feedback is used to devise the window-pane cleaning path and cross to the following pane when appropriate.

Cleanbot II shown in Figure 2.2 uses one large vacuum gripper and electrically actuated wheels. It moves on large flat surfaces including tile walls with small air-gaps or unevenness, and can cross 10mm high obstacles. It is

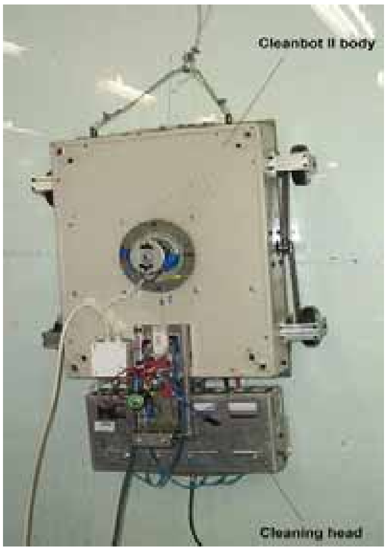


Figure 2.2 Climbing robot "Cleanbot II"

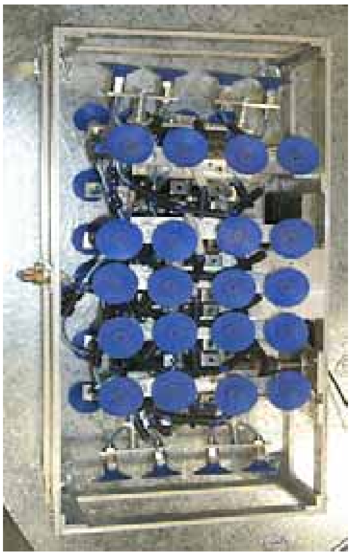


Figure 2.3 Climbing robot Cleanbot III

relatively fast, low cost, and easy to manoeuvre. Cleanbot II provides smooth motion. It is shown in Figure 2.2 with a wet scrubbing cleaning attachment.

Cleanbot III (Figure 2.3) uses a chain-track to move its 52 legs, each with a suction cup and passive compliance (CIDAM 2004). The robot provides smooth continuous motion and can step over window frames and obstacles 35mm high.

3. Wall-tile Debonding Inspection Robot and its Variants for High-rise Buildings

Facades of many concrete high-rise buildings are tiled or similarly clad for decoration and weather protection. Due to factors such as uneven temperature distribution, acid rain, and poor initial workmanship, these elements tend to debond before the end of the expected building life. In order to prevent loose tiles from falling and causing injuries, tile debonding inspection is frequently required. The manual method commonly used involves impacting every tile or wall region with a standardized hammer and listening to the tone-feedback. The sheer size of the building facades (there are approximately 50,000 high-rise buildings in Hong Kong alone, many rising well over 100 m) and the necessity of impacting every part at multiple characteristic locations make this task fatigue- and error-prone. It is also hazardous, requiring work in mid-air at high altitudes. The quality of the outcome is questionable in terms of the consistency of the inspection coverage and the workers' subjectivity in distinguishing the difference in the tone-feedback of impact sounds from the solidly bonded and debonded tile segments during countless hammer-strikes. Yet, bonding failure is a cause of serious concern, as even a single 250 g tile falling from the 10th floor height can gain a deadly momentum of 60 Ns at the ground level.

In order to improve the efficiency, accuracy and safety of this hazardous and markedly fatigue-prone manual inspection work, a robotic-NDT (non-destructive testing) system has been designed and constructed. It mimics the standardized manual inspection method. It is carried by a gondola-like structure which is driven by cable. Its key hardware components, an early hammering module and an impact-echo (feedback) acquisition module, are illustrated in Figure 3.1 together with a representation of an incompletely bonded horizontal tile. Estimating the extent of the void enclosed underneath the tile is the objective of the signal analysis module.

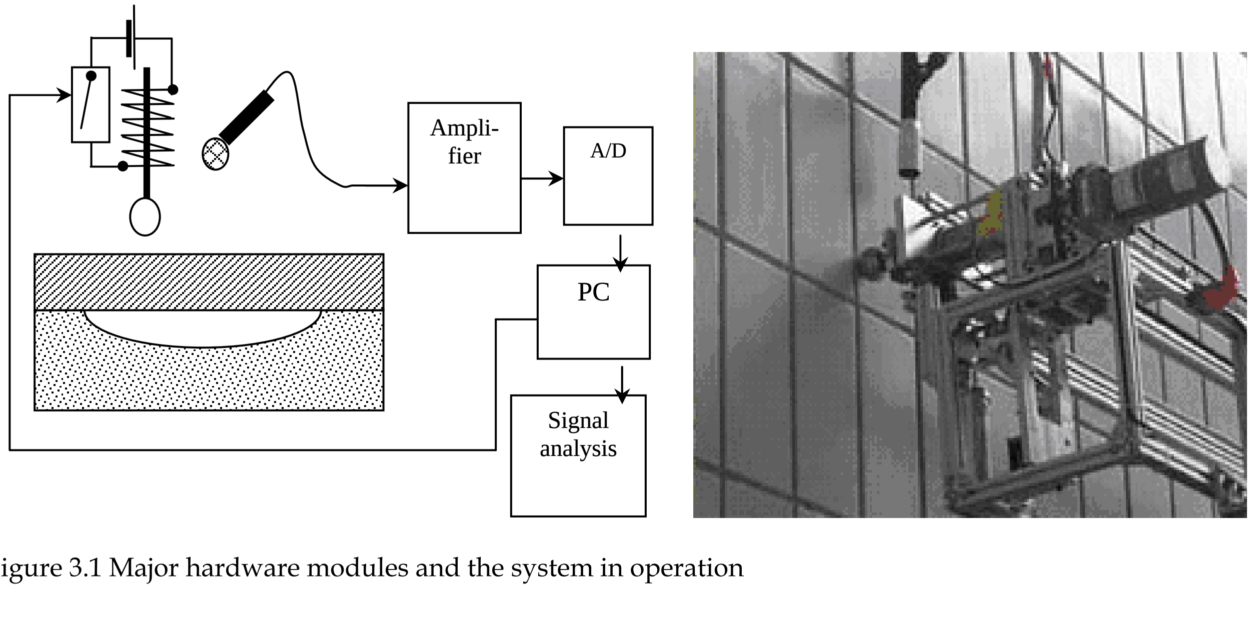


Figure 3.1 Major hardware modules and the system in operation

3.1 Theoretical Background

It can be readily shown that the fundamental frequency of flexural resonance of the tile increases with diminishing size of the void underneath it − for the same tile thickness (Tso *et al.*, 2000). The impact-generating side of the problem is modeled here by a two-degree-of-freedom spring-mass system, Figure 3.2. One spring with stiffness K_f represents the tile deflection, and the other spring with stiffness K_c represents the nonlinear contact stiffness. The two masses, M_2 and M_1, represent the tile and the impacting sphere, respectively.

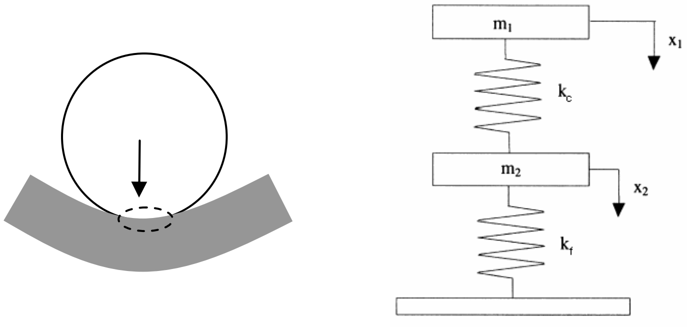


Figure 3.2 The spring-mass model of impact

Considering the energy distribution in the system, the original kinetic energy of the sphere deforms the structure during the impact. Assuming that the structure is elastic, as it reaches its maximum deformation the velocity of the sphere is zero and all of the initial kinetic energy has been converted to the energy stored by the deformation of the structure. Therefore, ignoring the shear and membrane components of structure deformation, the energy balance equation can be given as:

$$E_{sum} = \frac{1}{2}M_1 v_0{}^2 \approx E_f + E_c = E_f + E_{c1} + E_{c2}$$

where v_0 is the initial sphere speed, the subscripts f, c refer to the energy stored in the elastic deformation of the structure and sphere indentation in the contact region (c_1 pertains to the sphere and c_2 to plate).

It can be shown that the ratio of energy converted into flexural vibration depends on the thickness and radius of the plate. In the tile-wall structure, the thin tile layer caused by serious bonding degradation has small thickness and effective stiffness, leading to much stronger flexural vibration under impact compared to a solid tile-wall. Based on acoustics theory, the intensity of sound radiation is proportional to the vibration energy. Thus, the intensity of sound excited by flexural vibration after the impact can be used as an indicator for the structure-integrity identification for the tile-wall.

According to theoretical analysis for a degraded tile-wall, the thin tile layer formed by a void separation underneath will lead to the absorption of most of the kinetic energy of the impacting sphere through the flexural vibration mode of the tile. For a solid tile-wall, however, the loss of kinetic energy of the sphere is very small.

The strength of free vibrations of the sphere caused by impact indentation is also affected by the *vibration energy factor* $\lambda = E_f/E_{sum}$. As a result, the relative intensity of sound radiated from the vibrating sphere and plate can indicate the integrity status of the tiled structure.

Define R_{ps} as the ratio of sound intensities from the sphere and plate. Because the solid tile wall is generally over 20 times thicker than the thin layer of debonded tiles, the ratio of the sound intensities from the sphere and plate after impact R_{ps} will appear significantly different in the presence of debonding. Using this impact sound method, the need to use earlier reported coupling agents or to apply high pressure on tile-walls, can be avoided.

3.2 Void Size Versus Fundamental Frequency

By representing a tile with the void underneath as a thin rectangular plate of thickness *h* with simply supported edges, it has been shown analytically that the fundamental frequency of flexural resonance increases with diminishing size of the void (Rossing *et al.* 1994). Moreover, the shape of the void also has a significant influence on the fundamental frequency.

This finding forms the theoretical basis for operation of the robotic-NDT system shown in Figure 3.1. The system performance has been tested in practice on solid and degraded (with various debond size) tile-wall surfaces. In Figure 3.3, a stable spectrum peak at about 6.7 kHz is attributed to the free vibration of the steel ball. Other resonance frequency components are caused by flexural vibrations of the void-including tile structure. It is evident that with the decreasing void dimension the measured fundamental frequency increases from about 300Hz to 2.3 kHz, 2.9 kHz and 4.0 kHz. The measured and theoretical (with assumed parameters) fundamental frequencies for 7 cases with different void sizes in the specimens and site tests are given in Figure 3.4.

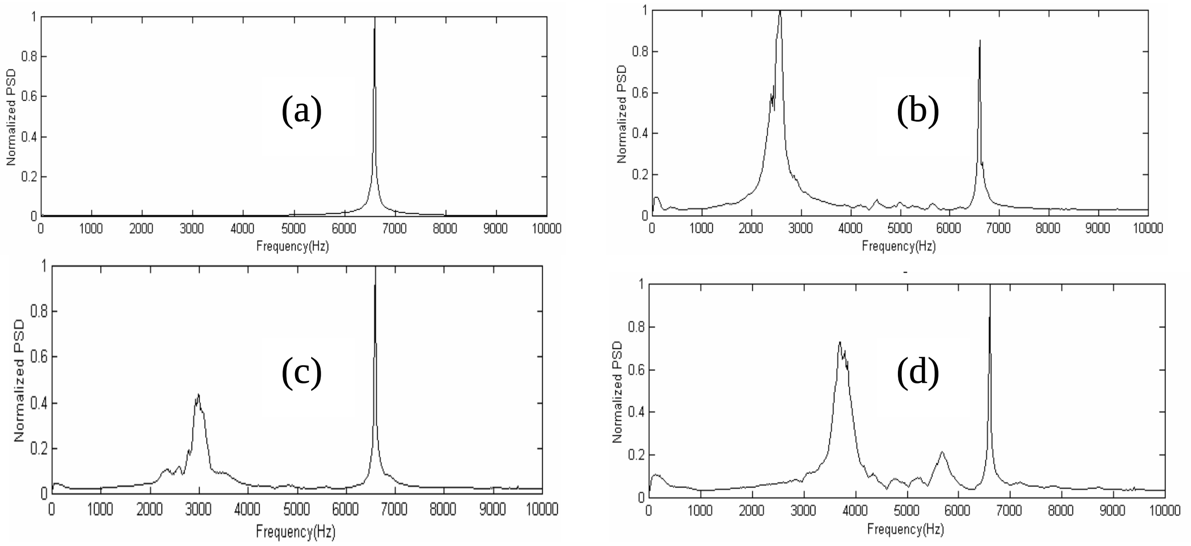


Figure 3.3 Impact sound feedback spectrum from a solid tile wall (a),
from a tile wall with the debond size 160mm×114mm (b),
with a debond 120mm×114mm (c), and with a debond 80mm×114mm (d)

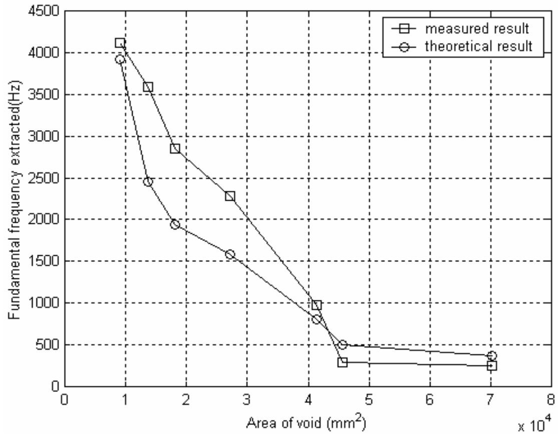


Figure 3.4 Theoretical and measured fundamental frequency versus debond size

The deviations between the theoretical (based on assumed geometry) and measured values are caused by many factors. Background noise and microphone distortions are just some of the disturbance effects. While the system therefore can provide only a rough estimation of the void size under individual tiles, there is little difficulty in identifying whether there is a void or a solid bond underneath.

3.3 Auxiliary Equipment

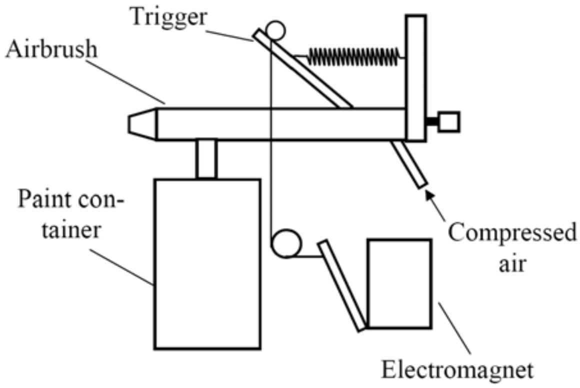


Figure 3.5 Automatic spray gun marks debonded tiles

Typically, one or more camera may be mounted on the robotic-NDT system to enhance its functionality on a high-rise building. In CityU, an automated spray gun shown in Figure 3.5 has been programmed to automatically mark debonded areas of the tiled wall for subsequent repairs, a function often allied to the inspection itself for many end-users. The device combines an airbrush and a solenoid. When energised, the compressed air-flow causes the paint to be sprayed on the wall (Figure 3.5). This auxiliary equipment, just as the robotic-NDT module, is supported on the same gondola-like structure using a cable drive (partially visible in Figure 3.1). As there is no need to support workers, the entire mechanism is much lighter and structurally simpler than would be needed for conventional gondolas with workers.

The full robot system has been designed for semi-autonomous operation. It can be telecontrolled from a base station to perform a raster scan on the wall surface, or the operator can guide it manually in a master-slave control configuration mode. The measurement obtained from the NDT tool is transmitted to the base station via RS422 and displayed graphically for decision making. The master controller is a PC which collects and displays all data in terms of positions, NDT signal and motion control. The on-board computer system is used to monitor the data flow and carry out data conversion. Only processed data are transmitted from the slave controller to the master.

Other fixtures may be developed and incorporated into the basic structure, to perform specific service tasks as assigned according to the application. In particular, a CityU custom-built cleaning mechanism has been used with this basic robotic system with a high degree of containment of undesirable water splash during liquid wash for a high-rise building.

4. Two Service Robots for Hospital Applications

Two service robots for hospitals are described in this section: one for automatic floor cleaning and the other for general-purpose courier services, including the logistical support of telemedicine. These robots can navigate on their own in their application environments. By automating the respective routine but essential services, the aim is to reduce close contacts between non-medical staff and patients, thereby lessening the risk of spreading diseases. The sheer number of casualties and infected among the hospital employees during the recent SARS outbreak, and their suffering, attest to the significance of this aim.

4.1 Automatic Floor Cleaning (AFC) Robot

Adapted to public hospitals in Hong Kong where wet mopping and dry dusting are carried out daily, the cleaning staff must bend to access the areas under the beds (Figure 4.1). Not only does this slow down the cleaning process and is the work physically more demanding, but it also inconveniences the patients and puts the staff at risk of contracting diseases during their unavoidably close proximity to the patients.

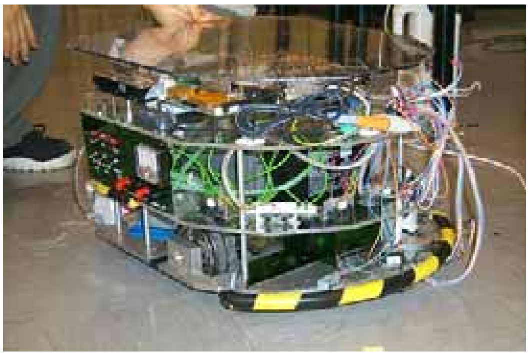


Figure 4.2 Automatic Floor Cleaning Robot Prototype

Shown in Figure 4.2, the developed AFC robot can navigate autonomously, move under beds, avoid obstacles, vacuum dust, and spray disinfectant. Since it is battery powered, its path planning had to be carefully designed to avoid cleaning the same area repeatedly. This task has therefore been partitioned into the "Strategic" and "Reflex" layers shown in Figure 4.3.

Implemented on an embedded PC, the Strategic layer is responsible for the path-planning, position estimation, and all strategic decision-making such as path correction. Implemented on a 68HC11-based "Handy board", the Reflex layer is responsible for all the low-level control and data acquisition of the robot, including the quick actions to deal with dynamic obstacles and unexpected events. Such actions are triggered by the sensor feedback.

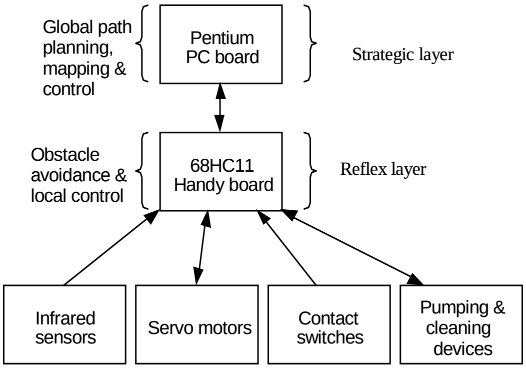


Figure 4.3 Control System

For obstacle detection, the robot is equipped with six IR-based distance-measurement sensors (Sharp-GP2D12) on the sides of the robot. Two additional sensors on top of the robot are for aligning the robot with the bed-edge when entering under beds using a dedicated algorithm. A light bumper, Figure 4.4, is also included. It operates two contact switches that can detect any side of the vehicle subject to impact to avoid conceivable obstacles.

Figure 4.4 Bumper

For the path planning, the cleaning area is divided into 24 squares illustrated in Figure 4.5. This simplifies the mapping process and data communication between the Strategic and Reflex layers. The robot starts at the square 1 and stops at square 24. Should there be no obstacles, the robot would follow the path shown in Figure 4.5 to finish the cleaning task. However, should it encounter obstacles, the robot takes an "obstacle avoidance" path to prevent a potential collision. If an obstacle is suddenly placed in front of the robot, the mechanical switch or the sensory system stops the robot immediately.

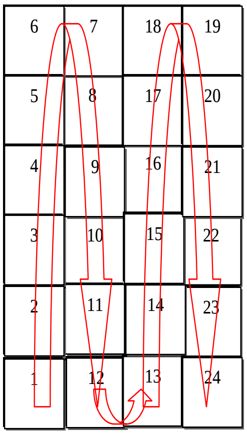


Figure 4.5 Path Planning

Four elementary strategies for avoiding obstacles are shown in Figure 4.6, labelled as "A", "B", "C", and "D". Strategy "A" is used when advancing forward; "B" when the obstacle is where the robot is about to turn to. Strategy "C" is used when the robot's turning path is blocked, and "D" when the obstacle is before a corner. In the first instance at this elementary level, obstacles are assumed to be smaller than the squares in Figure 4.6. Otherwise, they are treated as multiple obstacles, each one smaller than the square.

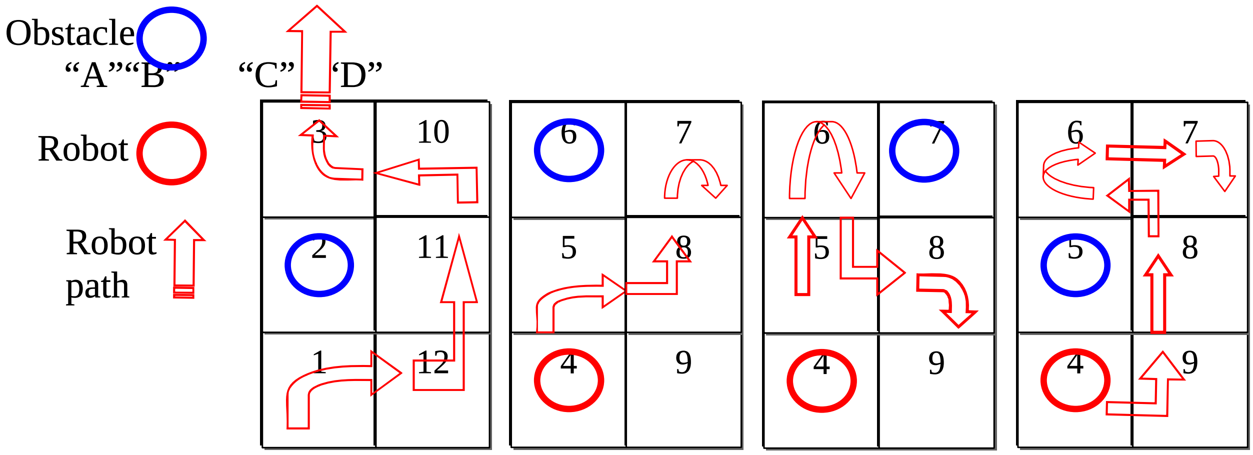


Figure 4.6 Four elementary Strategies

4.2 Multi-Purpose Autonomous Robust Carrier for Hospitals (MARCH)

MARCH is an autonomous transport system designed for door-to-door and floor-to-floor delivery using corridors and elevators. It accepts and carries a trolley with up to 300 kg of load such as meal trays, rubbish, linen, drugs, or other supplies including remotely controlled health-monitoring devices in logistical support of telemedicine.

Shown in Figure 4.9, MARCH robot uses short- and long-range infrared sensors for navigation and obstacle avoidance. The former are installed along the perimeter of the vehicle for avoiding obstacles and for wall following. The long-range sensors are used for detecting obstacles in front of the vehicle to facilitate local path planning.

To cope with the floor-to-floor mission, a specially designed robot arm is mounted on the vehicle to operate the elevator buttons. Since multiple robots could be used in the hospital, a central monitoring system based on 2.4 GHz wireless LAN technology has been installed to facilitate information exchange between the mobile vehicle and the control centre.

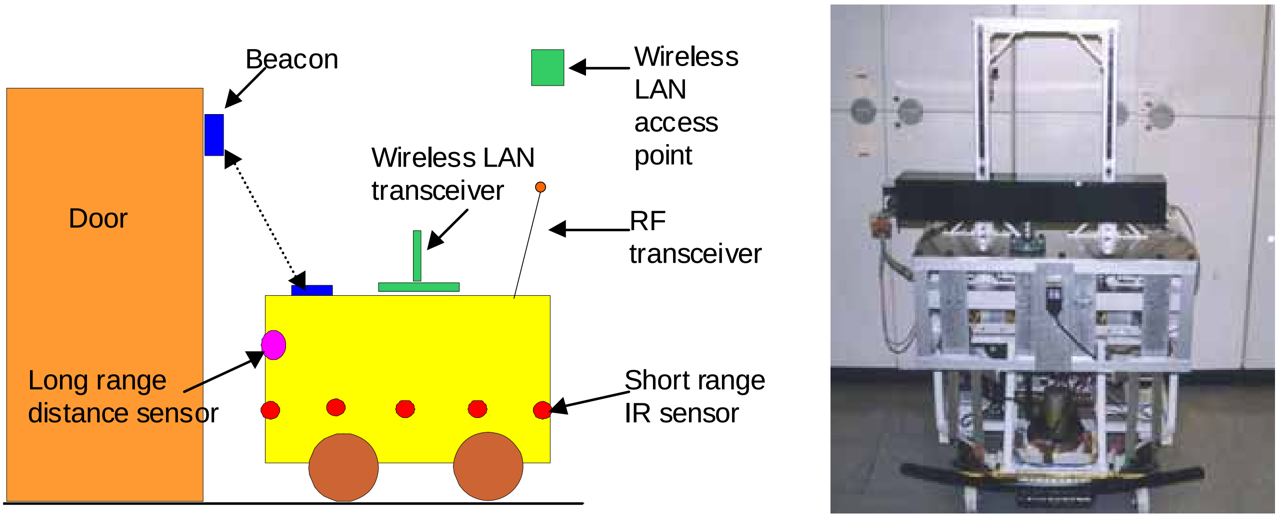


Figure 4.9 Navigation Support Schematics and Photograph of MARCH

The MARCH robot navigation combines the wall-following strategy, local path planning, IR beacon-based landmarks, and central monitoring. The destination and the preferred route are issued from central control room via the wireless LAN. To locate the position of the robot, beacon-based landmarks are adopted and are normally fixed at corners, doorways or places where beacon information is needed. Each beacon contains a unique ID code transmittable to the vehicle through a short-range IR communication device. The vehicle would be able to identify its position in the hospital map based on these ID codes and send it back to the centre via the wireless LAN for monitoring purposes.

A three-input (front/left/right proximity sensor) and two-output (speed, direction) fuzzy logic controller has been designed for collision avoidance. Two schemes of operating lifts are being investigated. One is based on a Cartesian robot manipulator for operating lift buttons and requires a simple vision system for the recognition of lift buttons, which slows its operation but requires no modifications to the lifts. The other scheme relies on the RF interface with the lift control system, which operates faster but requires prior installation work on lifts.

4.3 MARCH Robot and Telemedicine

Telemedicine for the remote delivery of home healthcare is in an advanced stage of development with multifunction devices, some wearable, included for the purpose of pre-diagnosis, patient-monitoring, and tele-consultation. With the MARCH delivery vehicle, this concept can include infections departments of hospitals for the delivery to patients of Bluetooth-enabled medical devices. As part of a larger telemedicine project, CityU team has contributed to the development of a "Medical Plug And Play" communications protocol and a multifunction stethoscope with simultaneous acquisition of ECG and heart and lung sounds. MARCH would be a delivery vehicle for this and videoconferencing equipment, and would host the required portable module for data compression and encryption, and for Bluetooth and Internet communication. With further development, it could also carry a longer teleoperated arm for the remote stethoscope manipulation.

5. Baggage-Carrying AGV

The intended purpose of the service AGV (automated guided vehicle) is to carry passenger luggage at airports, shopping malls, museums or theme parks for people with special needs. As the host passenger moves along corridors (structured or otherwise) on foot or in a wheelchair, the AGV loaded with baggage will follow him/her a small distance behind. Since the human user is within the operation loop, the AGV is not required to perform highly complex tasks such as comprehensive understanding of the environment or precise path planning. A low-cost, simple sensor and control system would suffice for this type of application. Important functions for the AGV are to identify the direction where the host passenger is, and avoid hard collisions. Within reason, accurate maintaining of the distance between the host and the AGV is not fundamental. Hence, the AGV is mainly concerned with two distinct task-achieving behaviours: (i) passenger following and (ii) collision avoidance.

5.1 Passenger Following

For the passenger following task, the passenger must wear a sonar emitter. There is a rotating sonar receiver on the AGV, Figure 5.1. For every full revolution, this receiver memorizes its principal angular orientation, θ, that provided the best alignment with the sonar emitter on the passenger. This reference is then used to generate steering commands for the vehicle motion direction.

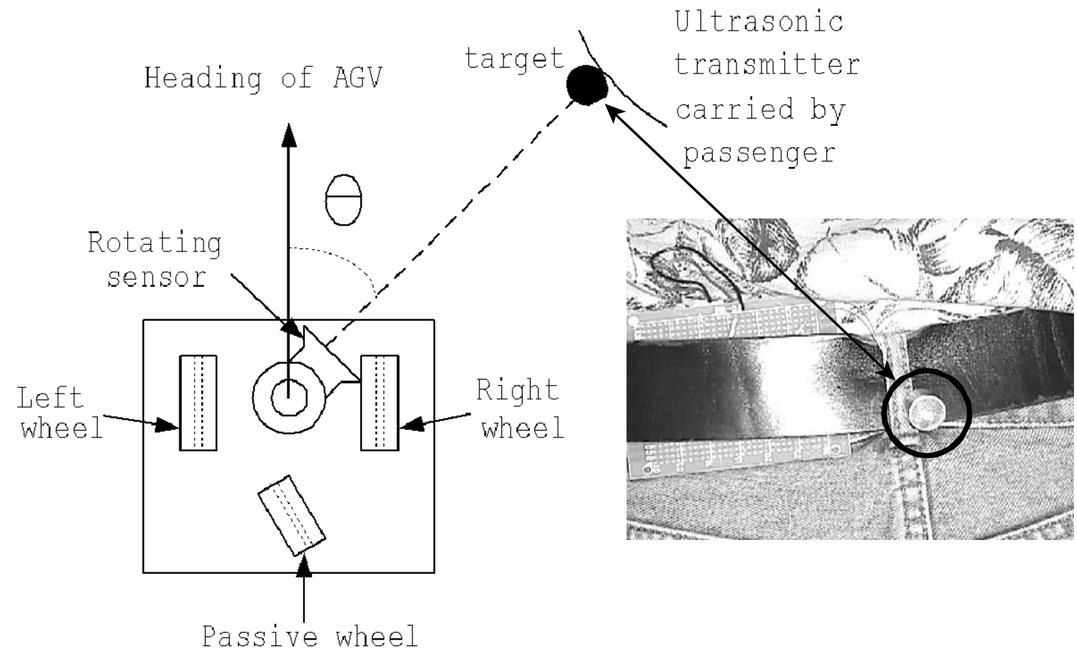


Figure 5.1 The Concept of the Passenger Following Task

The maximum signal intensity received during the sensor revolution, A_{max}, is checked. If it is greater than a preset threshold, the AGV is then deemed to be too close to the passenger and is stopped to wait for the passenger to move further.

5.2 Collision Avoidance

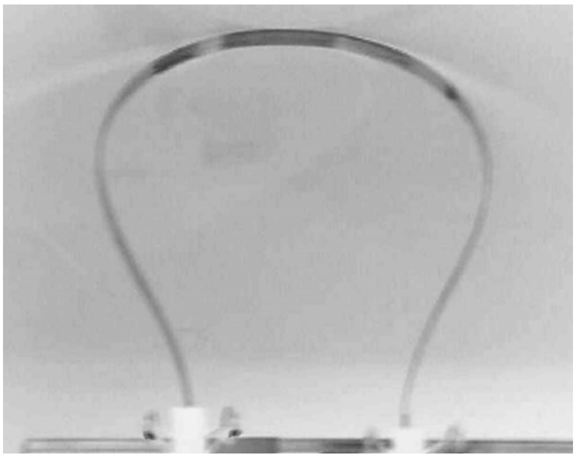


Figure 5.2 Fibre Ribbon Sensitive to Curvature

Many types of bumpers, levers and similar mechanical structures rely on the stiffness of the structural elements to transmit the contact forces onto sensing or switching elements. They may perform satisfactorily at very low AGV speeds when instantaneous stopping of the vehicle is possible. However, the short range of contact/tactile sensing is the constraint that limits AGV speed so that the vehicle stopping distance would not exceed this range. In order to increase the tactile sensing range and allow larger AGV speeds that result in larger vehicle stopping distances, fibre optic 'curvature gauges' (Djordjevich and Boskovic 1996) sensitized to their geometric curvature are arranged in loops around the AGV. One configuration of such loops is shown in Figure 5.2. When the AGV is driven into other objects (obstacles), these loops deform, resulting in the change of their curvature – which the specially sensitized optical fibres detect. Compared to the traditional bumpers and whiskers used in the past for a similar purpose, the difference here is that no intermediate mechanical elements are employed to either transfer the impact loads onto the sensitive element or provide mechanical compliance during the impact. Optical fibres themselves provide both functions simultaneously, resulting in tactile sensing with a range of over 15 cm. This range is two orders of magnitude larger than the range of traditional tactile sensors. Throughout their 15 cm tactility range, curvature gauge loops generate negligible reaction forces with the impacting body. (Djordjevich et al. 2000)

The range mentioned is on top of the one provided by the elastic bumper on which the optical fibres are mounted (Figure 5.3). Configuration of such an elastic bumper is also monitored by a separate curvature gauge taped along it, making it sensitive to deflection within its entire range of deflection. Instead a strip, this elastic bumper can be a sheet wide enough to cover the entire front surface of the AGV. Importantly, such elastic sheet would be sensitive to its own curvature. (Djordjevich *et al.* 2000)

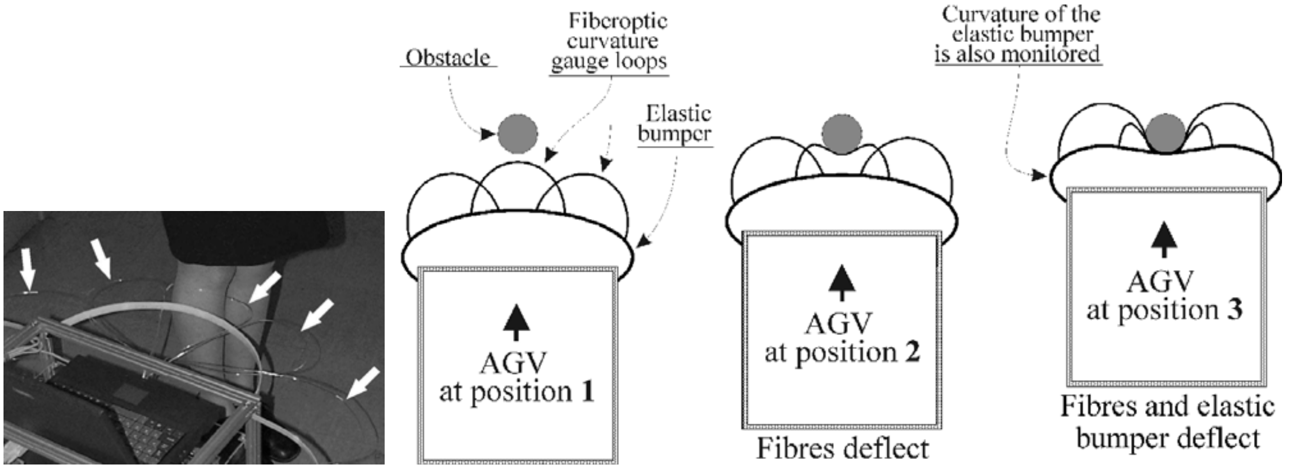

Obstacle
Fibcroptic curvature gauge loops
Elastic bumper
Curvature of the elastic bumper is also monitored
AGV at position 1
AGV at position 2
Fibres deflect
AGV at position 3
Fibres and elastic bumper deflect

Figure 5.3 The elastic bumper and fibre ribbons are sensitive to curvature

When an unexpected obstacle is detected between the passenger and the AGV (points B in Figures 5.4a and 5.4b), the guidance algorithm is such that the vehicle backs by a predetermined distance (to point A in Figure 5.4) and attempts to go around the obstacle with the objective of getting back onto the track behind it and continuing further along. Figure 5.4a shows such a manoeuvre in its left-hand version triggered by the deflection of sensors on the right half of the AGV. Deflecting other frontal sensors, whether alone or in a combination with those on the right-hand side of the vehicle, initiates, as the default case, a manoeuvre illustrated in Figure 5.4b. Depending on the passenger's location with

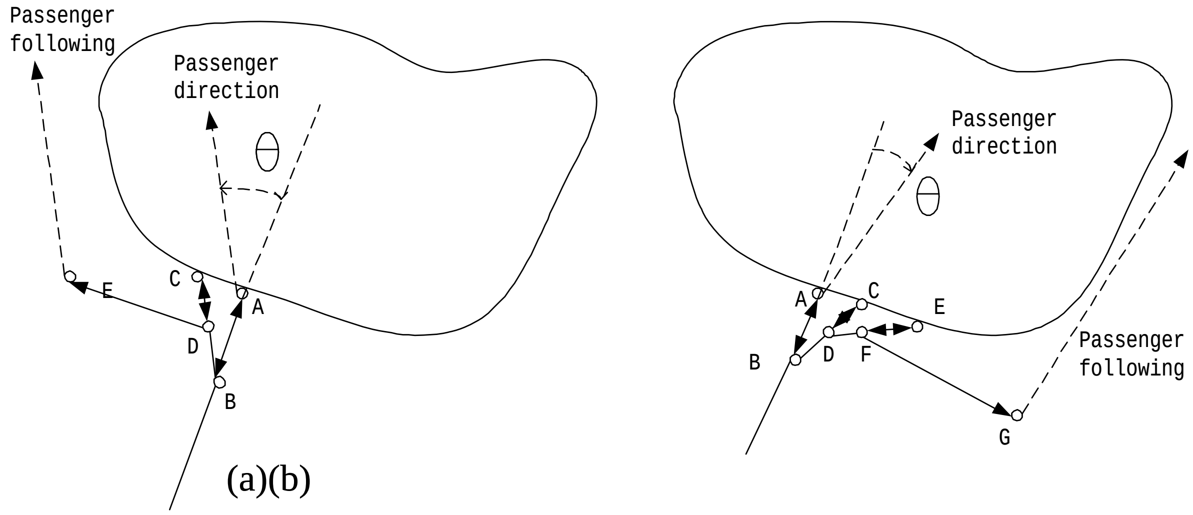


Figure 5.4 Left-hand (a) and Right-hand (b) Obstacle Avoidance Schemes

respect to the AGV principal direction (the sign and magnitude of angle θ), the two avoidance schemes may be swapped.

6. Ventilation-Duct Inspection Robot

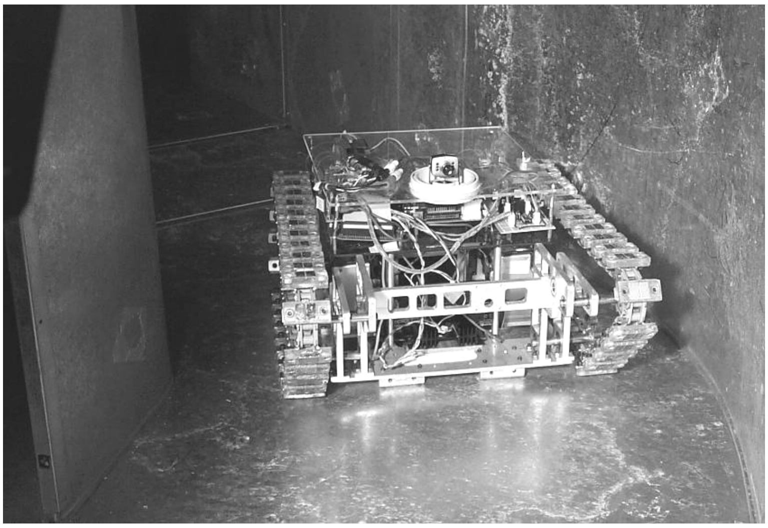


Figure 6.1 Ventilation-duct Inspection Vehicle

A maze of crisscrossing ventilation ducts is a rather common sight in modern high rise commercial and industrial buildings. Their unsightliness apart, a major problem with ventilation ducts is that they enclose often cool and condensate laden environment favourable for the development and spreading of mildew. Mildew irritants represent a serious health hazard for sensitive occupants in the buildings who are left with little choice on how to avoid the exposure or protect themselves. A tele-steerable tracked vehicle capable of navigating through common types of ventilation ducts, whether horizontal, inclined, vertical, or with sharp corners, is shown in Figure 6.1. The main purpose for its development is inspection, although it could easily be adapted to spray fungicides or multi component sealing or rust-proofing agents, or to perform duct cleaning and mechanical repair.

Equipped with a camera, the vehicle relays back to its operator outside the visual internal information about the duct. To allow track adhesion to duct walls while climbing, moving on the sealing, or floor/wall/sealing transitioning, the vehicle has many powerful rare-earth magnets incrusted in its tracks. The tracks are made of flexible plastics for firmer grip on metal sheets. Because the vehicle during the operation is enveloped by the steel sheet metal of the duct acting like the Faraday cage that hampers wireless communication, the vehicle drags its umbilical cord that must be coiled back when reversing the vehicle at the end of the mission.

7. An Intelligent Networking and Automation System for Home and SOHO Environments

The ever increasing desires for improvement in efficiency and comfort for individuals' home and office environments have provided the driving force for the recent development in Home/SOHO (Small Office and Home Office) automation and networking systems (Tso *et al.* 2003). One of the main problems in home/SOHO automation is the wide diversity of technologies, application requirements and limited cost allowance. As a result, it is difficult to use a single network technology to link all the resources together and hence the home/SOHO automation system will need to handle the heterogeneous nature of the home/SOHO network. Moreover, the automation system should be able to add or remove resources in an ad hoc manner in order to make it convenient for users to add or remove home/office devices. Since users of this type of system are usually not experienced in setting up the network for SOHO's devices and resources, the automation system should provide a plug and play feature to avoid any lengthy and complicated configuration process.

A Home/SOHO automation system presented in this Section handles heterogeneous network systems. The application of the UPnP (Universal Plug and Play) open standard makes the network connection of various devices as transparent to the users as possible. The current experimental Home/SOHO system can support Bluetooth wireless network, Lonworks, IR, C-bus, 315MHz, 418MHz and 433MHz ISM band RF, and broadband Internet communication. It is flexible enough to serve as an experimentation platform when developing other Home/SOHO automation technologies.

7.1 Heterogeneous System Architecture

Figure 7.1 illustrates a typical configuration of a heterogeneous Home/SOHO network system. It consists of a number of appliances connected via different networking technologies such as Bluetooth wireless network, Lonworks, CAN Bus, infrared remote control, C-bus, RF, Ethernet network and broadband Internet communication. The home gateway as a central control point of the network is able to communicate with all appliances within this Home/SOHO network environment. As a result, the message exchange between the gateway and the appliances and also the intercommunication between appliances is addressed. Besides, the interoperability between appliances of different manufacturers is also an important issue. In order to address it, a flexible control architecture shown in Figure 7.2 is devised. The architecture is divided into 5 layers: Physical Device Layer, Network Access Layer, Logical Appliance Layer, Common Service Access Layer and User Application Layer. The proposed architecture separates the implementation details of the low-level hardware network from the actual end-user application. This arrangement allows application developers to concentrate on the actual applications rather than on the details of the low-level hardware implementation.

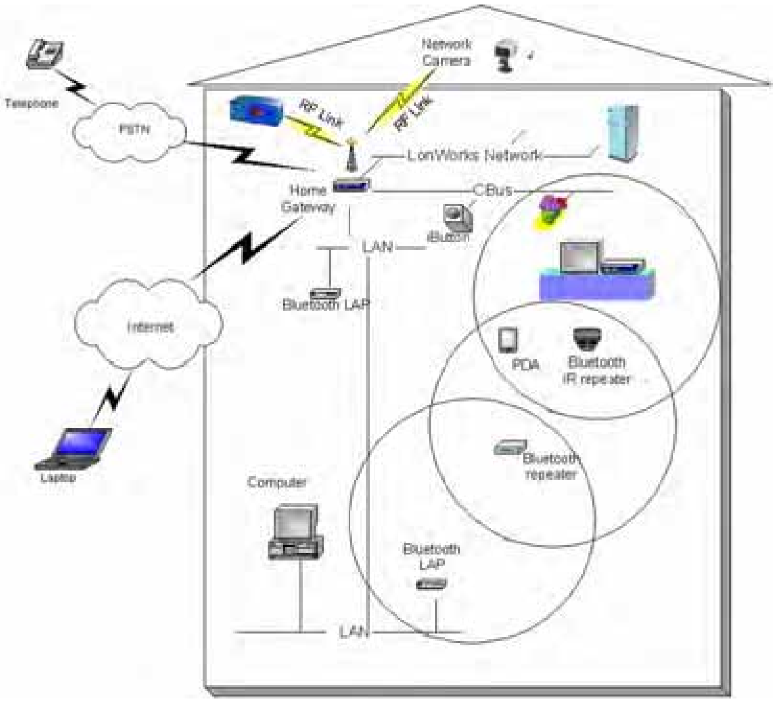


Figure 7.1 A typical heterogeneous Home/SOHO Network System

The Physical Device Layer is the low level driver for communicating with the device. The Network Access Layer provides a unified interface for accessing different network systems. The Logical Appliance Layer provides a unified interface for accessing individual appliance irrespective of its underlying technologies. The Logical Appliance Layer comprises a number of software modules (Control points) for different networking technologies. Irrespective of different networking technologies, all Control Points provide common services to the upper protocol layer, such as automatic discovery of devices, common device access API, event notification etc. The Control Point presents all discovered appliances as logical appliance objects to the Common Service Access Layer and the user applications can access all appliances via the Common Service Access Layer.

The User Application Layer provides the client applications, such as Web browser, WAP browser, Pocket PC application program or tailor-made control console, for the end users to access the resources managed by the system.

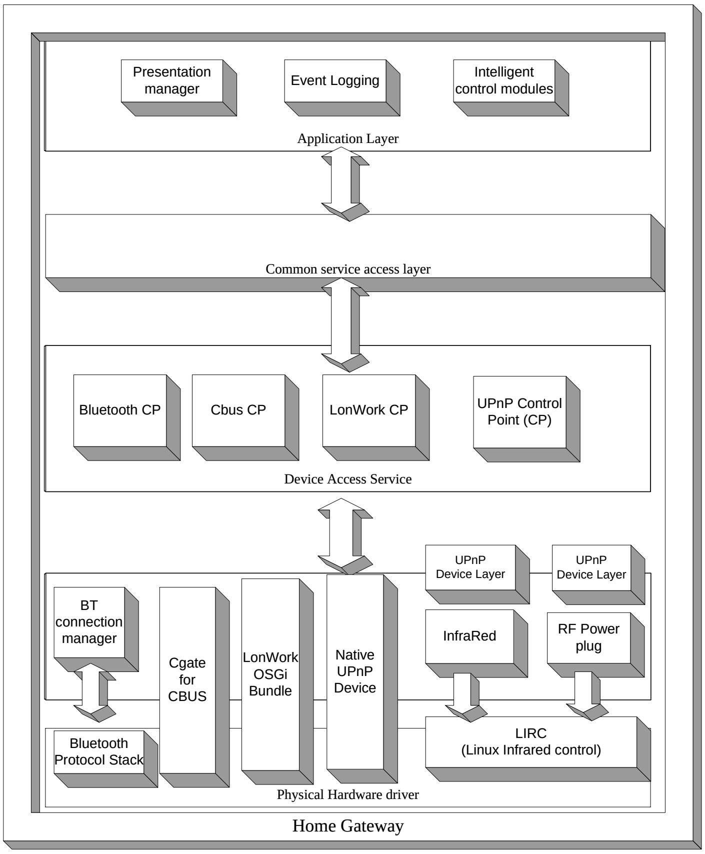


Figure 7.2 System Architecture

7.2 Java-Based Software Implementation to Facilitate Lay Users

The system is implemented on the Linux platform with Java Embedded Server (JES) technology. The Java Embedded Server is an implementation of OSGi architecture. Under the OSGi framework, most of services of the Home gateway are implemented as OSGi 'bundles'. The OSGi framework provides several advantages important for the Home/SOHO gateway, including:

- platform independence;
- service discovery and dependency resolution;
- dynamic service update; and
- sharing of service.

All applications are currently implemented as OSGi bundles under JES, which allows the just-in-time service delivery to the end-user.

User-friendliness is important. Since the end users will not normally be knowledgeable enough to set up a complicated network system for all of the home or office appliances, the whole system should be plug-and-play requiring only simple set up procedures. Hence, the Universal Plug and Play (UPnP) standard has been implemented. It does the features of automatic service discovery, remote access and event notification, which are essential for developing the plug-and-play capability for networking intelligent ad hoc devices. Since UPnP uses standard Internet protocols like TCP/IP, HTTP, XML, SOAP etc, it makes interoperability an inherent feature of the system. For legacy devices, such as IR appliances, additional UPnP bridging software modules are implemented on top of the device drivers in order to make the devices UPnP compliant with the UPnP framework.

Wireless communication is important for the Home/SOHO system because it enables users to add/remove devices from the network dynamically. Bluetooth has certainly attracted worldwide support and a number of home appliance manufacturers, such as Sony, intend to add Bluetooth communication to their future products. Hence, Bluetooth has been adopted as the main wireless communication network. Among the various profiles supported by Bluetooth, Audio/Video Remote Control Profile (AVRCP) is most suitable for the current applications. In this profile, the controller translates the detected user action to the A/V control signal, and then transmits it to a remote Bluetooth device. Two different roles are defined for devices in this profile, a controller (CT) and a target (TG). The CT, such as a personal computer, a PDA, a mobile phone or a remote controller, sends a command to a TG, such as a TV, a headphone or a video player/recorder. Upon receiving a command, the TG responds back to the CT as shown in Figure 7.3.

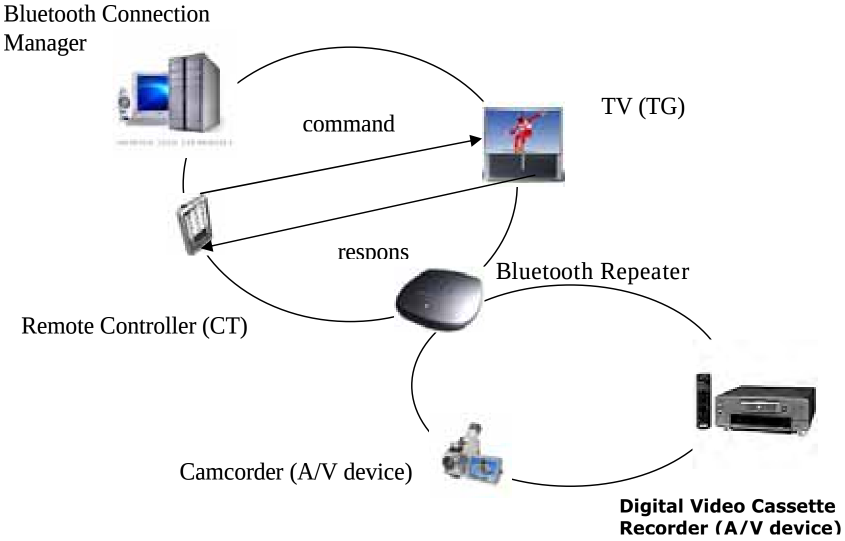


Figure 7.3 Bluetooth home system with AVRCP

Unlike the traditional IR remote controller, this one can support several A/V devices implemented with AVRCP. The communication between CT and various TGs is organised through the Bluetooth's piconet mechanism. In the present system, a Bluetooth connection manager has been developed and implemented with AVRCP. All the A/V devices within the coverage area are visible to this manager. It can control all the devices from the user input and the knowledge captured based on the user habits. Moreover, a Bluetooth repeater with AVRCP has also been developed to extend the coverage area of the whole system. This repeater keeps a list of all the A/V devices and sends the list to the connection manager periodically. The limited coverage area can be expanded by attaching more Bluetooth repeaters to cover a larger area as shown in Figure 7.3. The advantages of using this profile are that all the Bluetooth A/V devices have a standardised model, which makes the devices inherently interoperable, and also future A/V devices with AVRCP could be operated without major modifications.

The AVRCP communication among all devices and TCs is inherently bidirectional. This two-way communication network constitutes the fundamental requirement for development of an intelligent control system. To illustrate this application, an intelligent control module will be described in next section.

7.3 Inclusion of an Intelligent Control Module

The main aim of the intelligent control module is to automate daily operations of the system so that the system execution is more intelligent and pleasing to users. There are two main components of the module: knowledge-capturing and pattern recognition. As mentioned in section 7.2, all communication among devices and TCs are bidirectional.

Therefore, all user inputs to TCs and devices can be recorded. This information is basically an unorganised knowledge requiring pattern recognition to yield the rule description. The proposed intelligent module makes use of knowledge for intelligent operation and enhanced automation. Knowledge, in general, is either pre-installed or run-time captured. While the former source is described by commonly accepted rules, the latter source is initially empty and must be captured dynamically.

For example, a concentration of the indoor carbon dioxide (CO_2) exceeding 1000 PPM is harmful to humans and calls for fresh air intake. Therefore, a ventilation fan equipped with a CO_2 sensor is pre-programmed to operate based on this pre-installed rule. There may be further consequences to such an automated action as the indoor temperature might be altered in the process unintentionally. Because the user is then likely to raise the setting of the air-conditioning system, his/her behaviour would be recorded by the intelligent control module, thus dynamically capturing knowledge (or more precisely, the rule). The captured information is binary in nature and it can be represented in the form of a truth table. Pattern recognition technique is applied to build up the truth table.

Consider a typical truth-table data in Table 7.1 captured by the daily operation of a home/SOHO scenario. The events represented are: (1) an alarm clock goes on and then off; (2) bedroom light is switched on; (3) bathroom light is switched on; (4) hot water is used for shower. Some associated information, such as the time and day of the week following a predefined triggering event (the alarm clock in this case), is also recorded. This is then compared to two subsequently recorded patterns. Simple AND operation is carried out for each entry so that the event relations are extracted. In this case, the extracted rule is that when the alarm of the clock is switched off, the bedroom light should is switched on, then the bathroom light and water heater. The function of the associated information such as the time and day of the week is to restrict and narrow the record comparison.

Time index	Alarm clock ON	Bed room light ON	Bath room light ON	Water heater ON	•	Exhaust fan ON	A/C Power raised
1	1	0	0	0	•	0	0
2	0	0	0	0	•	0	0
3	0	1	0	0	•	0	0
4	0	1	1	0	•	0	0
5	0	1	1	1	•	0	0
•	•	•	•	•	•	0	0
•	•	•	•	•	•	1	0
•	•	•	•	•	•	1	1
•	•	•	•	•	•	•	•

Table 7.1. A typical example of captured user behaviour information

7.4 Presentation Manager

In the application layer, the intelligent control module is developed such that all devices are discovered and connected in the system as mentioned in section 7.3. Another important application in this layer is the user interface (Presentation manager) shown in Figure 7.2. Different features of user interface can be summarized as:

a) regular web browsers on PCs using HTML (HyperText Markup Language) –provide a very common means of controlling devices remotely from anywhere in the world;
b) web browsers on PDA using HTML and web enabled cellular phones using WML (Wireless Markup Language) – provide mobile solutions for users to control devices with limited functions;
c) Java client for systems with installed JVM (Java Virtual Machine) – requires that users pre-install a client application on the client machine (Java native client applications can generally provide an interface with faster response, and enable that voice technology be embedded in the Java client applications);
d) voice enabled bundle in server – all devices can be controlled through a microphone from the server.

Overall, the multi-layer software architecture applied and Java based implementation provides a flexible platform for developing ad hoc heterogeneous network system for intelligent devices. The two-way communication feature among all components (TC or TG) constitutes the infrastructure for developing an intelligent control module. Moreover, it allows the development of the user-friendly and reliable interface so that the status of all devices could be monitored by the user even remotely.

Acknowledgements

The MARCH robot, baggage carrying robot, Cleanbot III and safety support vehicle were financially supported by the City University of Hong Kong grant for CIDAM research unit (CityU #9360077), an RGC grant #9040407 and an RGC Central Allocation grant (CUHK 3/98C). Cleanbot I, Cleanbot II and WIC robots were funded by RGC grants #9040407 and #9040711, an ISF grant (AF/76/98) and an ITF grant (ITS/33/01) respectively. The home automation project was supported by an ITF grant (UIM-55). The telemedicine project was funded by an ITF grant (ITS/99/02).

The authors would like to express their sincere gratitude to the following persons:

- Prof. Y.T. Zhang and his research team, Mr. D. Lam, Mr. A. Choy and Dr. Y.H. Fung for their important collaboration and advice on the development of the telemedicine project;
- Dr. Y.H. Fung, Mr. K.L. Lee, Prof. P. Sooraksa and his research team for their help in some stages of the MARCH project;
- Prof. G.H. Zong and his research team in Beijing University of Aeronautics and Astronautics, Dr. W.L. Xu, Dr. Y.H. Fung, Dr. K.L. Fan, Dr. H. Shao, Dr Y.D. Zhang, Mr. W.L. Chow, Mr. J. Zhu, Mr. X.C. Meng, Dr. D.W. Tang, Dr. X.J. He and Mr. G. Cheung for their contributions on the Cleanbot I/II/III development;
- Mr. A. Ng, Dr. S.R. Yan and Prof. Y.S. Xu for their contributions and suggestions on the development of the safety supporting vehicle;
- Mr. A. Kar, Mr. V. Pjevalica and Dr. Y.H. Fung for their help in developing the baggage carrying robot;
- Dr. Y.H. Fung, Dr. K.L. Fan, Dr. L. Han, Dr. F. Tong and Mr. G. Cheung for his their contributions on the WIC robot development;
- Mr. D. Lam, Mr. A. Choy and Dr Y.H. Fung for their help and contributions in developing the home automation project.
- Dr. J. Zhang and Dr. K. Wong for their contributions to the development of the extended tactility.

8. References

CIDAM (2004). Promotional Brochure, City University of Hong Kong.

Djordjevich, A.; & Boskovic, M. (1996). Curvature Gauge, *Sensors and Actuators - Physical*, Vol. 51, pp. 193–198, 1996.

Djordjevich, A.; Tso, S.K.; & Zhang J. (2000). Extended range tactility in material handling, *International Journal of Production Research*, Vol. 38, No 17, pp. 4357-4367, Nov. 2000.

Engelberger, J.F. (1989). *Robotics in Service*, Cambridge, MA, MIT Press.

Huttenrauch, H.; & Eklundh, K.S. (2002). Fetch-and-carry with CERO: observations from a long-term user study with a service robot, *Proceedings of 11th IEEE International Workshop on Robot and Human Interactive Communication,* pp. 158-163, Sept 2002.

Ishii S.; Hiramatsu, F.; Tanake, S.; Amari, Y.; & Masuda, I. (1991). A meal assistance robot system for handicapped people's welfare, *Proceedings of Conference on Robots and Mechatronics*, Japan Society of Mechanical Engineers, 1991.

Jackson, R.D. (1993). Robotics and its role in helping disabled people, *Engineering Science and Educational Journal*, December 1993.

Kwee, H.H. (1998). Integrated control of MANUS manipulator and wheelchair enhanced by environmental docking, *Robotica*, vol.16, no.5, pp 491-498, 1998.

Paul, H.A.; Mittlestdt, B.; Bargar, W.L.; Musits, B.; Taylor, R.H.; Kazanzides, P.; Zuhars J.; Wlliamson, B.; & Hanson, W. (1992). A surgical robot for total hip replacement surgery, *Proceedings of IEEE International Conference on Robotics and Automation*, Nice, France, vol.1, pp 606-611, May 1992.

Rossing, T.D.; & Fletcher, N.H. (1994). *Principles of vibration and sound*, Springer-Verlag, New York, 1994.

Schraft, R.D.; Schaffer, C.; & May, T. (1998). Care-o-bot: The concept of a system for assisting elderly or disabled persons in home environments", *Proceedings of the 24th Annual Conference of the IEEE Industrial Electronics Society*, vol.4, pp 2476-2481, 1998.

Schraft, R.D. (1994). Mechatronics and robotics for service applications, *IEEE Robotics and Automation Magazine*, pp 31- 35, December 1994.

Song, W.K.; Lee, H.Y.; Kim, J.S.; Yoon, Y.S.; & Bien, Z. (1998). KARES: Intelligent rehabilitation robotic system for the disabled and the elderly, *Proceedings of the 20th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, vol. 5, pp.2682-2685, 1998.

Tso S.K.; Fung Y.H.; Chow W.L.; Zong G.H.; & Liu R. (2000). Design and implementation of a glasswall cleaning robot for high-rise buildings, *Proceedings of the World Automation Congress Eighth International Symposium on Robotics with Applications*, Maui, Hawaii, paper ID: ISORA123, June 2000.

Tso, S.K. (a); Luk, S.L.; Choy, W.H.; Liu, K.P.; Chow, C.S.; Leung, K.F.; Lee, G.; Yau, F.; & Lam, C.W. (2003). An intelligent networking and automation system for home and SOHO environments, *The Fourth International Conference on Control and Automation (ICCA),* Montreal, Canada, 2003.

Tso, S.K. (b); & Tong,F. (2003). Wall Inspection Robot for Improved Maintenance of High Rise Buildings, *20th International Symposium on Automation and Robotics in Construction (ISARC2003)*, Delft, Holland, pp. 449-455, Sept 2003.

Whittaker, M. (1992). HANDY 1 robotic aid to eating: A study in social impact, *Proceedings of RESNA International'92*, pp 589-594, June 1992.

Ylonen, S.J.; & Halme, A.J. (2002). WorkPartner - Centaur like service robot, *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and System*, Vol. 1, pp. 727-732, Sept/Oct 2002.

-VIII-

Legged Robots

Legged Robotic Systems

Giuseppe Carbone & Marco Ceccarelli

1. Introduction

Walking machines have been attempted since the beginning of the technology of transportation machinery with the aim to overpass the limits of wheeled systems by looking at legged solutions in nature. But only since the last part of the 20-th century very efficient walking machines have been conceived, designed, and built with good performances that are suitable for practical applications carrying significant payload with relevant flexibility and versatility.

In this chapter we have presented a survey of the variety of current solutions and prototypes of walking machines and we have illustrated fundamental characteristics and problems for their design and operation. The worldwide feasibility of walking machines is presented by discussing the activity at LARM: Laboratory of Robotics and Mechatronics in Cassino (Italy) as concerning with low-cost easy-operation solutions that can really make the walking machines available to non expert users for many applications.

2. Walking in Nature

Movement is a fundamental distinguishing feature of animal life. The locomotion over a surface by means of limbs or legs can be defined as walking whatever are the number of limbs or legs that are used. Different ways of walking have been achieved by the evolutionary process in nature. The vertebrate animals have a spinal column and one or two pairs of limbs that are used for the walking. These limbs are located beneath the body. Arthropoda animals including crustaceans, insects, centipedes, millipedes, symphylans, pauropodans and trilobites are characterized by a segmented body that is covered by a jointed external skeleton (exoskeleton), with paired jointed limbs on each segment so that they can have an high number of limbs. In this type of animals a stable walking can be achieved with a minimum of six limbs that are located in a side position with respect to the animal's body since they cannot use the flexibility of the spinal column for regulating the masses' positions during the walking.

A large variety of efficient mechanical and physiological designs have evolved in nature in order to fit with the characteristics of a given physical environment and different locomotion modes. Animals seem to have evolved to be as fast as possible, to have the best possible acceleration, maneuverability and endurance, and to have energy consumption as low as possible. However, these objectives are not always compatible with each others. For example tortoises are designed to walk with energy consumption as low as possible but they cannot be fast. Similarly, an animal that has been adapted to sprint as fast as possible, has not good endurance. Usually, nature evolution can be expected to have preferred compromises between the requirements of speed, endurance, and energy consumption. Thus, a wide range of different solutions can be found in nature.

The locomotion for a legged animal can be analyzed in term of three main components:
the power source, the transmission system, the power outputs.

The power source is located in the muscles where chemical energy is converted in mechanical energy. The power outputs are the parts of an animal that are directly in contact with the environment and produces the motion. The transmission system transmits the mechanical energy from the muscles to the power outputs. For vertebrate animals this transmission system is composed of bones and articulations, and the power outputs are usually the feet.

Legged locomotion systems that have evolved in nature, show very good performances in terms of stability, payload capabilities, dynamic behavior. Thus, usually they are considered a very important source of inspiration for designing legged robotic systems mainly for aspects ranging from the mechatronic design to the path planning and gait generation. Several researchers have stressed these topics by using a multidisciplinary approach. For example, several studies have been addressed to the transmission system of vertebrate legged animals from a kinematic point of view. In fact, bones and articulations can be easily modeled as links and joints of a kinematic architecture. Examples of for biped, quadruped and hexapod locomotions in nature are shown in Fig.1 to 5 with their simplified kinematic architectures. Those animals have been and still are inspiration both for design and operation of walking legged systems. In the following main features are reported for each animal but more details can be and have been considered in inspiring/mimicking for walking legged systems.

In particular, Fig.1a) shows the most attractive biped locomotion: a human being. Figure 1b) also shows a kinematic scheme of a human being. Each leg in Fig.1b) can be considered to have seven degrees of freedom. In a human being the muscles are distributed so that the forward motion is more efficient than the backward and side motion. The maximum speed is about 11.0 m/s during a 100 meter run. The average weight of a human being is 650 N. Main characteristics of the human being in terms of biped locomotion are reported in Tab. 1 in which data refer to general common operation. Maximum values of performances strongly depend on situations, environments of life, and training and they can reach even values higher then in Tab.1.

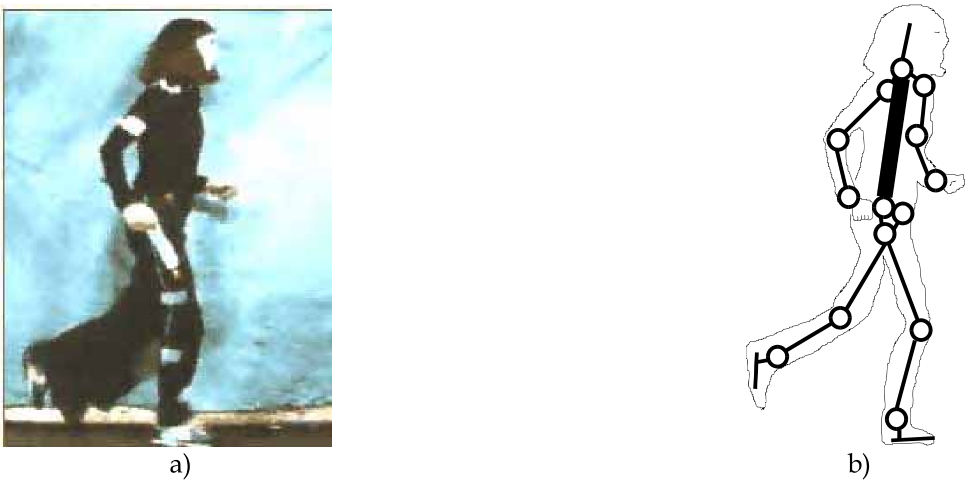

a) b)

Figure 1. Biped locomotion by a human being: a) a picture[1]; b) a kinematic scheme

[1] All the pictures have been taken from webpages that are available in internet.

Figure 2a) and b) show an ostrich and a simplified kinematic scheme of its two legs, respectively. Its kinematic scheme is similar to the human being and each limb can be considered to have seven degrees of freedom. Even in this case the muscle distribution facilitates the forward motion. The average weight is 1,800 N. but the mass distribution provides a lower center of mass and a better attitude to run compared with humans. Even if ostriches can be classified as birds they have lost their ability to fly but they can still escape predators with their fast running at a maximum speed of 19.4 m/s during a 800 meter run. Main characteristics of the ostrich are reported in Table 1.

It is worth noting that the stability of a body in the space can be guaranteed with a minimum of three points in contact with the ground. It this case, the walking stability is obtained if the projection of the center of mass of the body lays within the area obtained by connecting the contact points. However, the biped locomotion can provide only one or two limbs in contact with the ground. Thus, biped locomotion cannot be considered as statically stable. Indeed, bipeds do not fall down since they can control the posture of their upper body in order to keep the balance in dynamic conditions. This require clever control strategies based on the feedback of several vision, auditory, and tactile sensors. In addition, a control of the compliance through spinal cord, muscles, and feet is used for the compensation of dynamic effects. Beside the complexity of biped stable walking (but also run), the biped structure show the most flexible locomotion in term of obstacle avoidance and fast reshaping of walking mode.

Figure 3 a) and b) show a young child as an example of quadruped locomotion and a simplified kinematics scheme of his four limbs, respectively. In this case, feet and arms are both used as limbs for the locomotion. A clever example of using arms like legs can be recognized in monks, which are often considered of inspiration for robotic systems with variable capabilities.

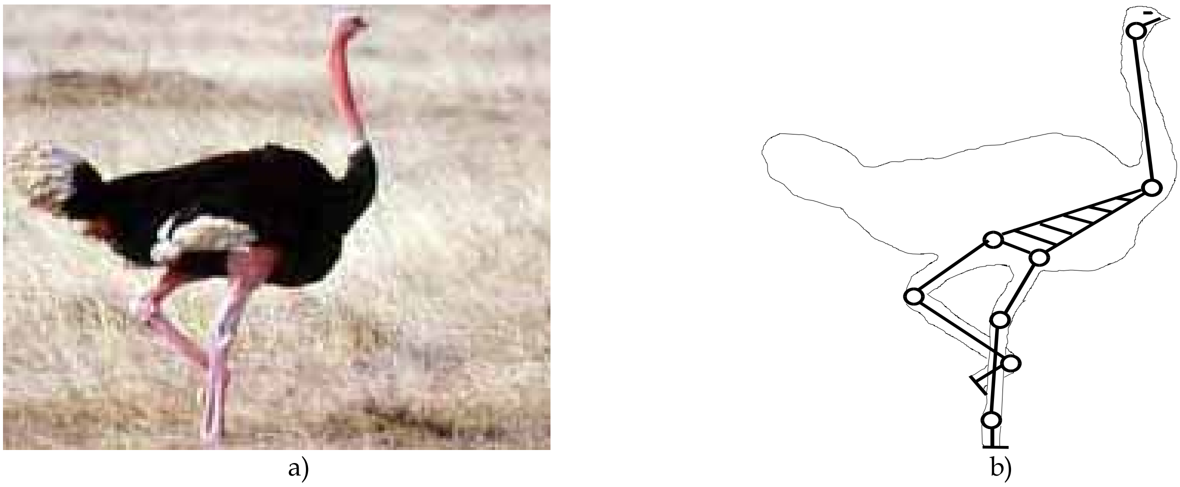
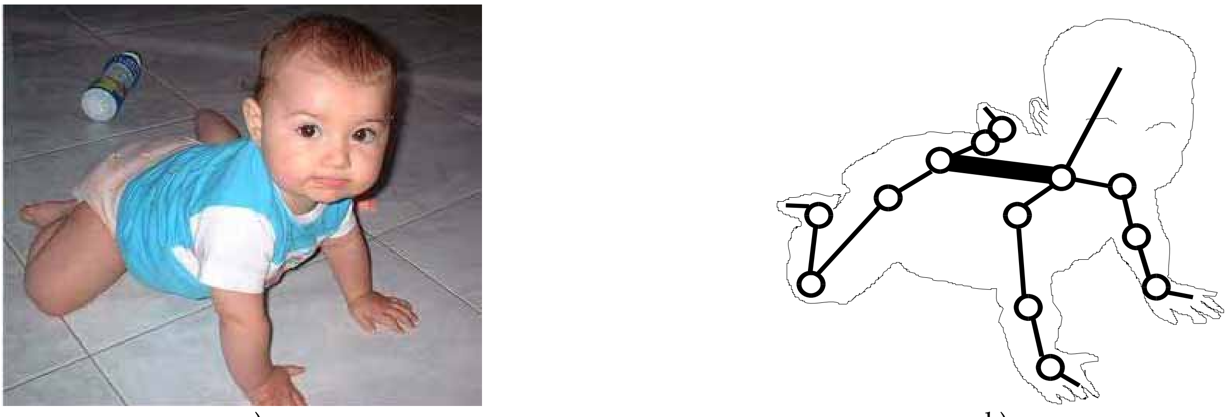


Figure 2. Biped locomotion by an ostrich: a) a picture; b) a kinematics scheme

This type of locomotion is used by human being at their first months of life since it is statically stable and requires less sensory feedback. It is worth noting that a young child can not achieve a very efficient quadruped motion since the human arms are not equipped with proper muscles. Thus, he starts to use a biped locomotion as soon as his body and brain are capable of keeping the equilibrium in dynamic conditions. More characteristics of a young child in terms of quadruped locomotion are reported in Table 1.

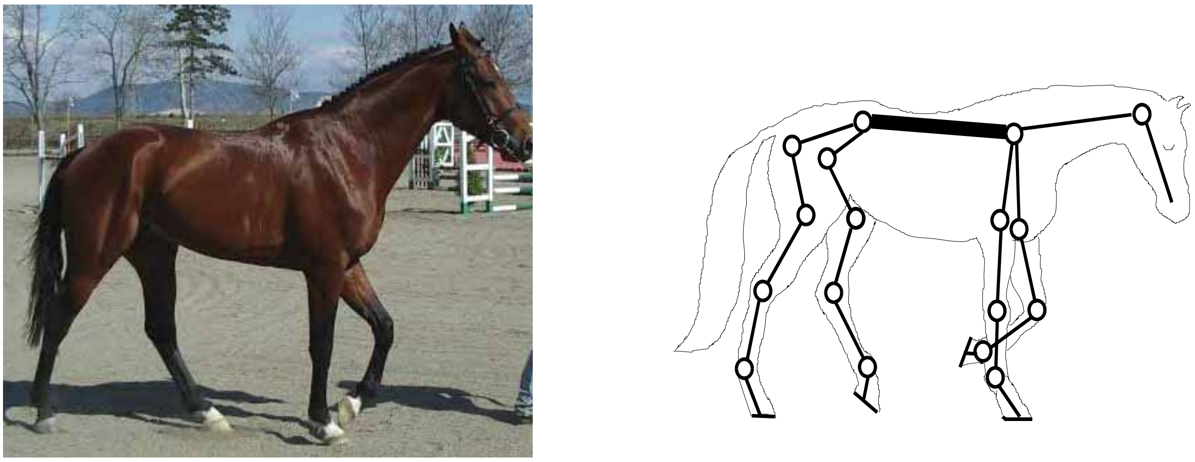
Figure 4a) and b) show a horse and a simplified kinematic scheme of its four limbs, respectively. In the case of horses, the quadruped locomotion can be considered very efficient. In fact, they can be considered among the fastest legged animals with a maximum speed of 21.1 m/s during a 800 meter run. Moreover, they show a good payload capacity and attitude to jumping. More details are reported in Table 1.

Figures 5 and 6 shows two examples of hexapod locomotion: a spider and a cockroach and a simplified kinematic scheme of their six limbs, respectively. The use of six limbs provide to these animals a very good ability of movements in rough terrain and also a surprisingly high payloads capability.



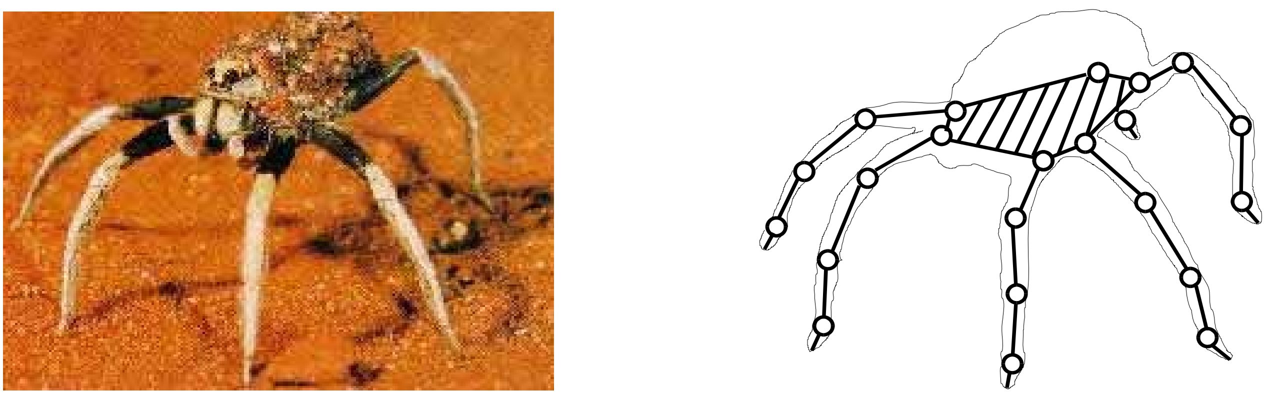
a) b)

Figure 3. Quadruped locomotion by a young child: a) a picture; b) a kinematic scheme



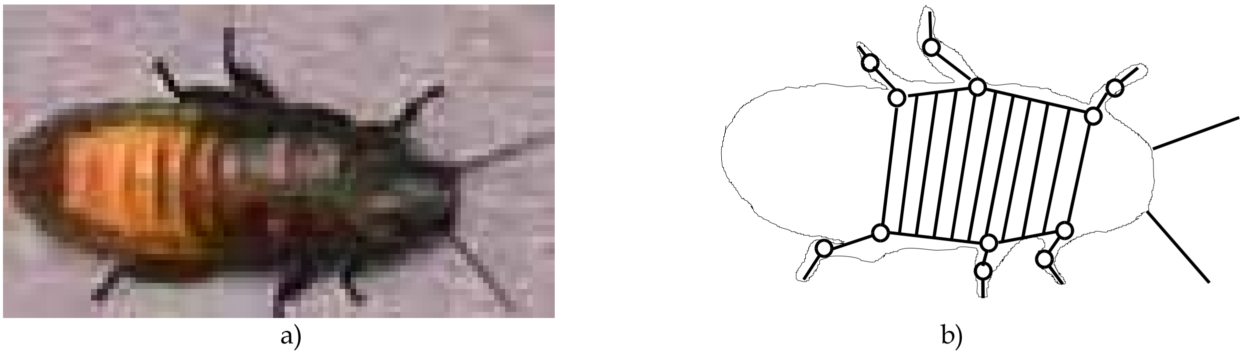
a) b)

Figure 4. Quadruped locomotion by a horse: a) a picture; b) a kinematic scheme



a) b)

Figure 5. Hexapod locomotion by a spider: a) a picture; b) a kinematic scheme



a) b)

Figure 6. Hexapod locomotion by a cockroach: a) a picture; b) a kinematic scheme

For example, a giant cockroach can move a weight of even 800 times its weight. Nevertheless, the hexapod locomotion is mainly used by animals having small size and weight. This is probably due to the complexity of distribution of muscles for all the limbs and also for the complexity of the control strategy for walking modes.

Name	Size [m]	Weight [N]	Max. Speed [m/s]	Step size [m]	Step height [m]	Payload [N]	Photo
Human being	1.7x0.5x0.3	650	11.0	0.6	0.3	500	Fig.1
Ostrich	2.5x1.6x0.8	1,800	19.4	0.6	0.6	500	Fig.2
Young child	0.4x0.2x0.2	100	0.5	0.1	0.05	5	Fig.3
Horse	1.6x2.0x0.6	4,500	21.1	0.6	0.6	1,000	Fig.4
Spider	0.1x0.1x0.1	1	1.5	0.03	0.03	100	Fig.5
cockroach	0.05x0.05x0.05	0.5	1.3	0.02	0.02	400	Fig.6

Table 1. Main characteristics of the animals shown in Figs.1 to 6. (Values, when not available, have been estimated from size, kinematic architecture, and mobility range)

3. Existing Walking Machines

The kinematic models of Fig.1 to 6 have inspired and still inspire the mechatronic design and operation for several biped, quadruped and hexapod walking machines.

In the recent past several walking machines have been developed for several different purposes mainly in research laboratories. Significant examples of walking machines are shown in Fig.7 to 13:

- Honda robot ASIMO, Fig.7a);
- Sony robot SDR-4X, , Fig.7b);
- Waseda robot WABIAN-RV, Fig.7c);
- Waseda biped locomotor WL-16R, Fig.8;
- CSIC robot RIMHO2, Fig.9a);
- Ambulatory Robotic Lab. robot Scout II, Fig.9b);
- Hirose & Yoneda Robotic Lab. robot TITAN VIII, Fig.10a);
- Intelligent Machines and Special Robotics Institute robot WorkPartner, Fig.10b);
- Chiba University COMET II, Fig.11a);
- Ambulatory Robotic Lab. robot Rhex, Fig.11b);
- Plustech Ltd. Walking Forest Machine, Fig.12a);
- Ohio State University Adaptive Suspension Vehicle, Fig.12b);
- Sony robot AIBO, Fig.13.

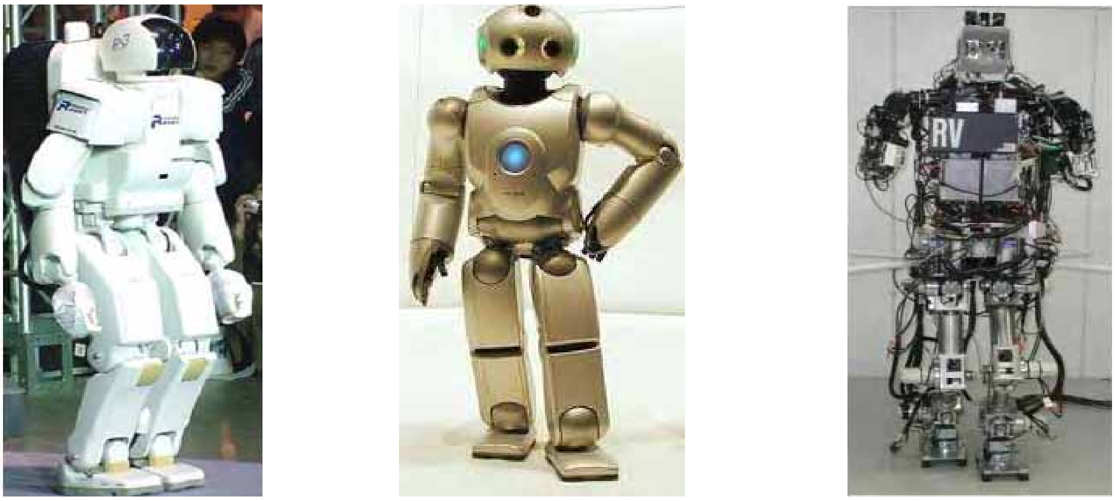
In Tab.2 main characteristics are reported with values that are indicative of design and operation performances. In the following, synthetic descriptions are discussed to outline basic problems and solutions that are available in the current state of walking machines.

ASIMO (Advanced Step in Innovative MObility), Fig.7a), has been built at Honda in the year 2000. It is a biped humanoid robot having a total of 26 degrees of freedom. Its size, weight and ranges of mobility have been conceived to mimic as much as possible a human child and move freely within the human living environment. ASIMO is able to ascend and descend stairs, to walk by following different patterns, to avoid obstacles, to grasp objects, to interact with humans by means of sound and image recognition. It is equipped with on board batteries for a continuous operating time of about 30 min.

SDR-4X (Sony Dream Robot version 4X), Fig.7b) has been built at Sony in the year 2002. It is a small biped humanoid robot that has been conceived for entertainment purposes. It has a total of 38 degrees of freedoms. It can walk on irregular (up to 10mm) and tilted surface (up to 10 degrees), and can prevent falling over when an external pressure is applied. Its software includes real time generator of walking patterns in order to adapt its behavior to various situations. Image and sound recognition features have been also implemented. It is equipped with on board batteries for a continuous operating time of 2 hour approximately.

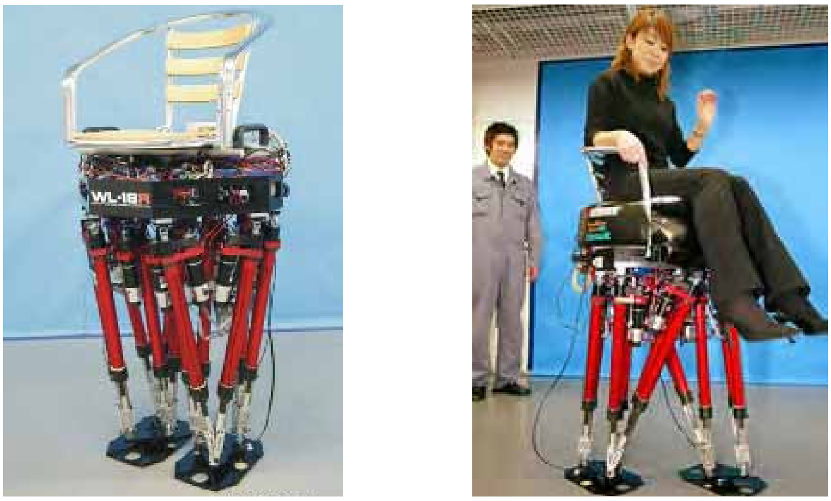
WABIAN-RV (Waseda Biped humANoid Refined V), Fig.7c), is a biped humanoid robot that has been developed for human-robot cooperation work at Waseda University, in Tokyo in the year 2002. It is the last version of WABIAN series started since 1972. It has a total of 43 dofs. The size and motion range of each link has been designed to be as human like as possible. By using WABIAN series, a variety of walking has been achieved such as dynamic forward and backward walking, marching in place, dancing, carrying a load, and emotional walking. Its software includes an on-line pattern generator, image and sound recognition features. It requires an external power supply.

WL-16R (Waseda-Leg No.16 Refined), Fig.8, has been designed at Waseda University in Tokyo in the year 2004 for practical use as a multi-purpose biped locomotor for robotic systems. It is composed of two 6-dof legs with parallel architecture design. It can be seen as a bipedal robot with only lower-limbs that can dynamically walk independently. Its upper body can be developed by users according to their use purposes.



a) b) c)

Figure 7. Examples of current biped walking machines: a) Honda robot ASIMO; b) Sony Robot SDR-4X, c) Waseda robot WABIAN-RV

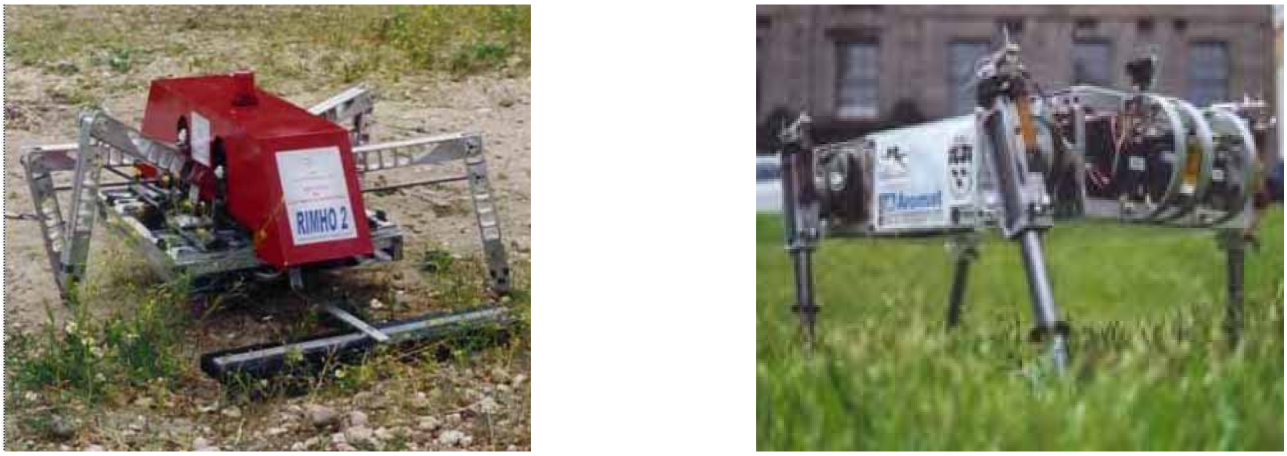


a) b)

Figure 8. Waseda biped locomotor WL-16R with non anthropomorphic legs: a) a side view; b) carrying a human

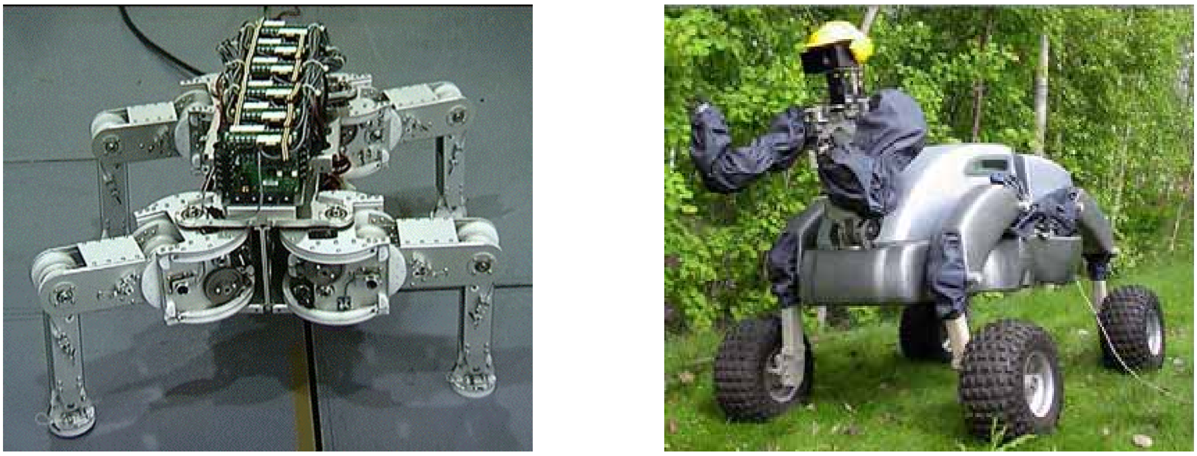
In particular, it would be applicable to the welfare field as a walking wheelchair or as a walking support machine that is able to walk up and down stairs carrying or assisting a human. It is equipped with an on board Nickel Metal Hydride battery for a continuous operating time of 1 hour approximately. The RIMHO II walking robot, Fig.9a), has been developed from the Industrial Automation Institute-CSIC and the CIEMAT in Madrid since 1993. It is a quadruped-walking machine of the insect type. Its four legs are based on a three dimensional Cartesian pantograph mechanism. The RIMHO walking robot can perform both discontinuous and wave gaits over irregular terrain including slopes and stairs, and has been tested also over natural terrain as shown in Fig.9a).

SCOUT II, Fig.9b) has been developed at Ambulatory Robotic Laboratory in Montreal since 1998. It is composed of four legs. Each leg has one active degree of freedom only. A spring and a passive knee are added in order to provide two additional passive degrees of freedom for each leg. These passive degrees of freedom make the Scout II capable of achieving dynamic running similar to gallop and trot. SCOUT II is fully autonomous having on board power, computing and sensing. Other features include an on board pan-tilt camera system and laser sensors.



a) b)

Figure 9. Examples of four legged walking machines: a) RIMHO2; b) Scout II



a) b)

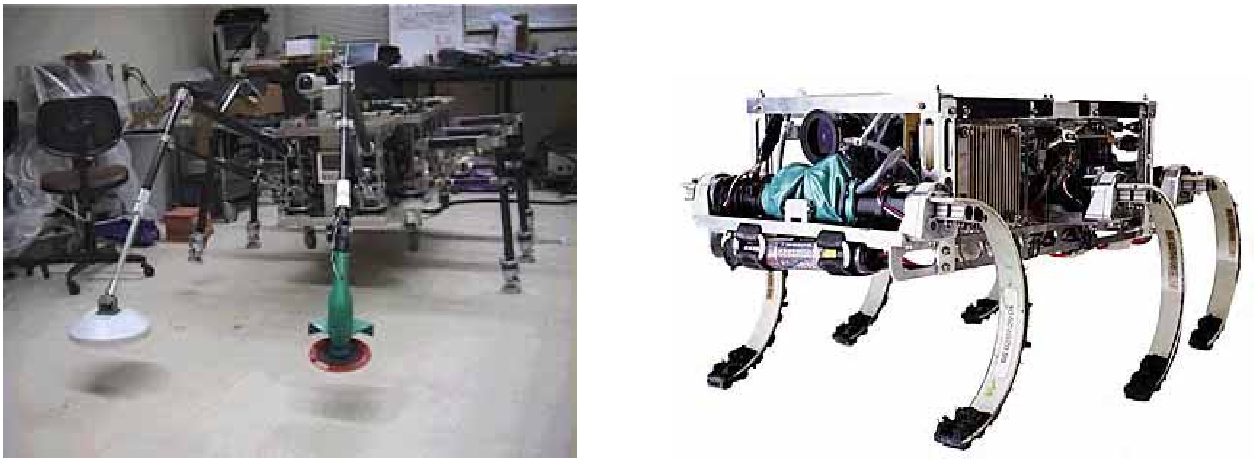
Figure 10. Examples of four legged walking machines with wheels: a) TITAN VIII; b) WorkPartner

TITAN VIII, Fig.10a) has been built at Hirose & Yoneda Robotic Lab. in Tokyo in the year 1996. TITAN VIII is a walking machine having four legs.

The leg mechanism is composed of a planar 2 degrees of freedom mechanism and a rotating mechanism which rotates this planar mechanism. So this leg mechanism has 3DOF. Wires and pulleys are used for the power transmission within the leg. The feet of TITAN VIII can be used also as wheels in order to achieve faster motion on flat surfaces.

WorkPartner, Fig.10b), is a four leg mobile robot that has been built at the Intelligent Machines and Special Robotics Institute in Helsinki in 2000. The locomotion system of WorkPartner is hybrid. In fact, it is possible to move by means of legs only, with legs and wheels powered at the same time or with wheels only. WorkPartner is equipped with two arms having 3 degrees of freedom arms and a two-degree of freedom camera head. Several sensors have been installed on board such as potentiometers, force sensors, inclinometers, gyro, accelerometers, ultra sonic sensors, laser scanner. A combustion engine and four batteries are also installed on board for a continuous operating time of 30 min. approximately.

COMET II has been developed at Chiba University in Tokyo in 2002. It can be used as fully autonomous system or teleoperated by a human for demining tasks. It is equipped with two manipulators that are used for mine detection and grass cutting.



a) b)

Figure 11. Examples of six legged walking machines: a) COMET II; b) RHex

Several sensors are installed on board such as metal detector, radar, cameras, force sensors, potenziometers.

The implemented software includes obstacle avoidance features. Power supply is provided by a gasoline power generator for outdoor operation or from an external power supply for indoor laboratory tests.

RHEx robot, Fig.11b), has been developed at Ambulatory Robotic Lab. in Montreal since 1999. It has six legs with only one degree of freedom. The leg has a very simple design. It is made by heat shaped Delrin rods and it has a soft feet at its free end. RHEx robot is equipped with gyros accelerometers, and optical encoders. It is capable of achieving a wide variety of dynamically dextrous tasks, such as walking, running, leaping over obstacles, climbing stairs. Two batteries are on board installed for a continuous operating time of about 10 min.

The Walking Forest Machine, Fig.12a), has been developed at Plustech Ltd. since 1995 for outdoor forest harvesting tasks. It is composed of six articulated legs. It can move forward, backward, sideways and diagonally. It can also turn in place and step over obstacles. Depending on the irregularity of the terrain, the operator can adjust both the ground clearance of the machine and height of each step. The operator-friendly controls are incorporated in a joystick that controls direction of movement, traveling speed, step height and gait, and the ground clearance.

Adaptive Suspension Vehicle, Fig.12b), has been developed at Ohio State University since early 80's. It is composed of six articulated legs. Each leg has three active degrees of freedom. It has been designed for walking on rough terrains by carrying a maximum load of 2156 N. It is equipped with gyros, laser sensors and a computer vision system that is used for adapting the gait to the environment. It can work either in teleoperated or operator-on-board mode by using active compliance control algorithms.

AIBO robot is a four legged robot for entertainment purposes. It has been developed by Sony since early 90's. It has evolved though several prototypes with different performance and shape. Figure 13 shows a version that is called ERS7 and is avaliable on the market since 2003 at a price of about 2000 Euros. This version has four legs with three degrees of freedom each. It is equipped with two microphones, a speaker, touch sensors, infrared sensors, accelerometers that provide AIBO with very high human-robot interaction capabilities. Rechargable batteries are on board installed allowing for a continuous operating time of 1.5 hours, approximately.

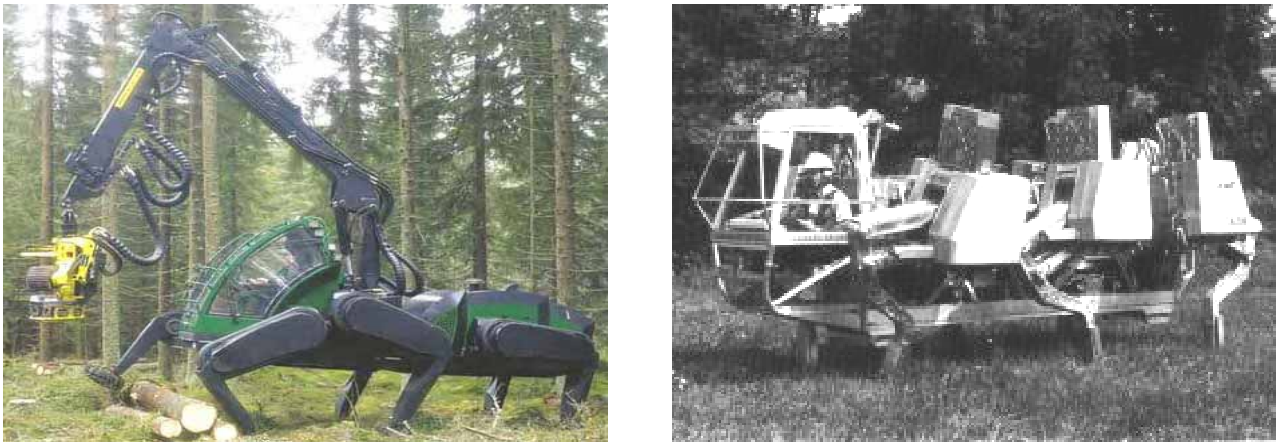

a) b)

Figure 12. Examples of six legged walking machines for outdoor applications: a) Walking forest machine; b) Adaptive Suspension Vehicle.

The reported examples in Figs.7 to 13 give an illustrative view of the variety of walking systems that have been developed all around the world with different solutions for different applications. It is worth noting that most of them (with the exception of AIBO) are not yet available in the market but they are under further development in Research Labs.

Name	Size [m]	Weight [N]	Max. Speed [m/s]	Step size [m]	Step height [m]	Payload [N]	Photo
ASIMO	1.2 x 0.4 x 0.5	520	0.4	0.3	0.1	5	Fig.7a)
SDR-4X	0.6 x 0.3 x 0.2	70	0.3	0.1	0.02	0	Fig.7b)
WABIAN-RV	1.8 x 0.4 x 0.6	1,300	0.3	0.3	0.1	10	Fig.7c)
WL-16	1.3 x 0.6 x 0.6	620	0.3	0.3	0.1	600	Fig.8
RIMHO2	0.7 x 0.7 x 0.3	650	0.1	0.3	0.1	0	Fig.9a)
Scout II	0.4 x 0.4 x 0.4	270	1.5	0.1	0.1	5	Fig.9b)
TITAN VIII	0.6 x 0.3 x 0.4	190	0.2	0.2	0.2	70	Fig.10a)
WorkPartner	1.4 x 1.2 x 1.2	2,300	1.94	0.5	0.3	100	Fig.10b)
COMET II	1.4 x 0.6 x 0.6	1,000	0.1	0.2	0.3	200	Fig.11a)
RHex	0.5 x 0.2 x 0.1	70	0.4	0.1	0.2	0	Fig.11b)
A.S.V.	5.2 x 2.4 x 3.0	32,000	2.3	0.5	0.3	2156	Fig.12a)
W.F.M.	7.4 x 2.7 x 3.7	34,000	2.0	0.5	0.4	2000	Fig.12b)
AIBO	0.2 x 0.3 x 0.3	16.5	0.3	0.1	0.1	0	Fig.13

Table 2. Main characteristics of walking machine prototypes of Figs.7-13. (Values, when not available, have been estimated from size, mechanical design, and mobility range)

Figure 13. The low-cost four legged robot AIBO

Generally, legged systems can be slow and more difficult to design and operate with respect to machines that are equipped with crawlers or wheels. But, legged robots are more suitable for rough terrain, where obstacles of any size can appear. In fact, the use of wheels or crawlers limits the size of the obstacle that can be climbed, to half the diameter of the wheels. On the contrary, legged machines can overcome obstacles that are comparable with the size of the machine leg. Therefore, hybrid solutions that have legs and wheels at the same time have been also developed as shown for example in Fig.8c) and d). This type of walking machines may range from wheeled devices to true walking machines with a set of wheels. In the first case, the suspensions are arms working like legs to overcome particularly difficult obstacles. In the second case wheels are used to enhance the speed when moving on flat terrain.

The advantage of the legged systems over the wheeled systems can be understood by looking at their kinematic capability and static performance. They can be deduced by the schemes of Fig.14 and 15 for legged and wheeled systems, respectively. In Fig.14 a biped system is represented by taking into account its weight P, the weight PL of a leg, the forward acceleration a, the reaction force R at the ground, and the actuating torque CL for a leg. The geometry of the system is modeled trough the distances shown in Fig.14 among which dL represents the step capability and h is for the step height. The walking capability is given by the size of the distance dL that avoids falling of the system, together with motion capability for each leg that is given by the mobility range and actuating torque.

In particular, looking at the instantaneous equilibrium gives the computation of necessary conditions for a stable walking and the evaluation of the maximum step height. In the sagittal plane, the equilibrium can be expressed by

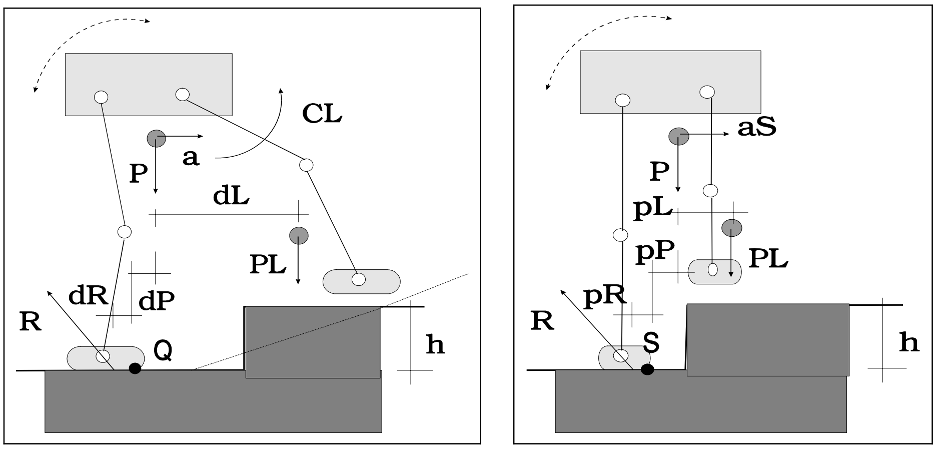
$$R_h - P\frac{a}{g} - PL\frac{a_L}{g} \geq 0;\;\; R_v - P - PL \geq 0;\;\; R_v dR - PdP - PL(dP + dL) - C_{inS} \geq 0 \qquad (1)$$

where R_h and R_v are the horizontal and vertical components of R; C_{inS} is the sagittal component of the inertial torque due to waist balancing movement.

The point Q is assumed as the foot contact point about which the system will rotate in the possible fall. In the front plane the equilibrium can be expressed by

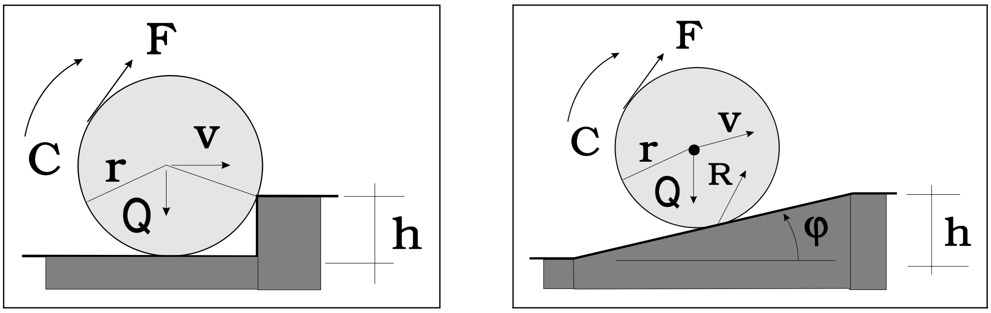
$$R_l - P\frac{aS}{g} \geq 0;\;\; R_v pR - PpP - PL(pP + pL) - C_{inl} \geq 0 \qquad (2)$$

where R_l is the lateral component of R; C_{inl} is the lateral component of the inertial torque of waist balancing movement.



a) b)

Figure 14. A scheme for performance evaluation of biped walking machines: a) sagittal view; b) front view



a) b)

Figure 15. A scheme for performance evaluation of wheeled machines

The point S is assumed as the foot contact point about which the system will rotate in the possible fall. The components R_h and R_f refer to friction actions at the foot contact area. By using Eqs.(1) and (2) is possible to compute conditions for design and operation in an environment with obstacles of height h.

From geometric viewpoint the obstacle/step of height h can be overpassed when the leg mobility gives

$$l1(1-\cos\phi_1)+l2(1-\cos\phi_2)>h \tag{3}$$

in which l1 and l2 are the lenghts of leg links, whose angles ϕ_1 and ϕ_2 are measured with respect to a vertical line.

Similarly, for a wheeled system the instantaneous equilibrium can be expressed as referring to Fig.15 for the case a), in the form

$$Q-Rv-N=0;\quad R_h-T=0;\quad C(=Fr)=f_v(Q+R) \tag{4}$$

and for the case b) in the form

$$Q\cos\varphi=N\quad Q\sin\varphi=T;\quad C=f_v rQ\cos\varphi \tag{5}$$

with the condition for pure rolling (without sliding)

$$T<f\ N \tag{6}$$

where f is the sliding friction coefficient and f_v (<<f) is the rolling friction coefficient; R is the reaction force at the contact point in the step wedge; with its horizontal and vertical components R_h and R_v; Q is the load and weight on the wheel axis; C is the actuating torque due to force F to maintain forward velocity v. The geometrical limits for overpassing an obstacle-step of height h through rolling a wheel can be expressed by

$$r>h \tag{7}$$

when the actuating torque acts only. Alternatively, a force will push upward the wheel or, as commonly used, the step is reshaped as an inclined plane, as shown in Fig.15b).

An intense activity is carried out not only for designing and prototyping walking machines but also for debating and exchanging information and experiences. The first activity is carried out in Research laboratories in Universities but also in Research Centres of governmental Institutions or Company Divisions. Very recently, some prototypes have been even commercialized and they are available in the market, like for example, AIBO robot for leisure/company applications or SDR-4X Sony robot for adverticing purposes. The success of these preliminary attempts of practical uses of walking machines stimulate more and more the development of systems for a variety of potential users.

Similarly entusiasmatic activity is that one with circulation and publication of results and information both on research activity and practical applications. This activity is mainly concentrated in forum like congress events or journal publications that can be considered also source for further reading and continuous updating of the State-of Art. The topic of walking machines is discussed in specific Conferences and Journals, but also as important section in more general events.

The following list is not exhaustive (also because new initiatives are continuously started) but it gives main sources and size of the publication activity on walking machines.

Conference Event Series:

International Conference on Climbing and Walking Robots (CLAWAR)

IEEE-RAS International Conference on Humanoid Robots (HUMANOIDS)
International Conference of Intelligent Autonomous Systems (IAS)
IEEE International Conference on Systems, Man and Cybernetics (SMC)
CISM-IFToMM Symposium on Robot Design, Dynamics and Control (ROMANSY)
IEEE Mediterranean Conference on Control and Automation (MED);
International Conference on Advanced Robotics (ICAR)
IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)
World Congress in Mechanism and Machine Science (IFToMM)
IEEE International Conference on Robotics and Automation (ICRA)
IEEE-RSJ International Conference on Intelligent Robots and Systems (IROS)
ASME International Conference on Mechanisms and Robotics (MECH)
International Symposium on Multibody Systems and Mechatronics (MUSME)
International Workshop on Robotics in Alpe-Andria-Danube Region (RAAD)
ECCOMAS International Conference on Advances in Computational Multi-body Dynamics (Multibody Dynamics)
Technical – Scientific Journals:
International Journal of Humanoid Robotics (IJHR)
International Journal of Advanced Robotic Systems
Mechanism and Machine Theory
ASME Journal of Mechanical Design
International Journal Advanced Robotics
International Journal of Robotics Research
International Journal Robotica
Fuji International Journal of Robotics and Mechatronics
IEEE Transaction on Robotics

4. Gait Analysis and Design Problems

A gait is a pattern of locomotion characteristic of a limited range of speeds, described by quantities of which one or more change discontinuously at transitions to other motion patterns. A duty factor can be defined as the fraction of the duration of the stride, for which each foot is on the ground. In walking, each foot is on the ground for more than half the time, and in running for less than half the time. As speed increases, the duty factor falls gradually from about 0.65 in slow walks to about 0.55 in the fastest walks; but at the change to running it drops to around 0.35. Also the forces that are exerted on the ground can significantly change as speed increases and during the transition from a locomotion pattern to another.
There are several types of gaits in nature that are suitable for walking machines. Feasible gaits can be considered the following:

- human-like walking behavior;
- horse-like tolt behavior;
- horse-like trot behavior;
- horse-like pacing behavior;
- horse-like canter behavior;
- horse-like gallop behavior;
- crab-like walking behavior.

In the human-like walking behavior each foot leaves the ground at a different time. This type of gait can be achieved with two or more legs. Figure 16 shows the movements of limbs in a four leg walking also with the footfall formula representation. In footfall formula representation the limbs that are in contact with the ground surface are shown as black circles in a table in which the entries represent the possible foot contacts with the ground.

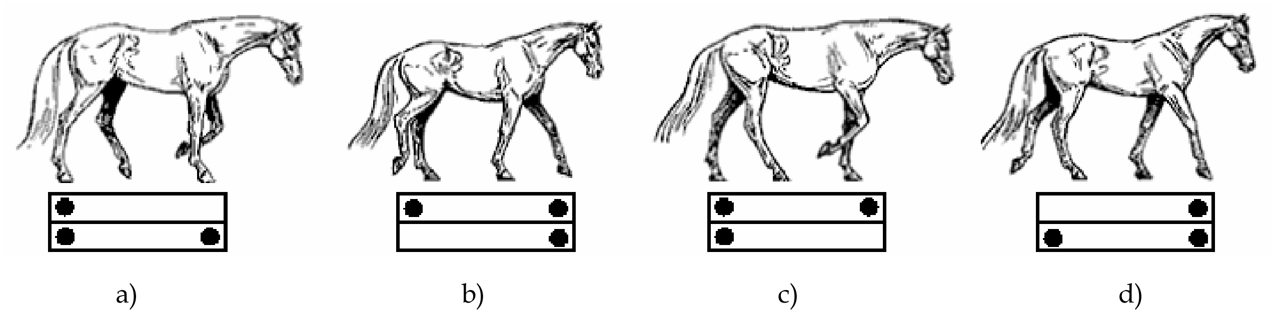


Figure 16. Movements of limbs in a four legs walking with footfall formulas representation (black circles stands for the limbs in contact with the ground surface): a) first beat; b) second beat; c) third beat; d) fourth beat

A similar behavior is achieved in the horse-like tolt behavior, which is a running walk used when covering broken ground. This gait can be achieved with four legs.

In the horse-like trot behavior of Fig.17 the legs move in diagonal pairs, with a moment of suspension between each stride.

This type of gait can be achieved with four or more legs. Figure 17 shows the movements of limbs in a four legs trot also with the footfall formulas representation.

In the horse-like pacing behavior the legs of the same side move simultaneously. Thus, one has the two left legs or the two right legs in contact with the ground while the other two legs move forward simultaneously.

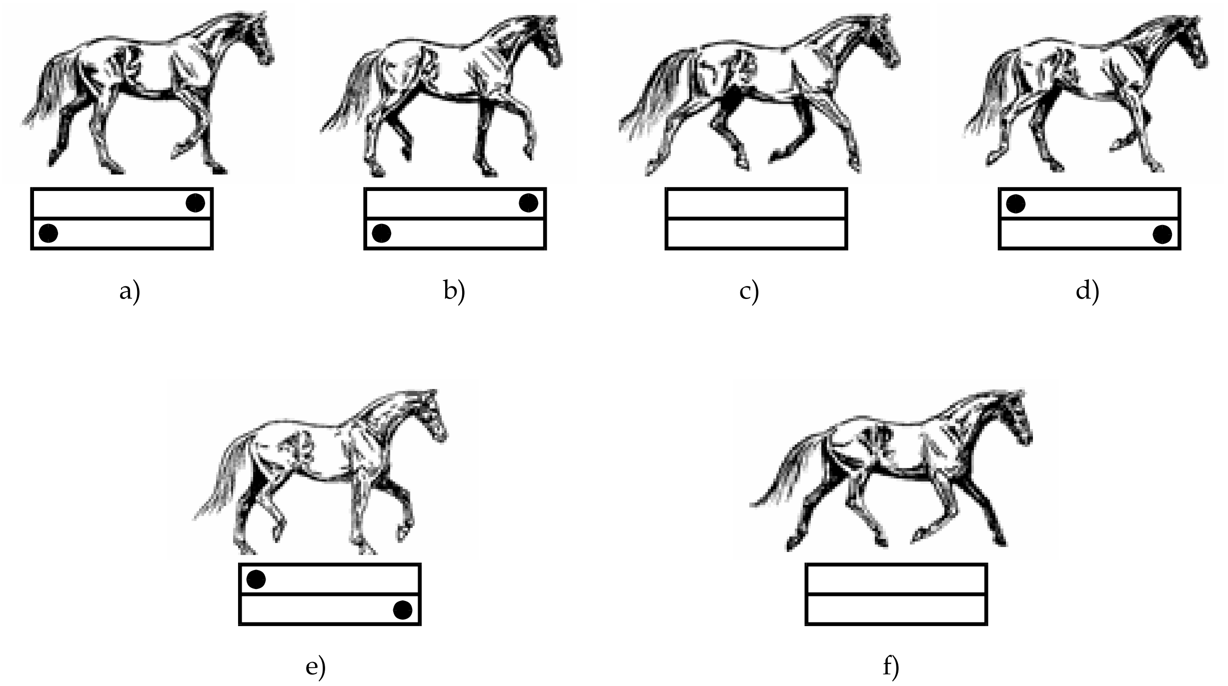
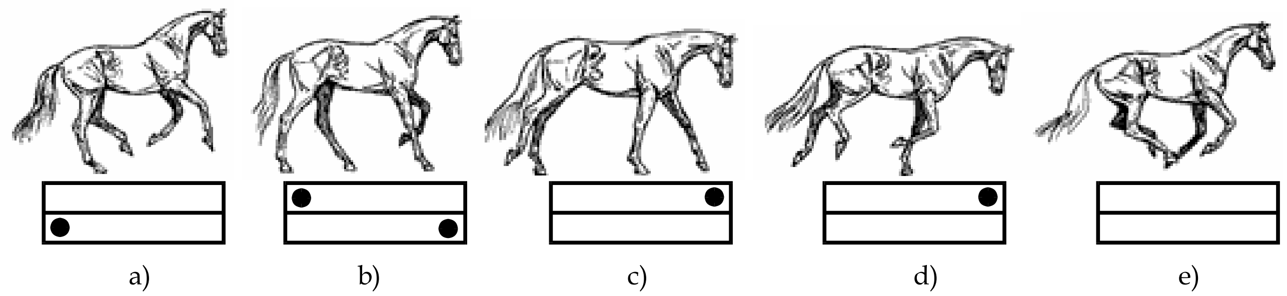


Figure 17. Movements of limbs in a four legs trot with footfall formulas representation (black circles stands for the limbs in contact with the ground surface): a) first beat; b) first support phase; c) first suspension phase (all feet are off the ground); d) second beat; e) second support phase; f) second suspension phase



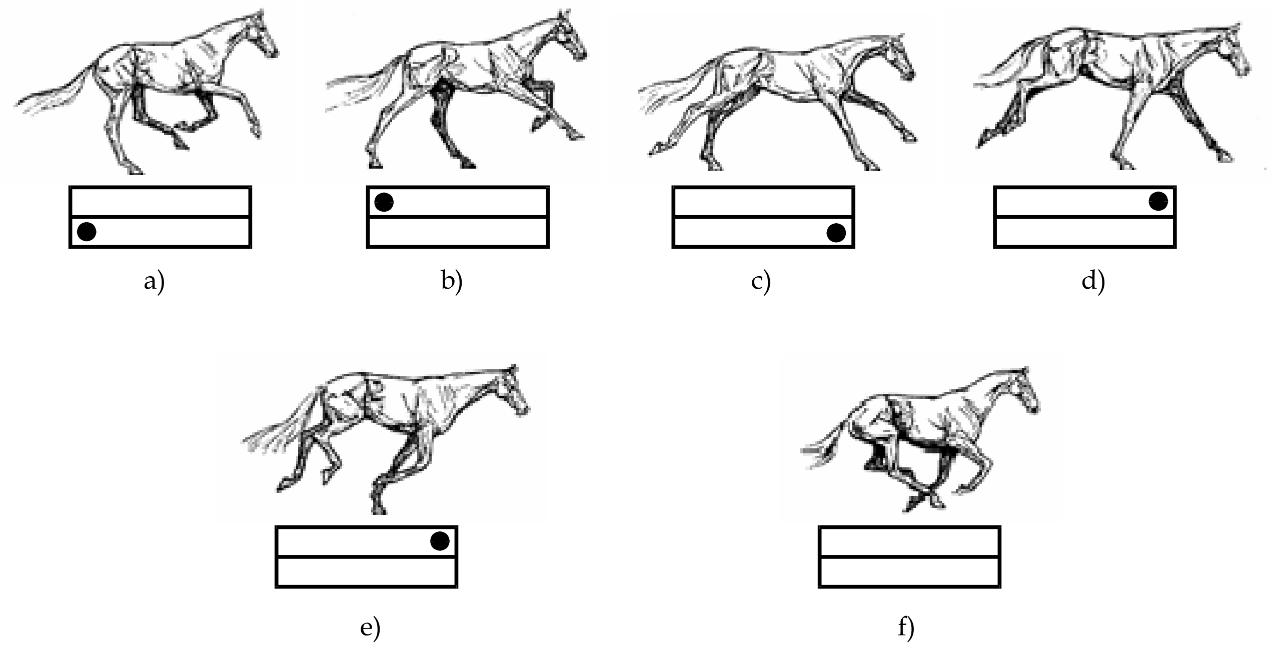
a) b) c) d) e)

Figure 18. Movements of limbs in a four legs canter with footfall formulas representation (black circles stands for the limbs in contact with the ground surface): a) first beat; b) second beat; c) third beat; d) fourth beat e) suspension phase (all feet are off the ground)

In the horse-like canter behavior in Fig.18 the legs move in the following 4 phases: an hind leg, the other hind leg together with the diagonal fore leg, the other fore leg, at this time there is a period where all four feet are off the ground. This type of gait can be achieved with four or more legs. Figure 18 shows the movements of limbs in a four legs canter also with the footfall formulas representation. In the horse-like gallop behavior in Fig.19 the legs move in the following 4 phases: an hind leg, the other hind leg, the diagonal fore leg, the other fore leg, at this time there is a period where all four feet are off the ground. This type of gait can be achieved with four or more legs. Figure 19 shows the movements of limbs in a four legs trot also with the footfall formulas representation.

There are also other horse-like gaits in which there are lateral movements. Crab-like walking behavior are similar to the human-like behavior but it is possible to have a lateral walking instead of forward or backward.

Increasing the number of limbs increases the number of feasible gaits, increases the flexibility of the motion but at the same time increases the complexity of the control for the coordination of the movements of each limb.



a) b) c) d)

e) f)

Figure 19. Movements of limbs in a four legs gallop with footfall formulas representation (black circles stands for the limbs in contact with the ground surface): a) first beat; b) second beat; c) third beat; d) fourth beat; e) preparing for the suspension phase; f) suspension phase (all feet are off the ground)

Also the choice of the most convenient gait and the transition from a gait to another can be a difficult task for systems having a high number of limbs. These aspects are crucial issues when one try to mimic animal or insect like gaits.

5. Low-Cost Designs at LARM in Cassino

A challenge for a practical use of walking machines both in industrial and non-industrial environments can be recognized in the development of design solutions and operation modes with low-cost easy-operation features. This is the approach that has been and still is used at LARM to develop projects, experiences, and teaching on walking machines. In this section those aspects are illustrated to show the feasibility of carrying out activity on walking machines at any level of expertise and fund compatibility.

At LARM activity on walking machines has been addressed to biped robots and modular leg designs with low-cost components and easy-operation via PLC programming. Three prototypes are illustrated: EP-WAR, 1dof leg, and leg module.

Each leg mechanism of EP-WAR, shown in Fig.20, is composed by a pantograph and a double articulated parallelogram. The pantograph has the fundamental function to transmit the trajectory of foot point C to the actuation point H and reduce the movement size. The pantograph proportions give same kinematic proprieties between the points H and C of Fig. 20 b), with a scaling factor equal to 4. The parallelogram mechanism ensures a pure translation of the foot in a vertical plane. A complex control for legs co-ordination and dynamic stability has been avoided by using suction-cups beneath the feet to obtain static walking. Thus, EP-WAR can follow general polygonal trajectory in the plane of the motion because of a suitable mechanical design of each ankle joint with an axial ball bearing and a pneumatic rotation actuator.

In order to mimic the human gait, a suitable path is required for C foot point. The foot path is imposed through the actuation point H by using the scale factor 4 of the pantograph mechanism.

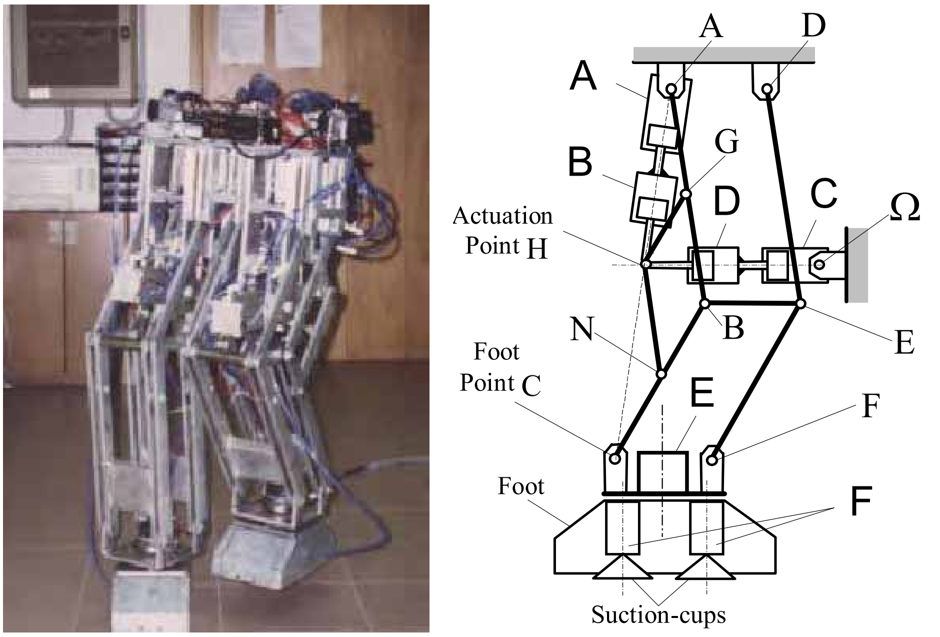


Figure 20. EPWAR (ElectroPneumatic WAlking Robot) built at LARM in Cassino: a) a prototype; b) a diagram for the leg design

A linear actuator A is connected to the body frame and actuation point H through revolute joints so that it is parallel to the terrain when the foot is at the start or end position.

Other two linear actuators B have been installed and connected both to the robot body and actuation point H through revolute joints. The operation of the proposed actuation system for each leg mechanism determines a trajectory of the actuation point H and consequently of the C foot point. The trajectory of EP-WAR foot can be even different to egg-shaped path of the human foot.

The trajectory of H gives a path of the foot, which is characterized by a forward displacement of half step p/2 length and a foot rise of height h. In addition the imposed trajectory permits an easy control of the actuation system in an On/Off environment via PLC. A controlled operation can be obtained through four phases: the first one corresponds to the in stroke movement of actuators B; the second one is related to the in stroke movement of the actuator A; the third one gives the outstroke movement of B; and finally the fourth phase corresponds to the outstroke movement of A.

The concept of elementary actions has been used to obtain suitable modules and subroutine programs, which can be easily combined to give any trajectory of EP-WAR. An elementary action can be defined as the smallest operation which may correspond to the simplest actuating action requiring one or few programming instructions. Additionally, it has been thought convenient and suitable to model any trajectory of a walking of EP-WAR with straight or turn paths since these kinds of motion can be easily performed by the prototype. Three walking modules and subroutine programs have been developed for straight line, right turn and left turn trajectories, respectively. They refer to a start position of EP-WAR with the right foot (a) ahead and on the ground, while the left foot (b) is free in back position. The turn walking module can be analyzed and specific sequences are reported in Figs. 21 and 22 for right turn trajectory. Referring to Fig. 21 with right foot in position (a), the first elementary action for EP-WAR is obtained by a forward and up motion of the leg from (b) to (c).

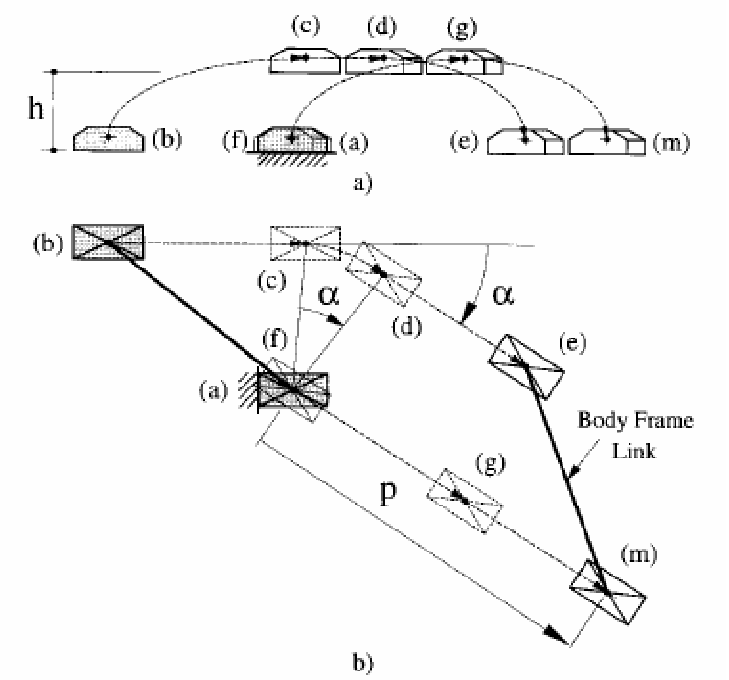


Figure 21. A walking analysis for a module of right turn trajectory of EP-WAR by using elementary actions: a) Sagittal view; b) Top view

The second action is performed for right turn of an angle a about vertical axis across the right foot.

Thus the left foot turns from (c) to (d) and then it is moved to the ground in (e). Successively the suction-cups of the left foot that is in (e) are operated and those of the right foot that is in (a) are switched-off so that the ground contact is moved from one foot to the other. Then the right foot turns from (a) to (f) and becomes parallel to the left foot. Successively it moves forward and up from (f) to (g) and finally to (m). This right turn module ends when the suction-cups of the right foot in (m) are operated again and those of the left foot in (e) are switched-off. Thus the EP-WAR reaches the start position again and a next walking module can be performed. The corresponding diagrams for a suitable flexible programming in a sub-routine for PLC is reported in Fig.22.

The length L of a general straight path can be given by a q value so that

$$L = q * p \tag{8}$$

and the subroutine represented by Fig.7b) is repeated until the counter variable q is equal to q*.

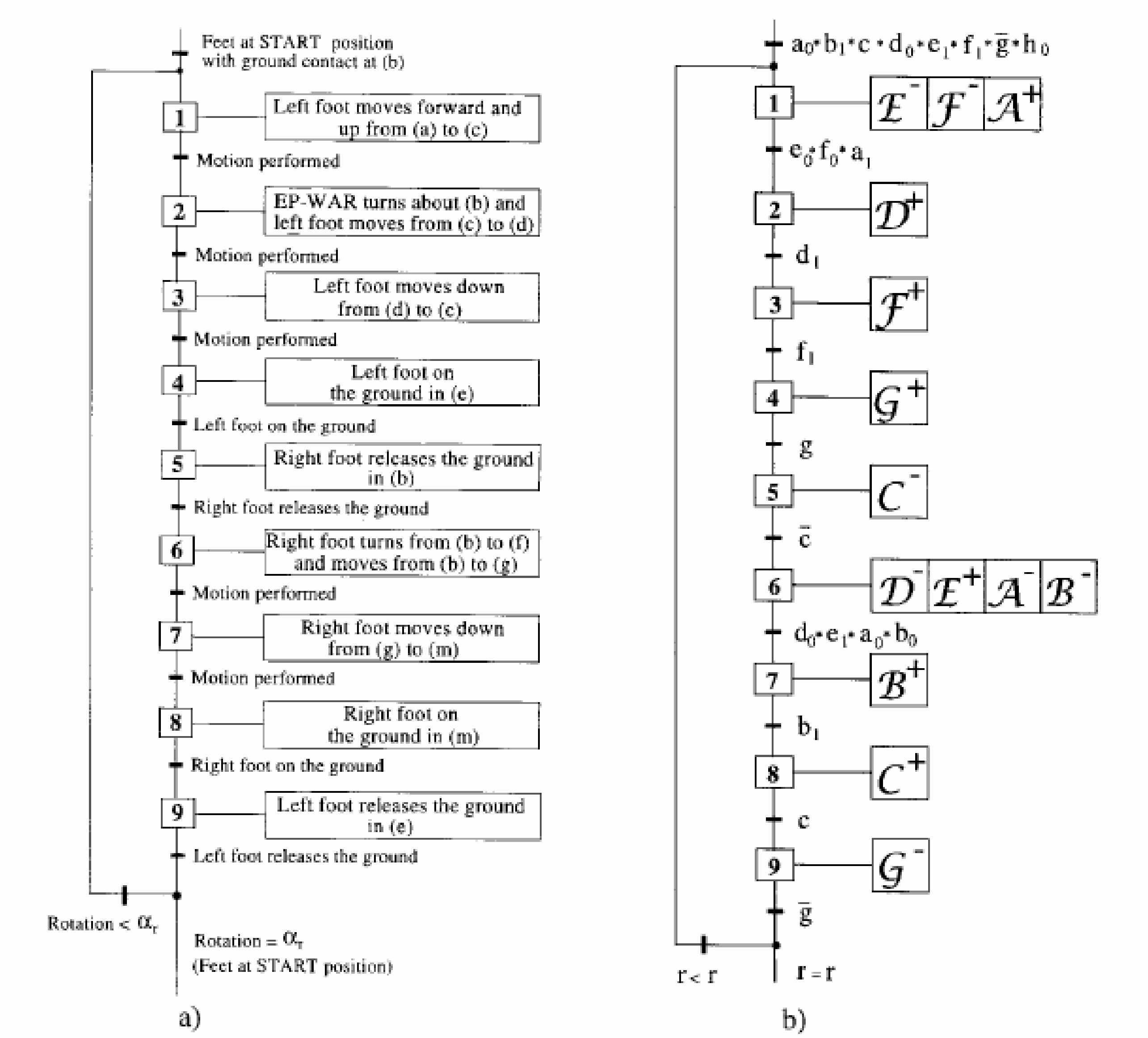

Feet at START position with ground contact at (b)
1
Left foot moves forward and up from (a) to (c)
Motion performed
2
EP-WAR turns about (b) and left foot moves from (c) to (d)
Motion performed
3
Left foot moves down from (d) to (e)
Motion performed
4
Left foot on the ground in (e)
Left foot on the ground
5
Right foot releases the ground in (b)
Right foot releases the ground
6
Right foot turns from (b) to (f) and moves from (b) to (g)
Motion performed
7
Right foot moves down from (g) to (m)
Motion performed
8
Right foot on the ground in (m)
Right foot on the ground
9
Left foot releases the ground in (e)
Left foot releases the ground
Rotation < α_r
Rotation = α_r
(Feet at START position)
a)
a_0·b_1·c·d_0·e_1·f_1·ḡ·h_0
1
E⁻ F⁻ A⁺
e_0·f_0·a_1
2
D⁺
d_1
3
F⁺
f_1
4
G⁺
g
5
C⁻
c̄
6
D⁻ E⁺ A⁻ B⁻
d_0·e_1·a_0·b_0
7
B⁺
b_1
8
C⁺
c
9
G⁻
ḡ
r< r
r = r
b)

Figure 22. Flowcharts for easy programming of the right turn walking module of Fig. 21: a) Sequences of the elementary actions; b) Grafcet diagram

Similarly, the turning displacement of the walking robot can be performed to the right or to the left by using the corresponding Grafcet diagrams to be repeated r* and s* times. In order to obtain a right turn angle equal to

$$\alpha_r = r * \alpha \tag{9}$$

or similarly for a left turn angle with s*, where α is the module angular turn corresponding to a step displacement. The modular angular turn is determined by the rotation capability of the rotative cylinders so that the radius R of the turn is fixed and equal to

$$R = \frac{p}{2\sin\frac{\alpha}{2}} \tag{10}$$

Finally, a general programming for a generic trajectory can be easily obtained by using the abovementioned sub-routines and a user will only assembly a suitable number of each sub-routines to achieve a desired trajectory of the walking robot. The control of the walking robot in a digital environment has been obtained by using a commercial PLC. The central unit can be connected as a remote terminal of the PLC with a common Personal Computer through a serial port RS-232 for an off-line programming, even for updating the sub-routines when additional features are provided to EPWAR.

At LARM the abovementioned approaches has been extended to design new leg systems with better features both in term of low-cost properties and easy operation programming.

Basic considerations for a low-cost leg design can be outlined as follows: the leg should generate an approximately straight-line trajectory for the foot with respect to the body; the leg should have an easy mechanical design; if it is specifically required it should posses the minimum number of DOFs to ensure the motion capability. Among many different structures, at LARM the so-called Chebyshev-pantograph leg has been developed. The proposed leg mechanism is shown in Fig.23. Its mechanical design is based on the use of a Chebyshev four-bar-linkage, a five-bar linkage, and a pantograph mechanism. For such a mechanism, the leg motion can be performed by using 1 actuator only. The leg has been designed by considering compactness, modularity, light weight, reduced number of DOF as basic objectives to achieve the walking operation.

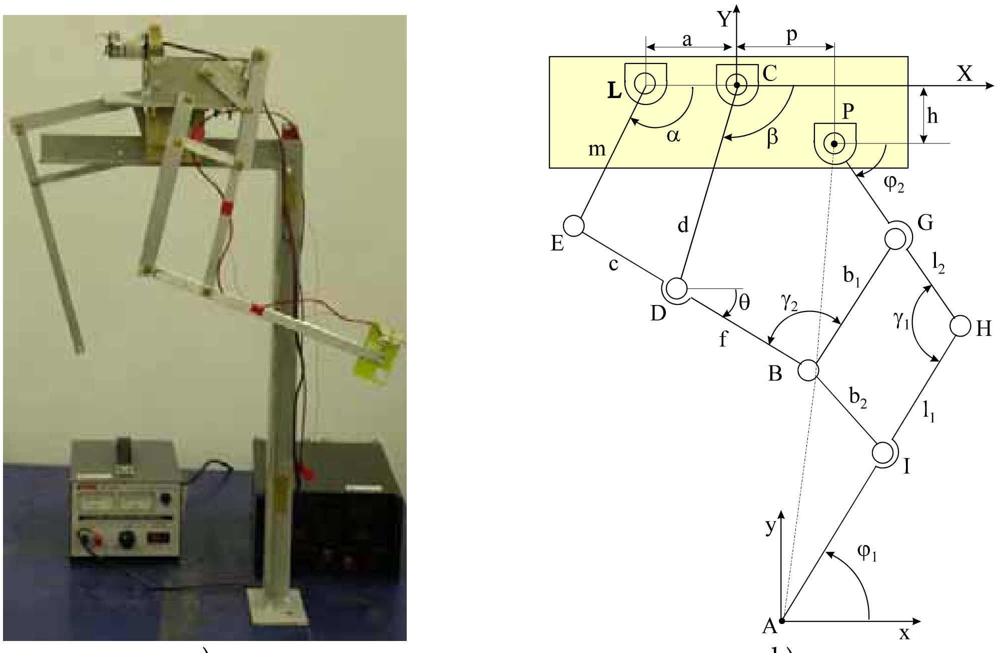
Numerical and experimental results show that good kinematic features can be obtained when points C and P in Fig. 23 are not coincident. The main characteristic of the proposed leg design consists in a fully-rotative actuation at point L to obtain the suitable trajectory of point B with one motor only that run continuously without any speed regulation. Furthermore, the trajectory of point B, and consequently, point A can be suitably modified by changing the design parameters shown in Fig. 23b). In particular, better features can be obtained if the transmission angles $\gamma_?$and γ_2 have suitable values.

Dimension of the leg prototypes are 400 mm high, 40 mmx250 mm so that they have a maximum lift of 80 mm and the step is of 470 mm.

In Figs. 24 and 25 the basic operation features of the Chebyshev-pantograph leg mechanism are reported through simulation results to show the feasibility of the low-cost easy operation design that have been experienced successfully by using a commercial DC motor without motion control equipment.

At LARM a so-called modular anthropomorphic leg has been obtained by defining a single link module that can be easily connected with other modules and can have inside all the needed actuators, transmissions and sensors. Figure 26 shows the proposed design for a single link module by using conic gears or timing belt transmissions. The main components of a single link module are:

- the body of the module;
- a dc motor with reduction gear train;
- two conic gears or a timing belt transmission;
- two mechanical switches.



a) b)

Figure 23. The so-called Chebyshev-pantograph leg developed at LARM: a) a first prototype; b) a kinematic diagram

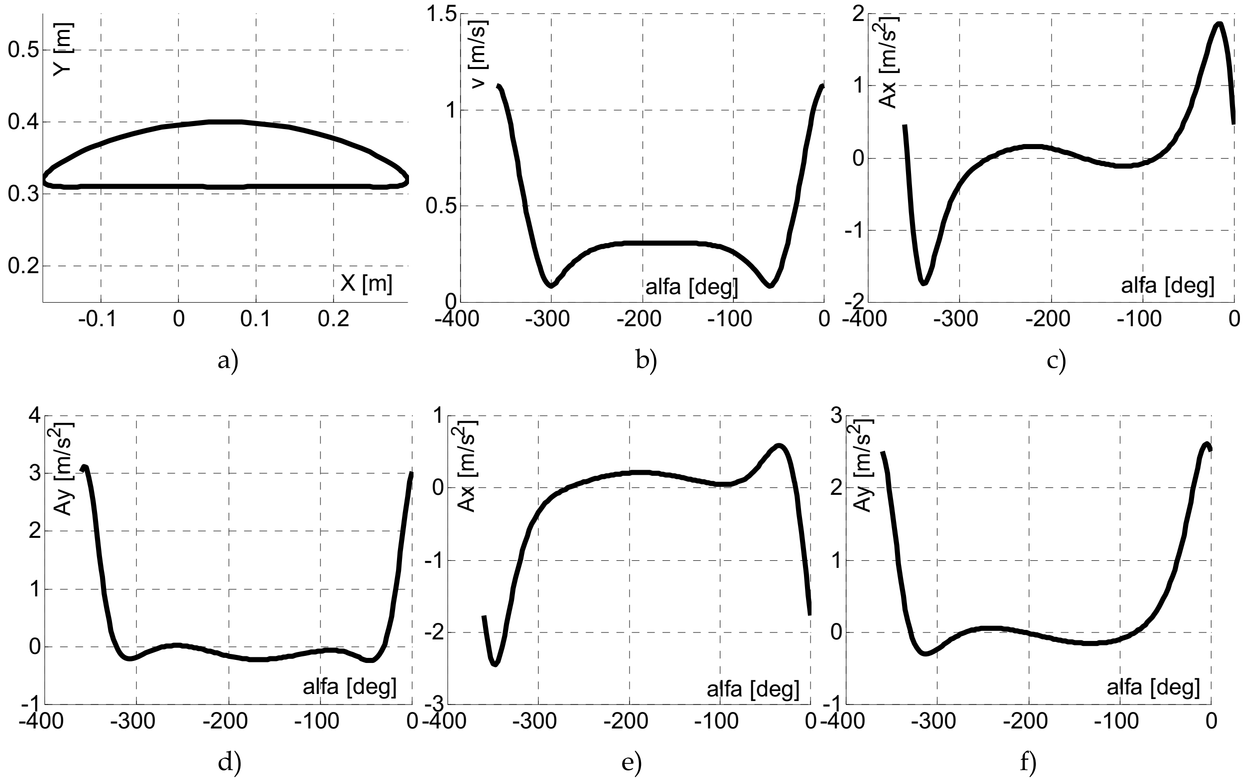


Figure 24. Simulation for the walking characteristics of the 1-DOF leg in Fig.23 with p=20 mm and h=-30 mm: a) Point C trajectory; b) Point A velocity; c) Acceleration a_{AX}; d) Acceleration a_{AY}; e) Acceleration a_{Ax}; f) Acceleration a_{Ay}

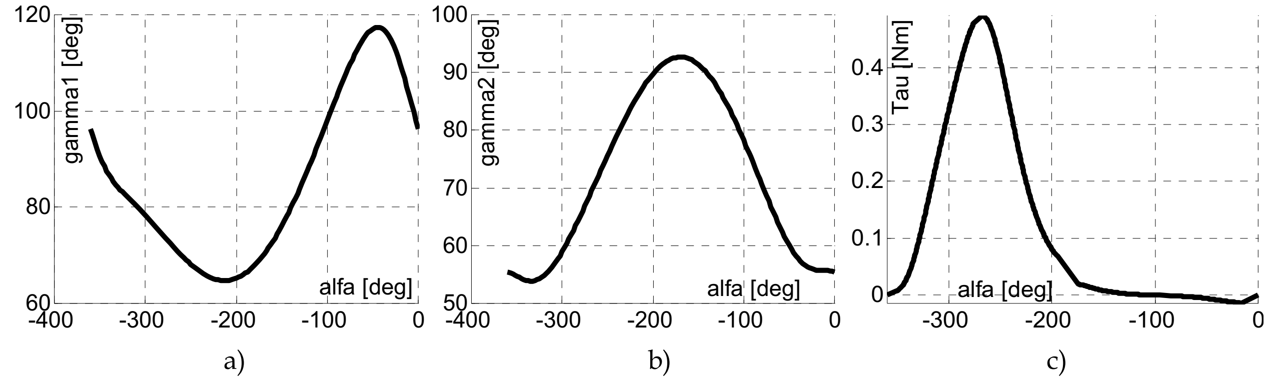


Figure 25. Simulation for the walking characteristics of the 1-DOF leg in Fig.23 with p=20 mm and h=-30 mm: a) Transmission angle γ_1 ; b) Transmission angle γ_2 ; c) actuating torque τ

It is worth noting that the number of link modules can be decided according with the needed number of degrees of freedom. The link modules can be also properly oriented with respect to the others in order to achieve the required pitch, jaw or roll motions. A link module can be also easily modified in order to drive a wheel.

Dimension of the built prototype leg that is composed by 3 modules and one wheel in the foot, is high 500 mm and has a cross-section of 60mm x 60mm. the built leg prototype in Fig. 26 has a maximum lift of 155 mm and the step is of 310 mm. Maximum rotation for each joint is +/- 90 deg.

In Fig. 27 the programming of walking is reported through the scheme of the analysis of elementary actions and corresponding Grafect to use a PLC that will control the operation of the actuators by using signals by suitable switches for the leg mobility. In Fig.28 an example is shown of using the leg for an hexapod design that has been simulated and is under construction at LARM with low-cost easy-operation features.

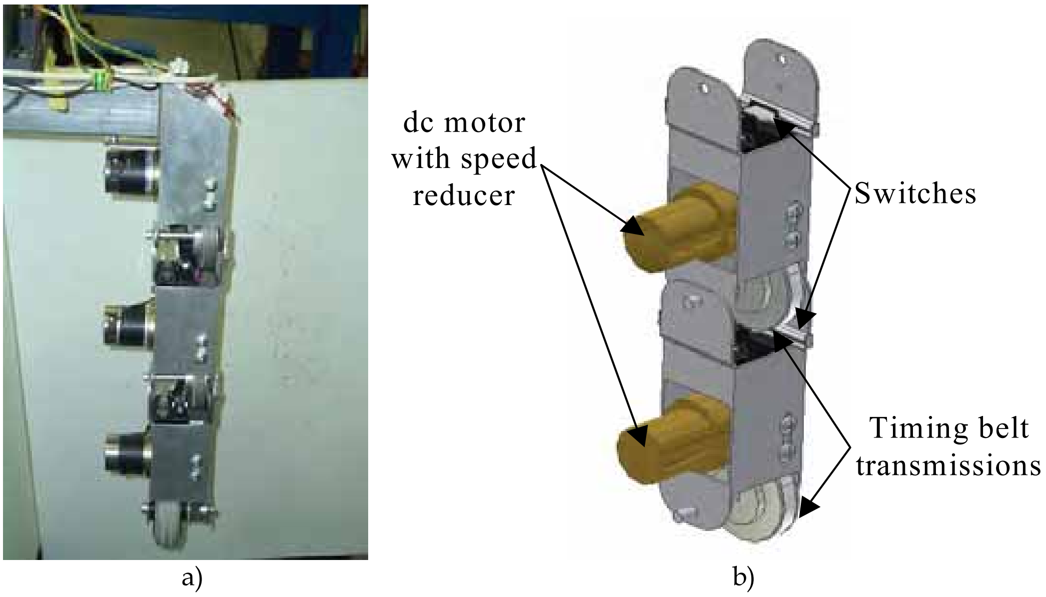


Figure 26. The so-called modular anthropomorphic leg developed at LARM: a) a built prototype; b) the mechanical design

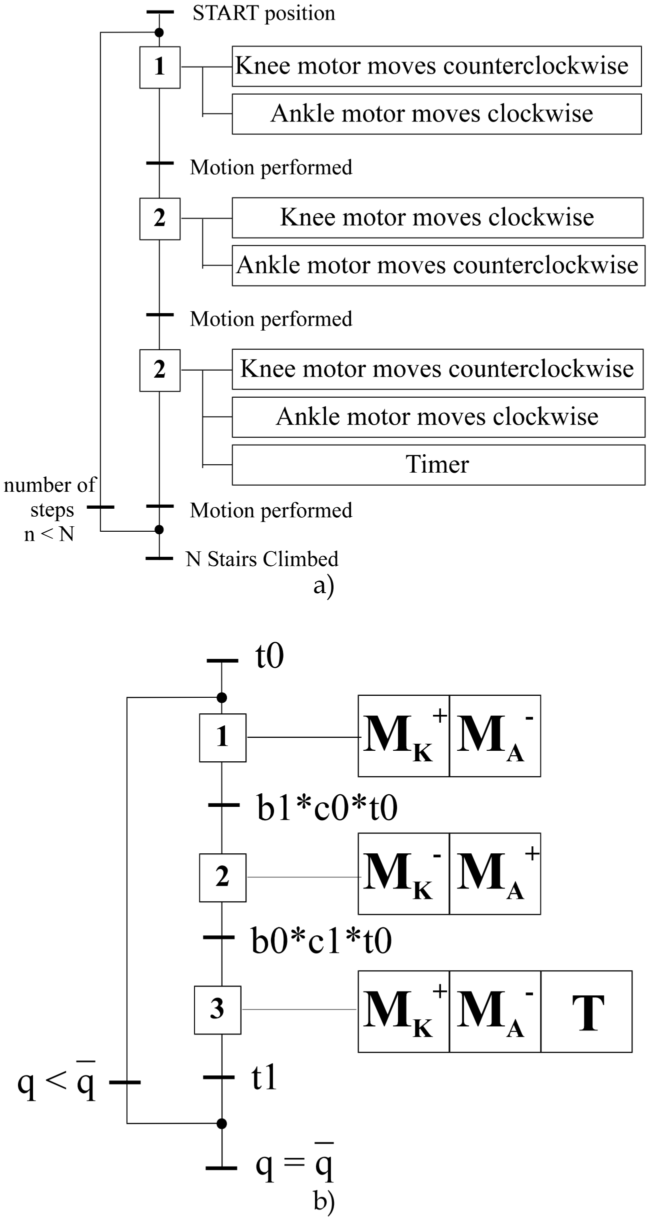


Figure 27. An example of the operation programming for the proposed low-cost anthropomorphic wheeled leg in Fig.26 for the straight walking: a) flowchart of elementary actions; b) Grafcet for PLC programming

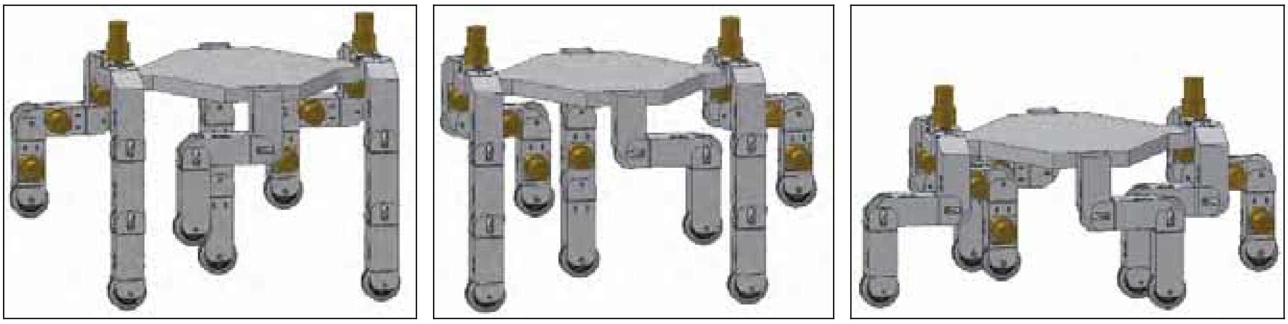


Figure 28. An application of the proposed low-cost modular anthropomorphic leg for hexapod walking machines

6. Conclusions

The variety of currently available walking machines gives many solutions to the problem of artificial walking for many applications. Combining properly suitable designs of mechanical architectures, actuator systems, sensors equipment can give walking machines with very good characteristics and performances that can make prototypes very promising and even already available in the market. However, the challenge for future assessment of walking machines as convenient transportation machinery can be recognized in the development of low-cost easy-operation systems whose designs try to mimic the legged systems in nature but overpassing their performances and reducing their complexity in functionality.

7. References

Technical and scientific literature on walking machines is very reach in terms of Conference and Journal papers, monographs, and books. Since space limits and the introductory character of this manuscript, we have limited the reference list to the main sources for information and further reading.

Active Structures Laboratory, Université Libre de Bruxelles, http://www.ulb.ac.be/scmero/robotics.html, 2005.

Aibo Europe homepage, http://www.aibo-europe.com/, 2005.

Ambulatory Robotic Lab., http://www.cim.mcgill.ca/~arlweb/, 2005.

Androidworld homepage, http://www.androidworld.com, 2005.

Ceccarelli M., "Fundamentals of Mechanics of Robotic Manipulation", *Kluwer Academic Publishers*, Dordrecht, 2004.

Control and Robotics Lab., Chiba University, http://mec2.tm.chiba-u.jp/~nonami/, 2005.

Dickinson M.H., Farley C.T., Full R.J., Koehl M.A., Kram R., Lehman S., "How Animals Move: An Integrative View", *Science,* Vol.288, pp.100-106, 2000.

Figliolini G., Ceccarelli M., "Walking Programming for an Electropneumatic Biped Robot", *Journal of Mechatronics*, Vol.9, pp.941-964, 1999.

Gonzalez de Santos P., Jimenez M.A., Armada M.A., "Dynamic Effects in Statically Stable Walking Machines", *Journal of Intelligent and Robotic Systems*, Vol.23, n.1, pp. 71-85, 1998.

Harris S.E., "Horse Gaits, Balance and Movement", Howell Reference Books, New York, 1993.

Hirose & Yoneda Robotic Lab., Tokyo Institute of Technology, http://www-robot.mes.titech.ac.jp/home_e.html, 2005.

Industrial Automation Institute of CSIC in Madrid, http://www.iai.csic.es/users/gds/web1c2.htm#rimho, 2005.

Iagnemma K., Dubowsky S., "Mobile Robots in Rough Terrain. Estimation, Motion Planning, and Control with Aplication to Planetary Rovers", Springer Verlag, Berlin, 2004.

Intelligent Machines and Special Robotics Institute, Helsinki University of Technology, http://www.automation.hut. fi/IMSRI/workpartner/, 2005.

Joker Robotics, http://www.joker-robotics.com/walker/, 2005.

LARM: Laboratory of Robotics and Mechatronics, University of Cassino, http://webuser.unicas.it/weblarm/larmindex.htm, 2005.

Ottaviano E., Lanni C., Ceccarelli M., "Numerical and Experimental Analysis of a Pantograph-Leg with a Fully-Rotative Actuating Mechanism", 11th World Congress in Mechanism and Machine Science IFToMM2004, pp.1537-1541, 2004.

Plustech Ltd., http://www.plustech.fi/Plmain01.html, 2005.

Research Group IDS, Center of Research FZI, http://www.fzi.de/ids/eng/projekte.php?id=18, 2005.

Salmi S., Halme A., "Implementing and testing a reasoning-based free gait algorithm in the six-legged walking machine Mecant" *Control Engineering Practice*, Vol. 4, n.4, pp. 487-492, 1996.

Walking Machine Catalogue, http://www.walking-machines.org/, 2005.

Zhao Y.S., Lu L., Zhao T.S., Du Y.H., Huang Z., "Dynamic performance analysis of six-legged walking machines" *Mechanism and Machine Theory*, Vol. 35, n.1, pp. 155-163, 2000.

Zielinska T., Heng J.,"Development of a walking machine: mechanical design and control problems", *Mechatronics*, Vol.12, n.5, pp.737-754, 2002.

Humanoid Robot Motion in Unstructured Environment – Generation of Various Gait Patterns from a Single Nominal

Miomir Vukobratovic, Dejan Andric & Branislav Borovac

1. Introduction

Humanoid robotics has been in the focus of scientific community for decades. For a long time already, robots have not been present only in the industrial plants, at a time their traditional work space, but have been increasingly more engaged in the close living and working environment of humans. This fact inevitably leads to the need of "working coexistence" of man and robot and sharing their common working environment. The fact that no significant rearrangement of the environment of humans could be expected as a consequence of the presence of their mechatronic partners, robots will have to further "adapt" to the environment previously dedicated only to men. Besides, it is expected that the robots cooperating with humans will have an operation efficiency matching that of humans. The working and living environment, adapted to humans, imposes on robot's mechanical-control structure at least two types of tasks related to its motion: motion in a specific environment with the obstacles of the type of staircases, thresholds, multi-level floors, etc., and the motion within a very dynamic scene in which the trajectory of system's motion can be only globally planned and the actual trajectory determined on the basis of the instantaneous situation.

The first task is solved by using legged locomotion instead of wheels, and since the robots will be in a "close contact" with men and will act in the working and living environment adapted to humans, the most logical choice is the bipedal locomotion.

The other important aspect of the above problem is the planning of the robot's motion. While in industrial robotics, dealing with a very structured environment, it is possible to plan the entire robot's motion in the task execution and then this motion is repeated over and over again, in this sort of tasks such detailed planning is not possible because the human's environment in which robot is to move is very dynamic and unstructured. A basic characteristic of the locomotion activity in a dynamic and unstructured environment is the need to swiftly modify the motion trajectory, planned previously only in a global way. Thus, for example, there may appear the need to bypass an obstacle that suddenly appeared, accelerate or slow down the walk, and the like. Naturally, each of these modified gaits can be synthesized as a separate pattern, stored in the memory of the robot control system, and requested when needed. However, two question arise then: a) How many trajectories are to be synthesized in order to "cover" all the situations that might be of interest, that is that might appear?, and b) Is it possible to accomplish in real time to search through all the set of synthesized trajectories and retrieve the most appropriate one in the given situation? In order to cope with these issues and enable robot to react sufficiently fast to the disturbances and realize the previously “unplanned” trajectories in real time, we propose here an approach based on modification of the adopted in advance, nominal motion (straight motion on a level surface) that has been synthesized by semi-inverse method. The basis for preserving dynamic balance is the method of Zero-Moment Point (ZMP).

During the walk, all biped mechanism joints are powered and directly controllable except for the contact of the foot and the ground (it can be considered as an additional DOF), which is the only site in which the mechanism interacts with the environment. This contact is essential for the walk realization because the mechanism's position with respect to the environment depends on the relative position of the foot with respect to the ground. The foot cannot be controlled directly but in an indirect way – by ensuring appropriate dynamics of the mechanism above the foot. Thus, the overall indicator of the mechanism behavior is the point where the influence of all the forces acting on the mechanism can be replaced by one single force; hence, this point was termed Zero-Moment Point (Vukobratovic M. & Juricic D. 1968, Vukobratovic M. & Juricic D. 1969, Vukobratovic M. & Stepanenko Yu., 1972). Recognition of the significance and role of ZMP in biped artificial walk was a turning point in robotic gait planning and control.

In the gait synthesis by semi-inverse method (Vukobratovic M. & Juricic D. 1968, Vukobratovic M. & Juricic D. 1969, Vukobratovic M. & Stepanenko Yu., 1972) legs’ trajectories are predefined while the trunk motion is determined so as to ensure dynamic balance of the system as a whole (the ZMP position is within the desired area under the supporting foot). Such motion is called nominal motion.

The problem addressed in this paper is how to achieve robot's coping with the situations in the dynamic scene and the motion realization if the change of the planned trajectory is to be carried out on-line. The idea we propose is to use a nominal motion selected in advance as a basic motion pattern (we used a nominal synthesized by semi-inverse method for the straight walk on a flat ground), whereas all other motions are to be obtained by its on-line modification (Vukobratovic et al. 2004), taking simulatneously care that the condition of dynamic balance is constantly satisfied. This approach is also convenient because it encompasses the cases when a situation arises when small modifications of the basic gait regime are needed to adapt to the momentary requirements imposed on the biped system. We believe that, at least in some basic cases, it is not necessary to perform a new nominal synthesis but is possible to do on-line modification of the existing nominal.

2. Mechanism Structure and its Nominal Motion

Our idea was tested on the example of a locomotion mechanism (Fig. 1) having 20 DOFs, with all joints being rotational (Vukobratovic et al. 1990). Between two joints is placed one link. A joint having more DOFs is modeled as a set of simple rotational joints, each having only one DOF. In the case of a multi-DOF joint, modeling the links between single-DOF joints are considered as being of zero length, mass and moments of inertia. In Fig. 1, these "fictitious" links are denoted by dashed line (for example link 3 at the ankle joint, links 6 and 7 at the hip joint of the right leg, etc.). Accordingly, the ankle joint having 2 DOFs is represented by joints 3 and 4 (the joint axes are defined by the unit vectors e3 and e4), hip joint of the right leg by joints 6, 7 and 8, trunk joint (waist) by joints 15 and 16. All other joints are modeled as simple rotational joints having one DOF.

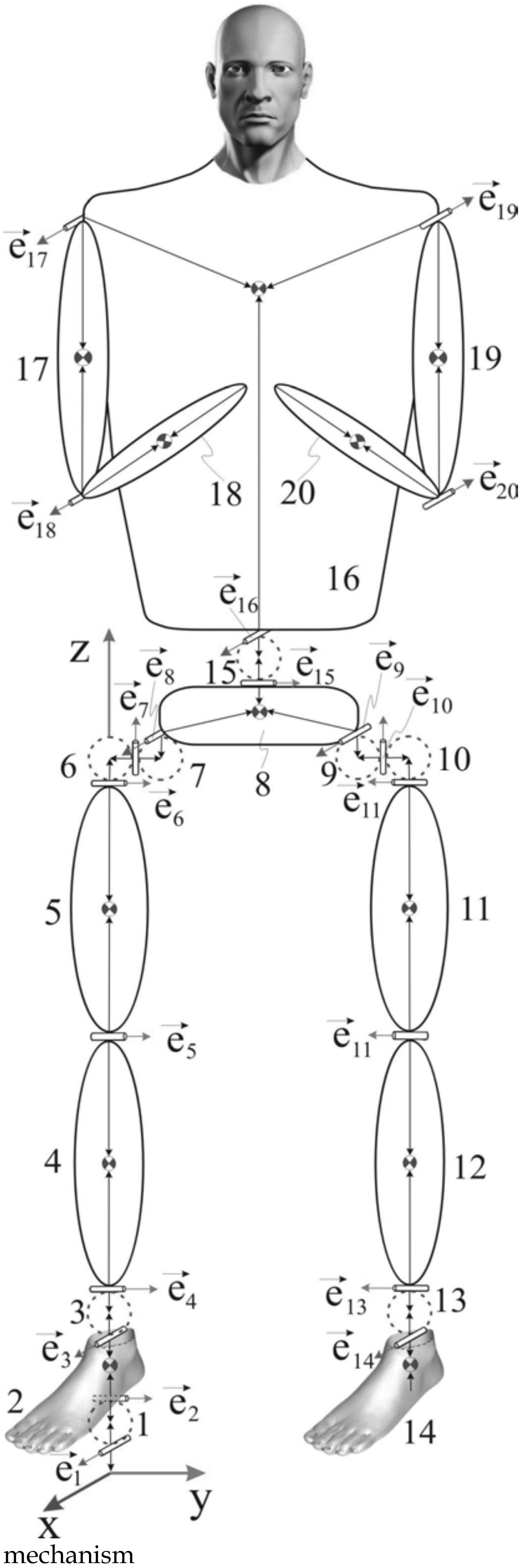


Figure 1. Scheme of locomotion mechanism

The possibility of the mechanism's overturning about the foot edge is modeled by two "fictitious" joints in the directions of the axes x and y to detect if the mechanism as a whole will rotate in the y-z or x-z plane. Each mechanism joint is powered by a DC motor except for joints 1 and 2 under the foot, which represent unpowered DOFs. All parameters of the mechanism can be found in [4]. The mechanism motion is synthesized using semi-inverse method, whereby dynamic balance is permanently ensured during the walk

(Vukobratovic et al. 1990). Legs' motion pattern was obtained by recording the performance of a human subject, and then, the trunk motion was synthesized in such a way to ensure that ground reaction force under the foot is in a certain predefined position (in our example this is the point O of the coordinate system Oxyz), ensuring simultaneously that horizontal components of the ground reaction moment are equal to zero, i.e. Mx=My=0. As was already said, this point is known as ZMP. Each change of the dynamics above the supporting foot causes displacement of the ZMP out of its nominal position. Walk synthesis has been carried out for one half-step of the single-support phase only. The mechanism walks in such a way that while one leg is in support phase the other one is transferred from the back to front position. When it reaches the position just above the ground floor the supporting foot is changed and the walk continues. Footprints of the mechanism nominal motion (ZMP nominal positions) are presented in Fig. 2.

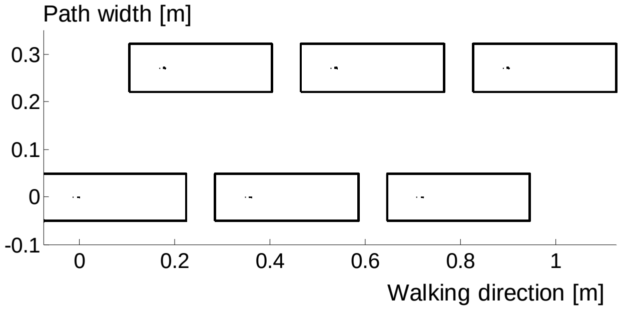


Figure 2. Mechanism footprints and ZMP positions during nominal motion

If during the walk the ZMP position comes out of the support polygon, this means that the mechanism is not performing dynamically balanced gait any more, and it collapses by rotating about the foot edge. ZMP position, being very sensitive to the changes of dynamics of the mechanism above the foot, is the best indicator of the overall mechanism dynamic balance. In Fig. 3. is shown a stick diagram of the mechanism performing nominal motion.

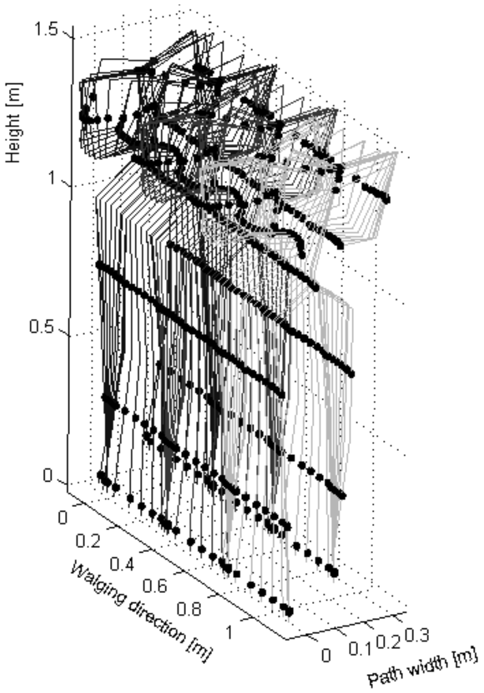


Figure 3. Stick diagram of the mechanism performing nominal motion

3. Modification of the Nominal Motion

3.1 Turning

The nominal motion modification to be considered first is the change of the walk direction. Namely, very often there arises a need to switch from a rectilinear motion to the right/left direction without interrupting the walk, so that it would be highly desirable to achieve such a gait by modifying the nominal rectilinear motion.

A first thing to be considered is how to perform turning. The angle α formed between the foot at the end of a single-support phase and the foot that up to that moment was in contact with the ground will be adopted as the measure of the extent of turning.

A smaller α corresponds to a "milder" and a larger to a "sharper" turning. Besides, care should be exercised as to the relative position of the mechanism's feet (legs) with respect to the "body". In the case of the mechanism used (see Fig. 2) the turning will be realized by rotation at joints 7 and 10, so that both joints are activated simultaneously. A simplest algorithm would be to perform turning so that in each half-step the leg being in the swing phase rotates so that the foot that is making contact with the ground forms an angle α with respect to the foot that was previously in contact with the ground. However, this algorithm did not perform well since the "outer" and "inner" leg followed the arcs of different lengths so that a situation evolved soon when the motion could not be continued at all.

A solution to this problem would be to perform deflection in the first half-step, and in the second half-step the correction of deviation due to the foot rotation, so that in the beginning of the next step the feet are not deflected with respect to the trunk.

The next thing to be considered is the way in which foot turning by the angle α will be realized – whether once or incrementally in the course of the whole half-step, so that at the end the foot deflects by the full value of the angle α. In the present work we chose gradual increase of the turning angle.

Let the desired value of foot deflection at the end of a half-step be defined by the angle α. If the number of integration intervals during a half-step is n_{int} the value of deflection angle to be realized in one integration interval is $\Delta\alpha = \alpha / n_{int}$. This value is halved and the value (Δα/2) is added to each of joints 7 and 10. Let us explain this on the example of turning to the left, as shown in Fig. 4. During the first half-step, the right leg is in the support and the left leg in the swing phase. The angles are added in the following way:

$$\begin{aligned} \text{angle}7 &= \text{angle}7_{\text{nom}} + i \cdot (\Delta\alpha/2) \\ \text{angle}10 &= \text{angle}10_{\text{nom}} + i \cdot (\Delta\alpha/2) \end{aligned} \tag{1}$$

where i denotes ordinal number of the integration interval while $\text{angle}7_{\text{nom}}$ and $\text{angle}10_{\text{nom}}$ are angle values for nominal motion. In the end of this phase, the left foot is deflected relative to the right foot by the angle α (note that first and second footsteps are not parallel). In the second half-step, to compensate for the deviation of legs' positions relative to the mechanism body, the hip positions change in the following way:

$$\begin{aligned} \text{angle}7 &= -(\text{angle}7_{\text{nom}} + i \cdot (\Delta\alpha/2)) \\ \text{angle}10 &= -(\text{angle}10_{\text{nom}} + i \cdot (\Delta\alpha/2)) \end{aligned} \tag{2}$$

In the end of the second half-step the right foot becomes parallel to the left foot. After that, the procedure continues and the left foot angle relative to its previous position increases again, the right foot compensates for this deviation, and so on.

In the following figures are given examples of the algorithm application. In Figs. 4, 5 and 6 are presented footprints for the angle α of 5o, 15o and 25o, respectively, to illustrate how the change of walking direction affects dynamic balance of nominal motion. From Fig. 4 it is clear that the ZMP is close to its nominal position (far enough from the foot edge), and no compensation is needed. However, as is evident from Figs. 5 and 6 that, when the angle α increases the ZMP deviation from its nominal position increases too. In Fig. 5, it is very close to the foot edge, while in Fig. 6 it goes out of the support polygon and correction of ZMP position is necessary. To compensate for ZMP deviation we propose to add a permanent gravitational moment which can be produced by body inclination. Now, a question arises as to the choice of the joint most suitable for this operation. We investigated all possibilities (trunk, hip, and ankle) and found that the compensation by the ankle gave the best results.

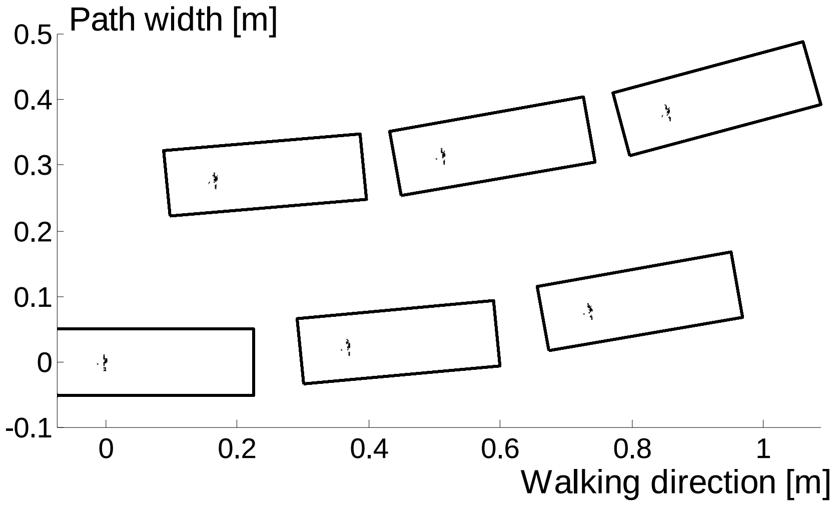


Figure 4. Mechanism footprints when α is 5°. No compensation applied

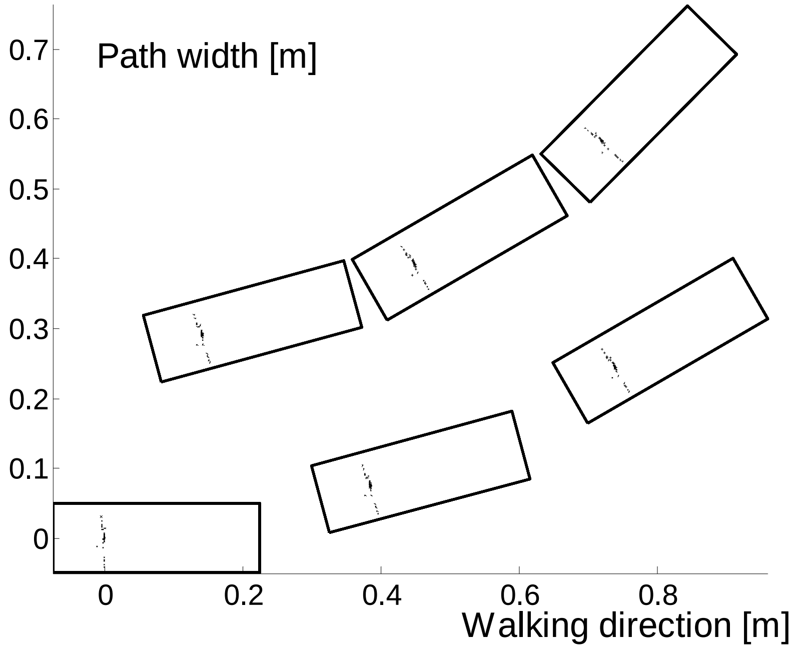


Figure 5. Mechanism footprints when α is 15 ° . No compensation applied.

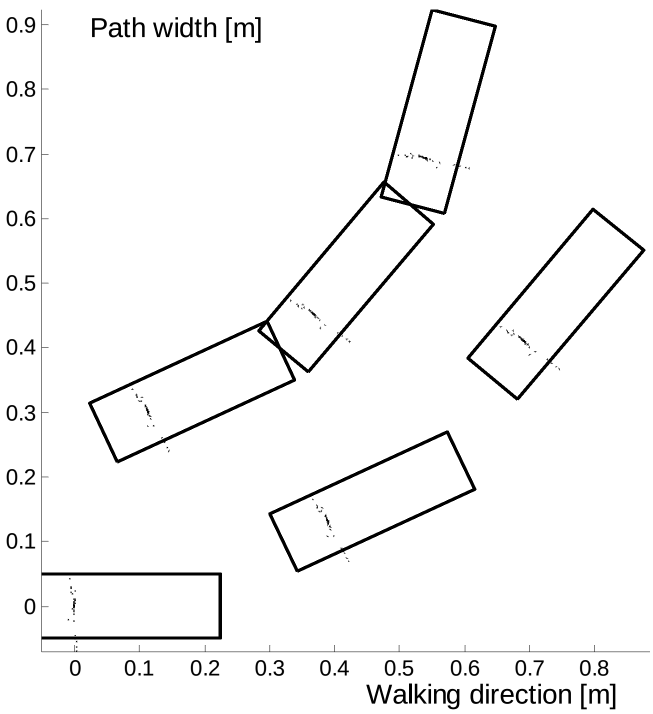


Figure 6. Mechanism footprints when α is 25° . No compensation applied.

In Fig. 7 are given footprints for the same case as shown in Fig. 6 (α = 25o) but with the compensation of 3o at joint 3. It can be seen that the ZMP position deviations are significantly reduced and that all ZMP positions are within the support polygon. Thus, by such compensation, dynamic balance is preserved and the walk continuation is ensured.

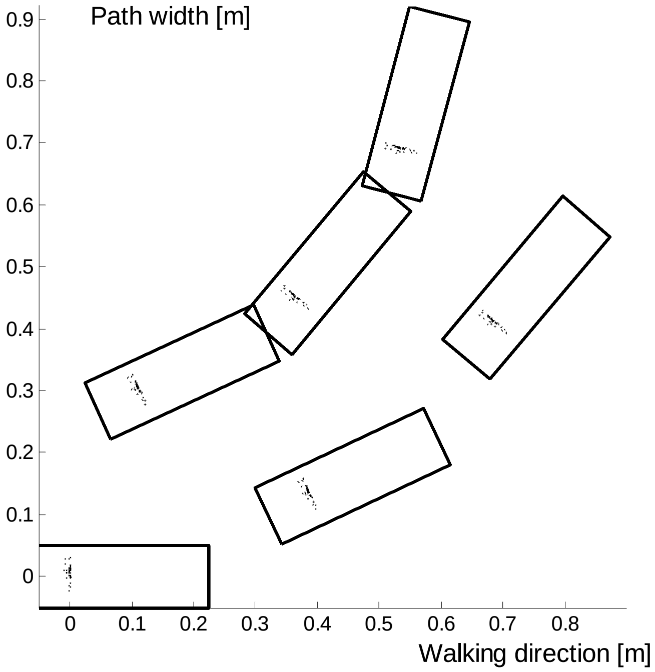


Figure 7. Mechanism footprints when α is 25° . Compensation at the ankle joint (joint 3) 3° .

3.1 Walk Speed-Up and Slow-Down

The next case we investigated was how to speed up and slow down the walk. Nominal motion was the same as before. To change walking speed, moments applied at all joints have to be changed. To avoid recomputing the complete dynamics we used the procedure proposed by Hollerbach (1984) for changing robotic mechanism motion speed without recomputing completely its dynamics.

Let us remind that the mechanism dynamics is given in the following general form:

$$\tau = H(q)\ddot{q} + h(\dot{q}) + G \tag{3}$$

where τ is the vector of driving torques at joints; q, $\dot{q}$ and $\ddot{q}$ are the vectors of joint angles, velocities and accelerations, respectively; $H(q)$ and $h(\dot{q})$ are the inertial matrix and the vector that includes all velocity effects, while the vector G represents gravitational moments. Gravitational forces and their moments do not depend on motion speed. Thus, in (3) we can separate the moments that are acceleration and velocity-dependent from those which are not. If we denote by τ_n the acceleration- and velocity-dependent torques of non-accelerated motion and by τ_a those of accelerated motion, then the following relation (Hollerbach J. M. 1984) will hold:

$$\tau_a(t) = c^2 \cdot \tau_n(ct) \tag{4}$$

whereby c > 1 stands for the accelerated and c < 1 for decelerated motion.

We applied this method to investigate how the acceleration/deceleration of nominal (dynamically balanced) walk would affect its dynamic balance. In Figs. 8-12 are shown the cases of walk speeding up for various values of c.

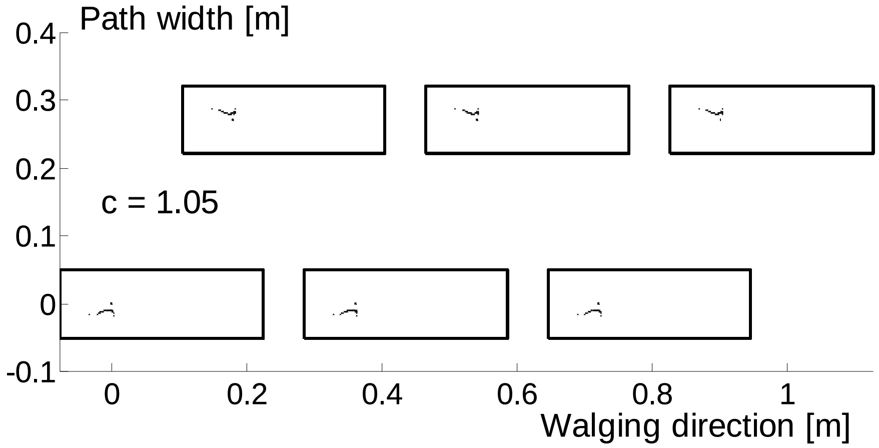


Figure 8. Mechanism footprints of accelerated motion for *c* = 1.05. No compensation applied.

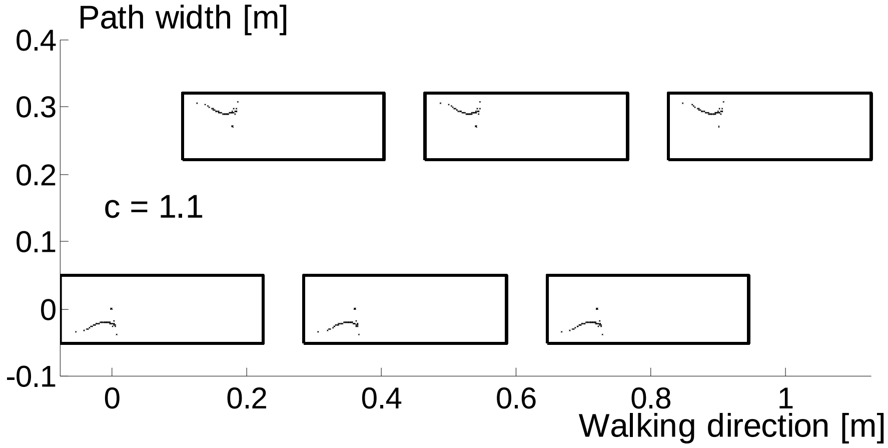


Figure 9. Mechanism footprints of accelerated motion for c = 1.1. No compensation applied.

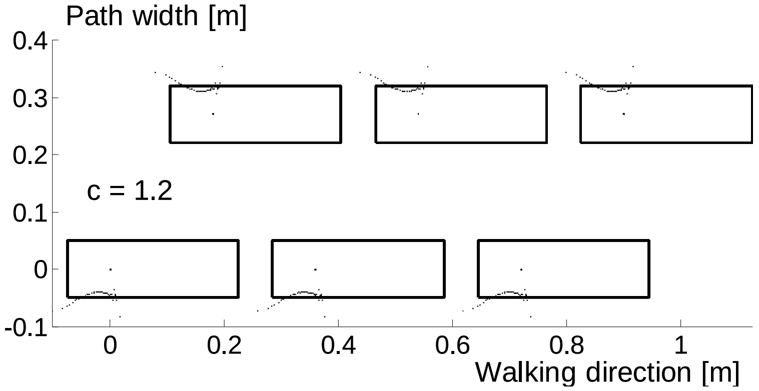


Figure 10. Mechanism footprints of accelerated motion for c = 1.2. No compensation applied.

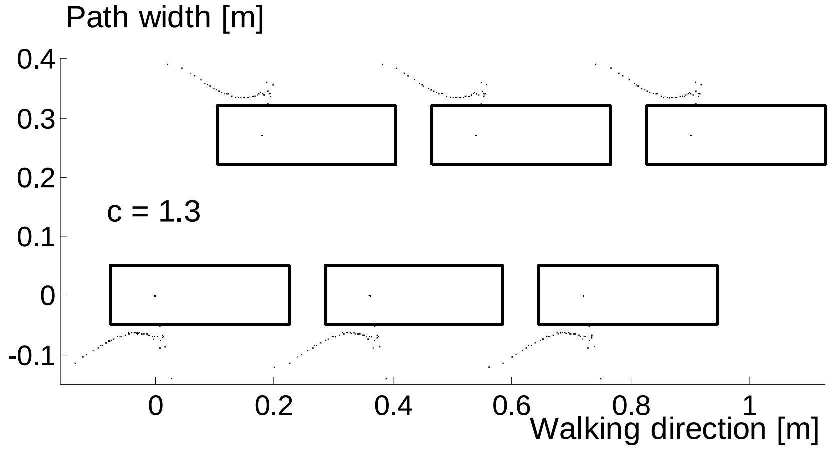


Figure 11. Mechanism footprints of accelerated motion for c = 1.3. No compensation applied.

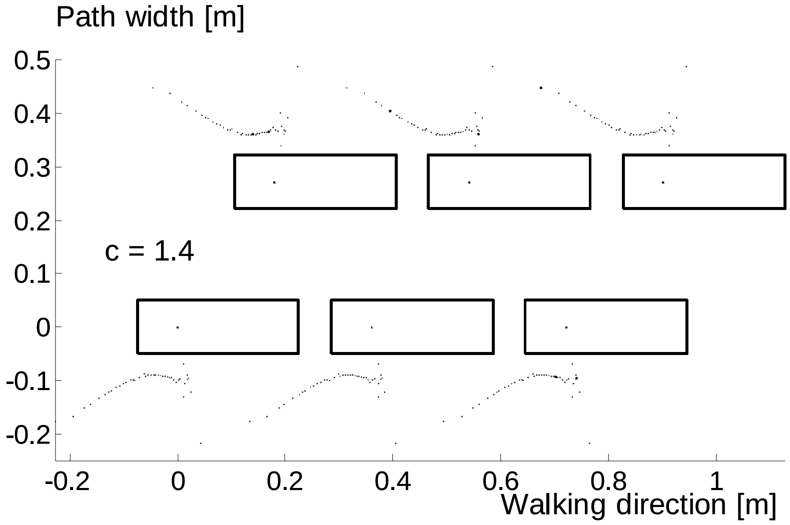


Figure 12. Mechanism footprints of accelerated motion for c = 1.4. No compensation applied.

The presented figures illustrate well the importance of the altered mechanism dynamics on the ZMP position. For c > 1.1 (Figs. 10 – 12) it is clear that the ZMP is out of the support polygon and that the mechanism's dynamic balance is not preserved. To make possible continuation of the walk it is necessary to correct the ZMP position.

To achieve this we applied the same strategy as before – inclination of the body. In Figs. 13-17 are presented the correction effects when compensation was performed by joints 3 and 4 (ankle). Joint 3 compensated for deviations in the sagittal and joint 4 in the frontal plane.

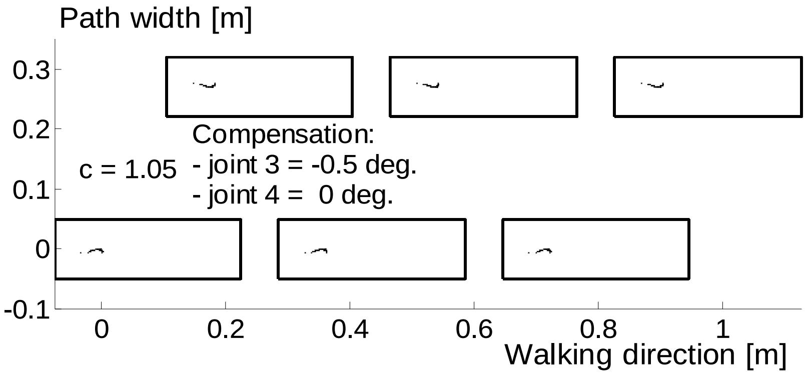


Figure 13. Mechanism footprints of accelerated motion for c = 1.05. Compensation of –0.5° applied at joint 3; no compensation at joint 4.

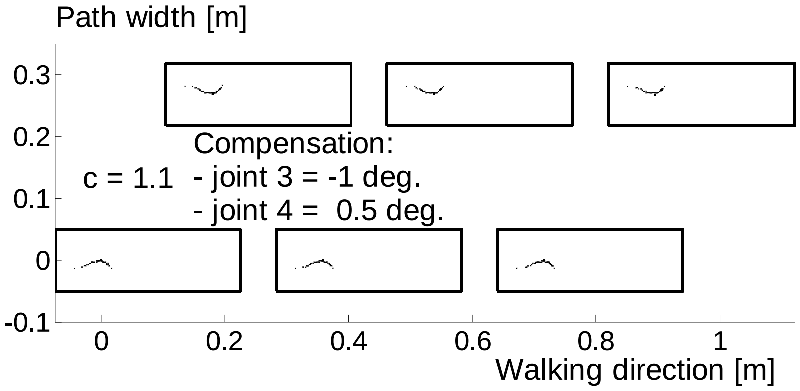


Figure 14. Mechanism footprints of accelerated motion for c = 1.1. Compensation of –2 ° applied at joint 3, of 0.5 ° at joint 4.

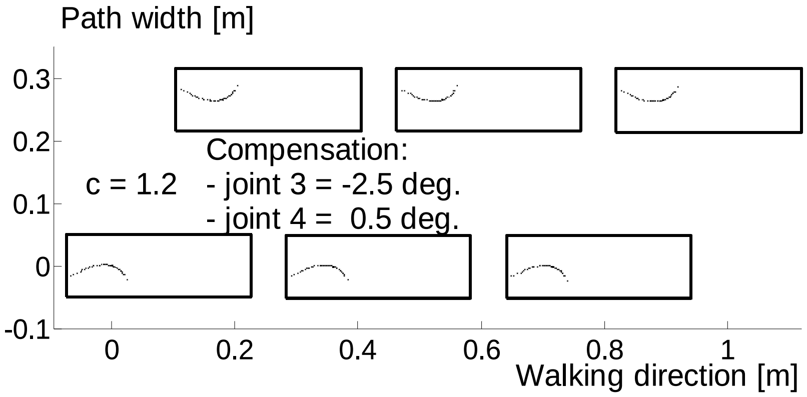


Figure 15. Mechanism footprints of accelerated motion for c = 1.2. Compensation of –2.5 ° applied at joint 3, of 0.5 ° at joint 4.

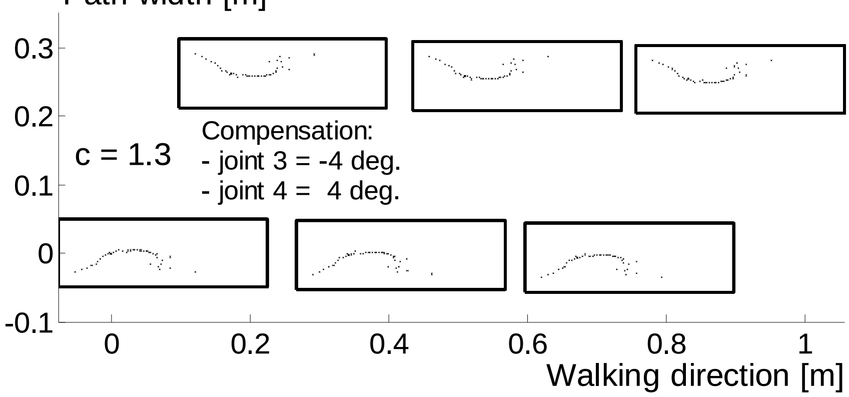


Figure 16. Mechanism footprints of accelerated motion for c = 1.3. Compensation of –4 ° applied at joint 3, of 4 ° at joint 4.

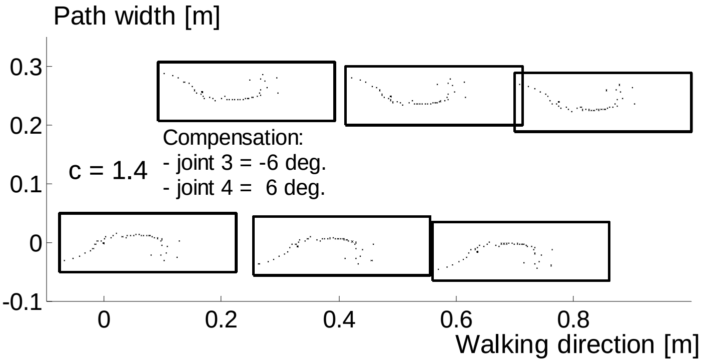


Figure 17. Mechanism footprints of accelerated motion for c =1.4. Compensation of –6° applied at joint 3, of 6° at joint 4.

The presented examples show that the corrections made by the ankle joint brought the ZMP back to the support polygon. Figs. 18-22 illustrate the situations when the ZMP deviation was compensated by the trunk joints (joints 15 and 16).

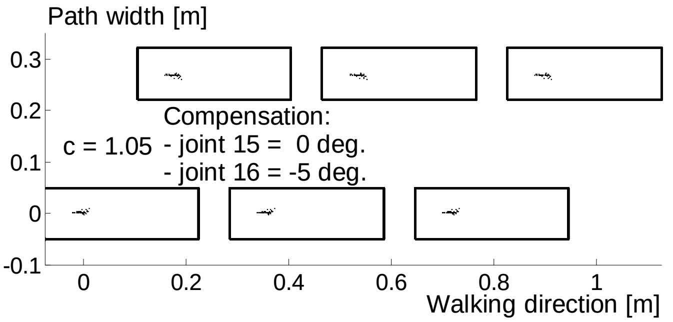


Figure 18. Mechanism footprints of accelerated motion for c = 1.05. Compensation of 0° applied at joint 15, of -5° at joint 16.

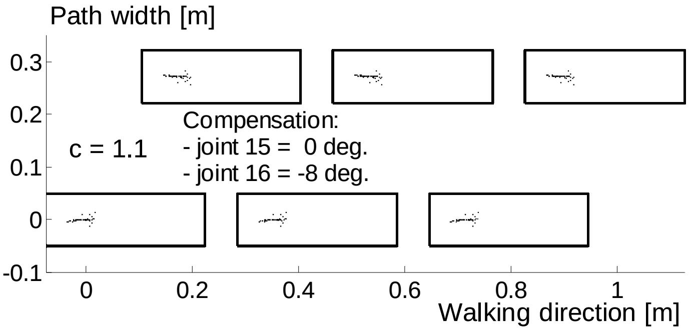


Figure 19. Mechanism footprints of accelerated motion for c = 1.1. Compensation of 0° applied at joint 15, of –8° at joint 16.

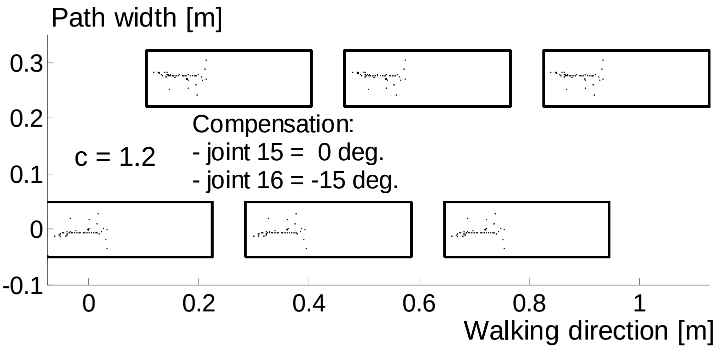


Figure 20. Mechanism footprints of accelerated motion for c = 1.2. Compensation of 0° applied at joint 15, of –15° at joint 16.

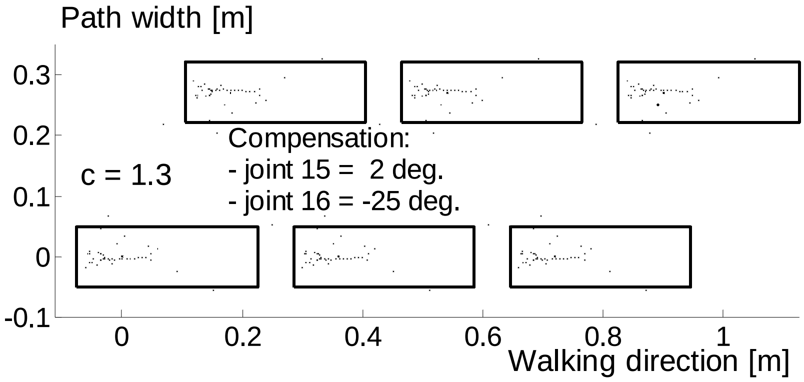


Figure 21. Mechanism footprints of accelerated motion for c =1.3. Compensation of 2° applied at joint 15, of –25° at joint 16.

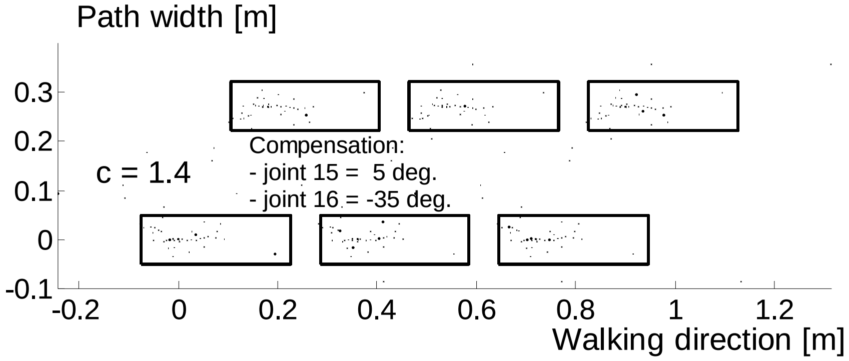


Figure 22. Mechanism footprints of accelerated motion for c =1.4. Compensation of 5° applied at joint 15, of –35° at joint 16.

It is evident that much larger inclination angles are needed for compensation performed by the trunk compared to ankle joint (35° compared to 6° for c =1.4). Such large compensation inclinations may jeopardize the anthropomorphism of the walk. Besides, in case of compensation by the trunk for c = 1.3 and 1.4 the ZMP positions were not brought completely back to the support polygon. Hence, it is clear that compensation by the ankle joint is more efficient.

In Figs. 23-27 are presented examples of decelarated motion (c = 0.95 – 0.5). It is evident that spatial dissipation of the ZMP trajectory is lower. This can be explained by the diminished overall influence of inertial forces, so that the ZMP trajectory converges to the trajectory of the system's gravity center, which is a sinusoide lying symmetrically between the mechanism's footprints.

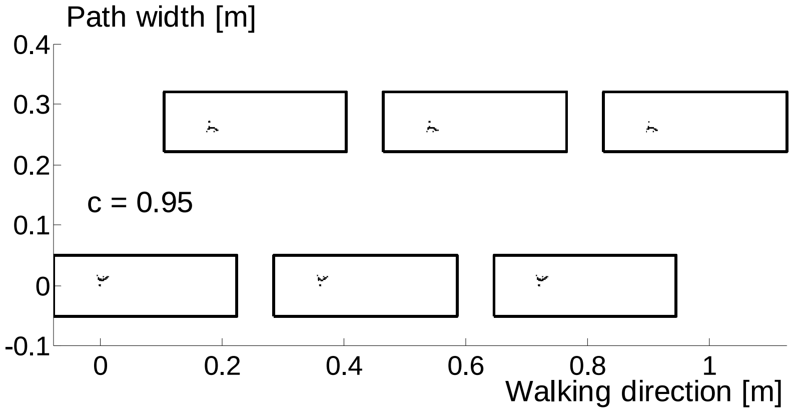


Figure 23. Mechanism footprints of decelerated motion for c = 0.95. No compensation applied.

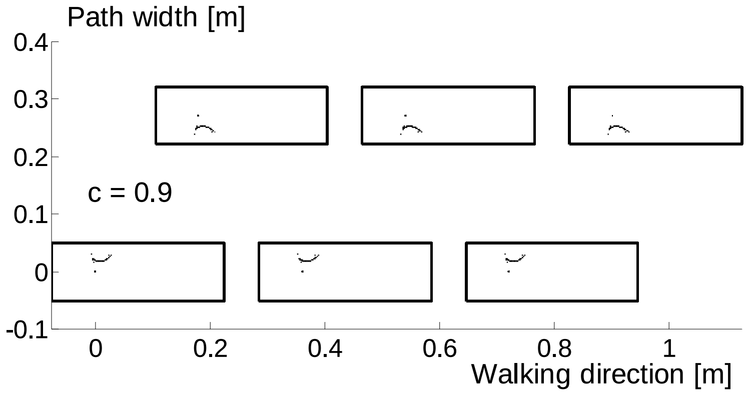


Figure 24. Mechanism footprints of decelerated motion for c = 0.9. No compensation applied.

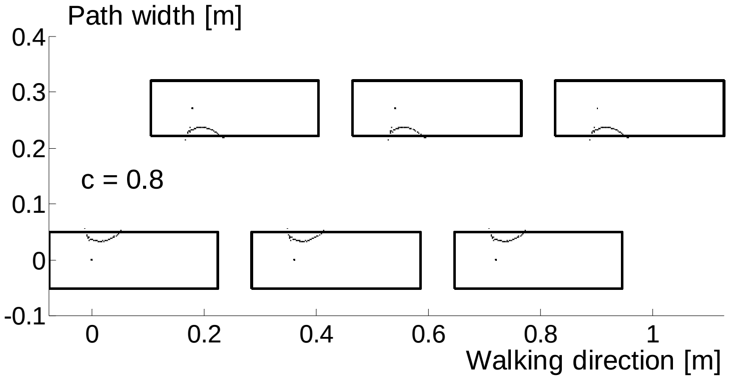


Figure 25. Mechanism footprints of decelerated motion for c = 0.8. No compensation applied.

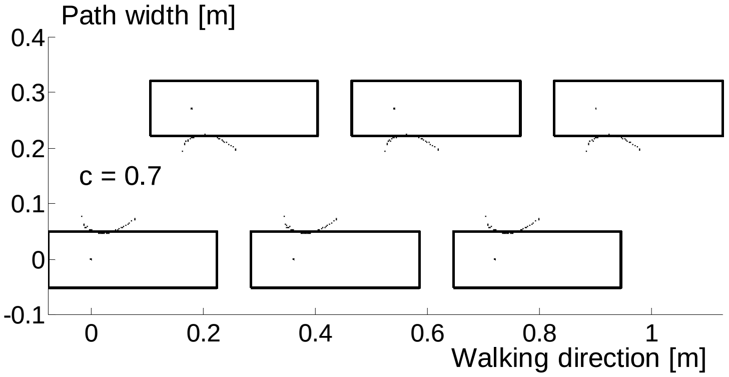


Figure 26. Mechanism footprints of decelerated motion for c = 0.7. No compensation applied.

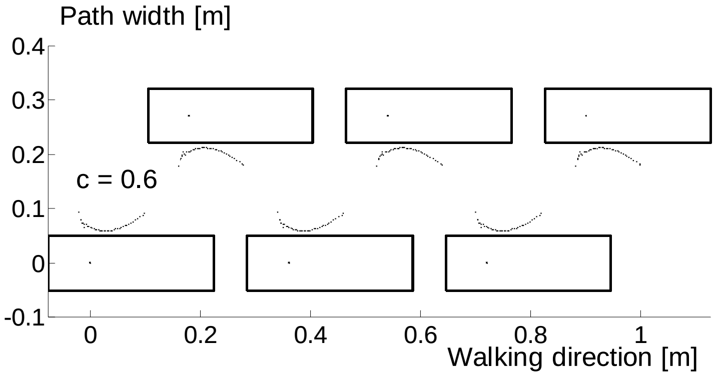


Figure 27. Mechanism footprints of decelerated motion for c = 0.6. No compensation applied.

To compensate for ZMP deviations we applied again the same strategy, i.e. body inclination. As can be seen, in both cases compensation (by the ankle (Figs 28-32) and the trunk joint (Figs. 33-37)) was successful.

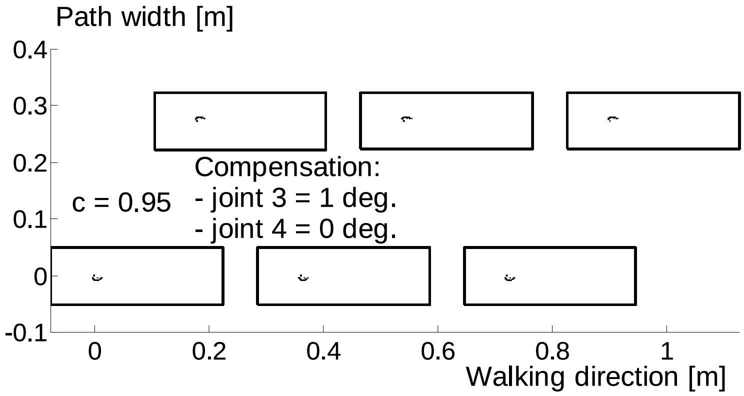


Figure 28. Mechanism footprints of decelerated motion for c = 0.95. Compensation of 1° applied at joint 3; no compensation at joint 4.

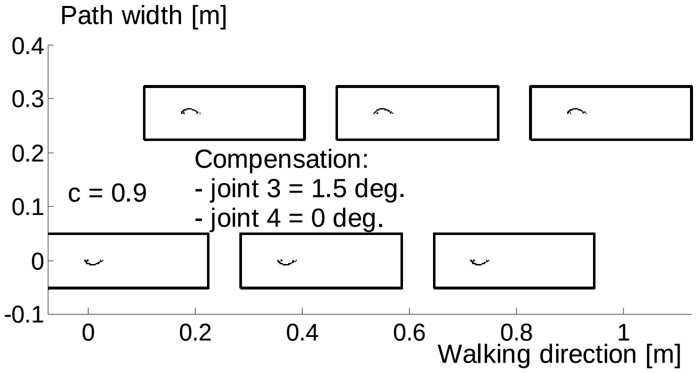


Figure 29. Mechanism footprints of decelerated motion for c = 0.9. Compensation of 1.5° applied at joint 3; no compensation at joint 4.

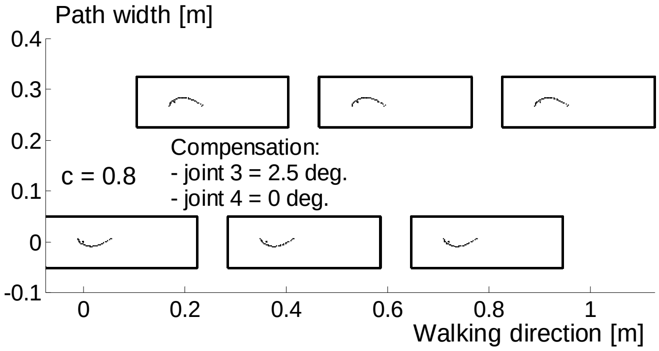


Figure 30. Mechanism footprints of decelerated motion for c = 0.8. Compensation of 2.5° applied at joint 3; no compensation at joint 4.

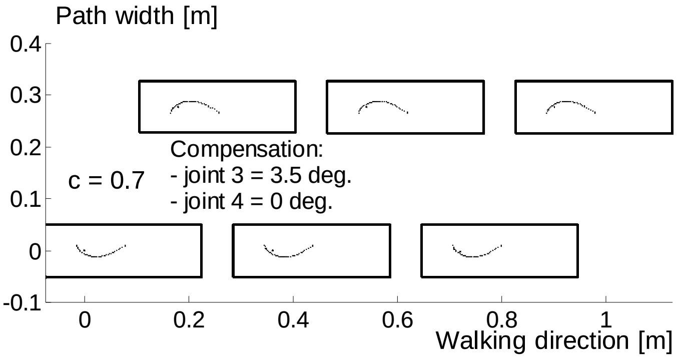


Figure 31. Mechanism footprints of decelerated motion for c = 0.7. Compensation of 3.5° applied at joint 3; no compensation at joint 4.

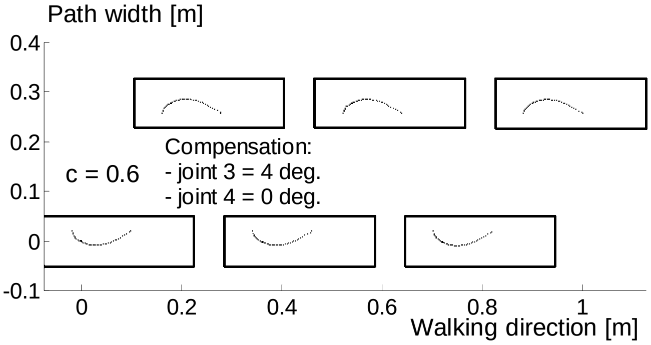


Figure 32. Mechanism footprints of decelerated motion for c = 0.6. Compensation of 4° applied at joint 3; no compensation at joint 4.

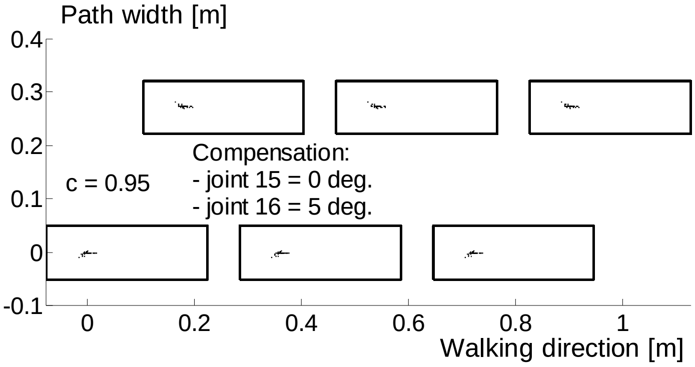


Figure 33. Mechanism footprints of decelerated motion for c = 0.95. Compensation of 0° applied at joint 15, of 5° at joint 16.

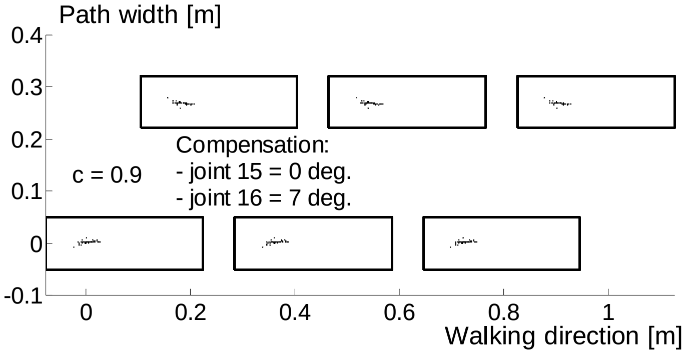


Figure 34. Mechanism footprints of decelerated motion for c = 0.9. Compensation of 0° applied at joint 15, of 7° at joint 16.

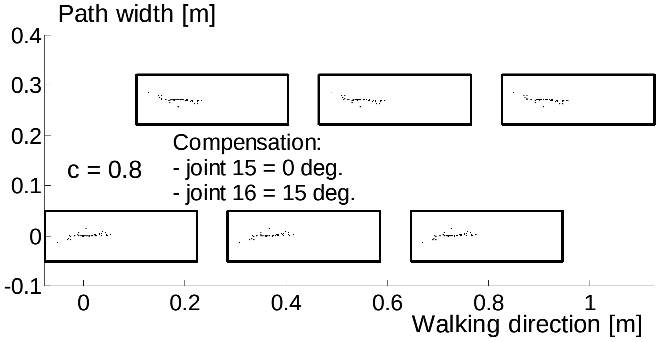


Figure 35. Mechanism footprints of decelerated motion for c = 0.8. Compensation of 0° applied at joint 15, of 15° at joint 16.

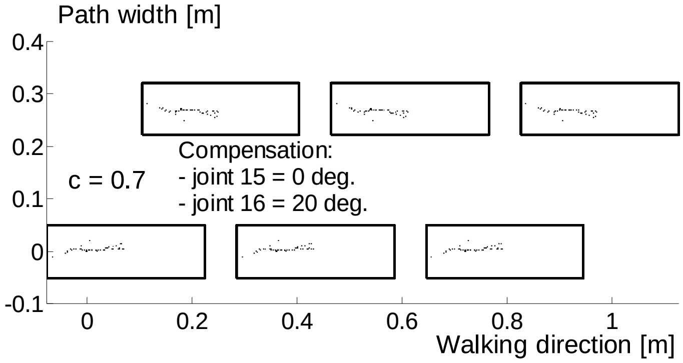


Figure 36. Mechanism footprints of decelerated motion for c = 0.7. Compensation of 0° applied at joint 15, of 20° at joint 16.

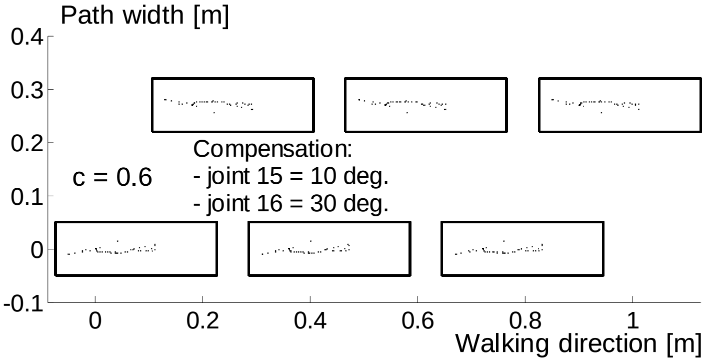


Figure 37. Mechanism footprints of decelerated motion for c = 0.6. Compensation of 10° applied at joint 15, of 30° at joint 16.

Dispersion of ZMP positions is smaller in case of compensation performed by the ankle joint. Thus, it can be concluded that although both joints are suitable for compensation in case of deceleration, the ankle is advantageous over the trunk.

3.3 Step Extension

The next basic modification of the nominal walk is change of step length. In this paper step extension will be considered, only. To extend step maximal angle between two legs for nominal step (in Fig. 38 denoted by β_{nom}) should be enlarged.

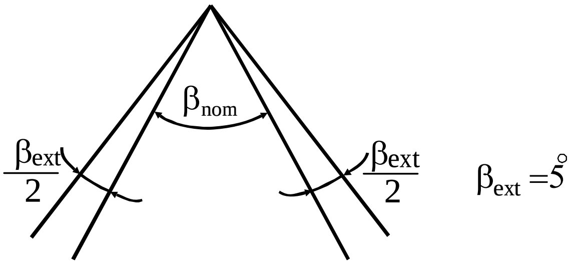


Figure 38. Angles β_{nom} and β_{ext}

Let us denote total value of the added angle as $\beta_{ext.}$ Total amount of β_{ext} is divided between two hip joints (joint 6 for the right, joint 11 for left leg) and, accordingly, $\beta_{ext}/2$ is added per each leg. Thus, maximal angle between the legs for an extended step is $\Delta\beta$==β_{ext}/n_{int}. As in the case of turning, the angle beta should be enlarged gradually. This means that in each sampling time actual value of angles 6 and 11 is enlarged by

$\Delta\beta=\beta_{ext}/n_{int}$, where n_{int} is the number of integration intervals during a half-step:

$$\begin{aligned}\text{angle}6 &= \text{angle}6_{\text{nom}} + i\cdot(\Delta\beta/2)\\ \text{angle}11 &= \text{angle}11_{\text{nom}} + i\cdot(\Delta\beta/2)\end{aligned} \tag{5}$$

where, again, i denotes ordinal number of the integration interval while $\text{angle}6_{\text{nom}}$ and $\text{angle}11_{\text{nom}}$ are the angle values for nominal motion. Of course, to preserve position of the foot relative to the ground, ankle joints 4 and 13 should be corrected accordingly.

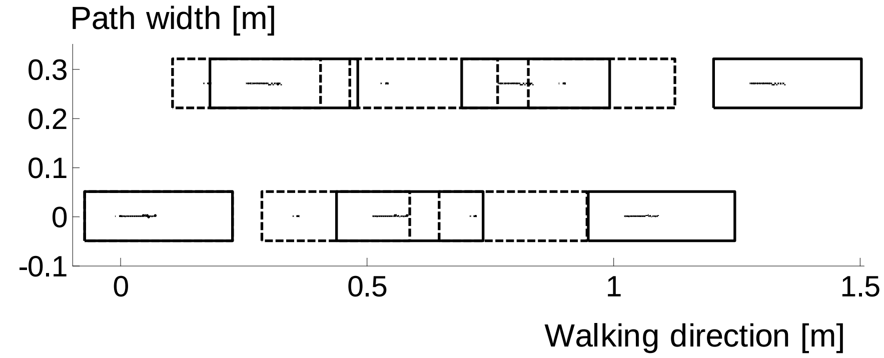


Figure 39. Extended step

In Fig. 39 are given footprints for such extended gait. Dashed line denotes the footsteps of the nominal gait, solid line of the enlarged stride. It can be noticed that the ZMP position deviates along the footprint axis and, actually, no correction is needed.

4. Combination of Basic Modifications of the Nominal Gait

4.1 Step Extension and Speed-Up for Rectilinear and Turning Motion

In the following section we will show how the basic modifications already described can be combined simultaneously.

In Figs. 40-47 are given examples of the combination of basic modifications for rectilinear motion: gait speed-up and step extension (the same as in the example shown in Fig. 39). In Figs. 40, 42, 44 and 46 are given examples of non-compensated and in Figs. 41, 43, 45 and 47 of compensated motion. Step extension was combined with gait speed-up, for c = 1.1, c = 1.2, c = 1.3 and c = 1.4.

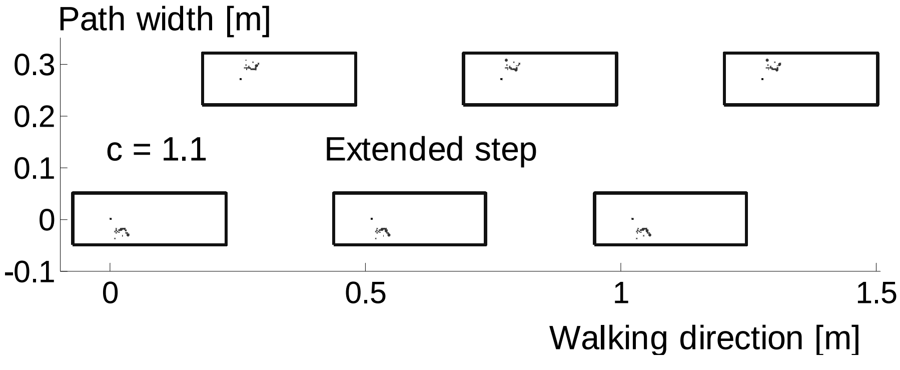


Figure 40. Mechanism footprints of extended step and accelerated motion for *c* = 1.1. No compensation applied.

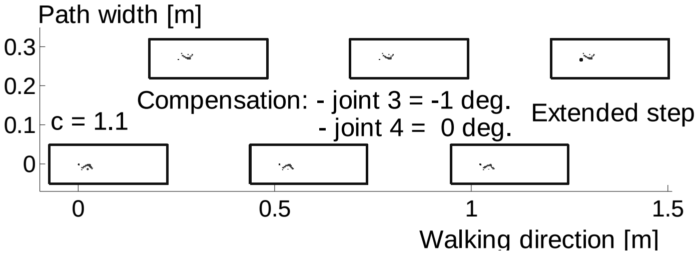


Figure 41. Mechanism footprints of extended step and accelerated motion for c = 1.1. Compensation of –1° applied at joint 3, of 0° at joint 4.

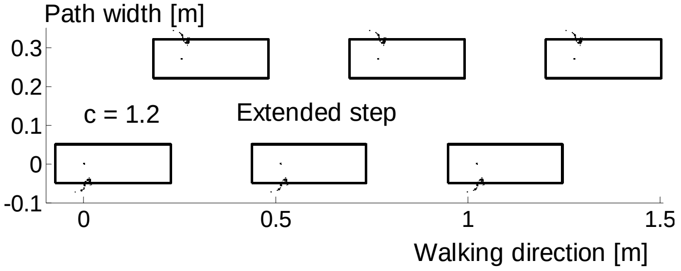


Figure 42. Mechanism footprints of extended step and accelerated motion for c = 1.2. No compensation applied.

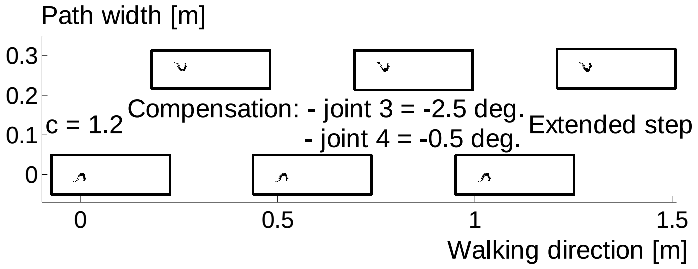


Figure 43. Mechanism footprints of extended step and accelerated motion for c = 1.2. Compensation of –2.5° applied at joint 3, of –0.5° at joint 4.

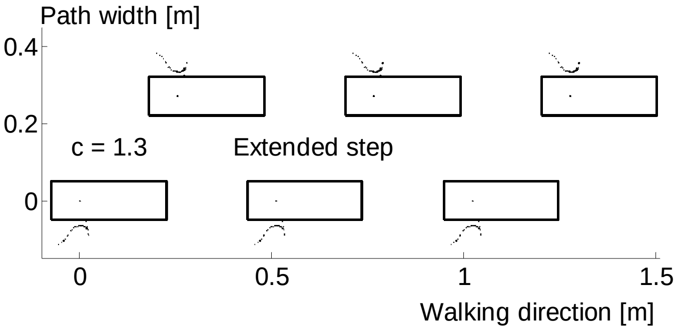


Figure 44. Mechanism footprints of extended step and accelerated motion for c = 1.3. No compensation applied.

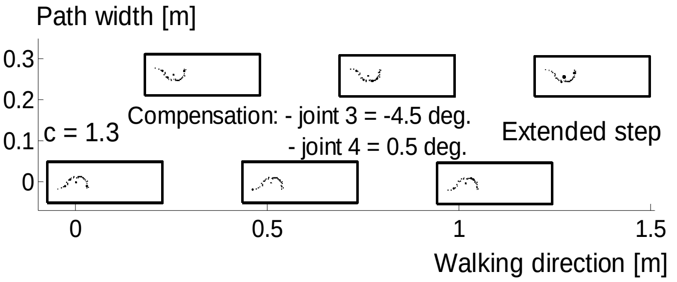


Figure 45. Mechanism footprints of extended step and accelerated motion for c = 1.3. Compensation of –4.5° applied at joint 3, of 0.5 ° at joint 4.

It is again demonstrated that simple inclination at the appropriately selected joint (the ankle joint is again used for compensation) can ensure dynamic balance in spite of the ZMP displacement, even in the case of such step extension as shown in Fig. 46.

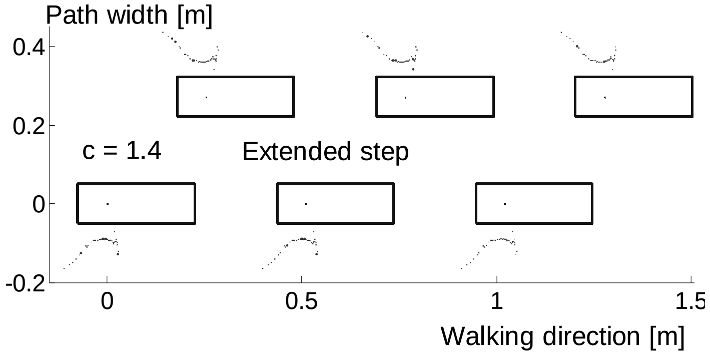


Figure 46. Mechanism footprints of extended step and accelerated motion for c = 1.4. No compensation applied.

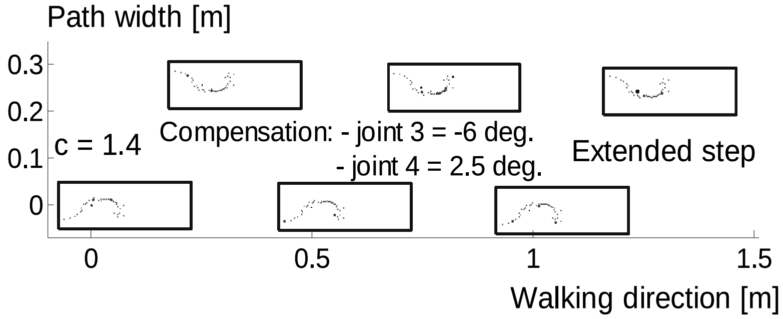


Figure 47. Mechanism footprints of extended step and accelerated motion for c = 1.4. Compensation of –6 ° applied at joint 3, of 2.5 ° at joint 4.

Let us investigate now the influence of the combination of combined modifications of the basic nominal. Once more, the nominal position of the ZMP is the point just below the ankle joint and any deviation from it is easily visible. Mechanism will preserve its dynamic balance (it will not fall down) if the ZMP position is within the support polygon, which in the considered examples corresponds to the footprint area. In Figs. 48-50 are shown examples of the combination of extended step and turning left with different intensities of turning without ZMP displacement compensation because the ZMP is still within the support polygon. In Figs. 51 and 53, to gait extension and turning left, accelerations of c =1.2 and c =1.3 are added.

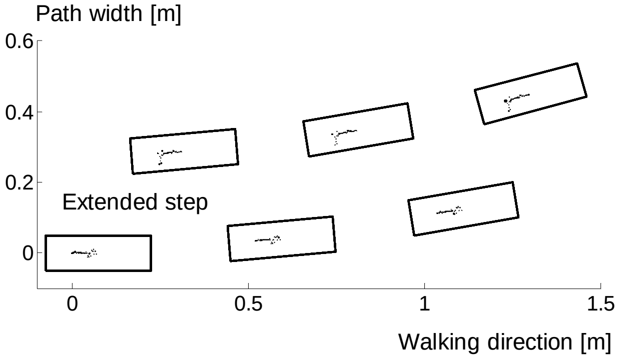


Figure 48. Mechanism footprints of extended step in turning left when α is 5°. No compensation applied.

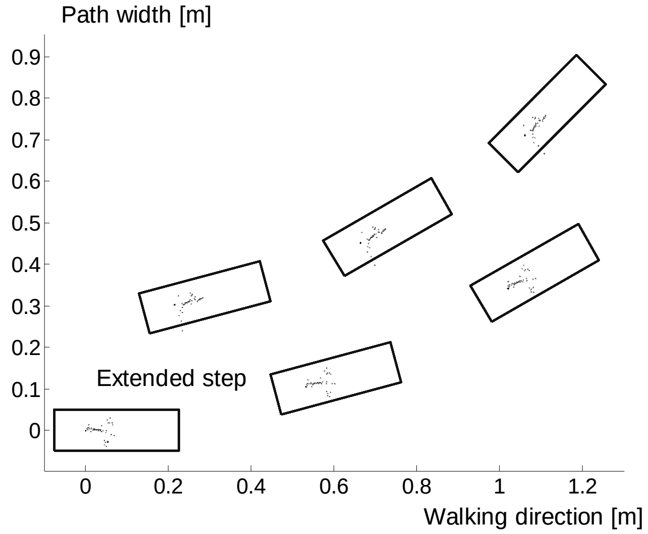


Figure 49. Mechanism footprints of extended step in turning left when α is 15°. No compensation applied.

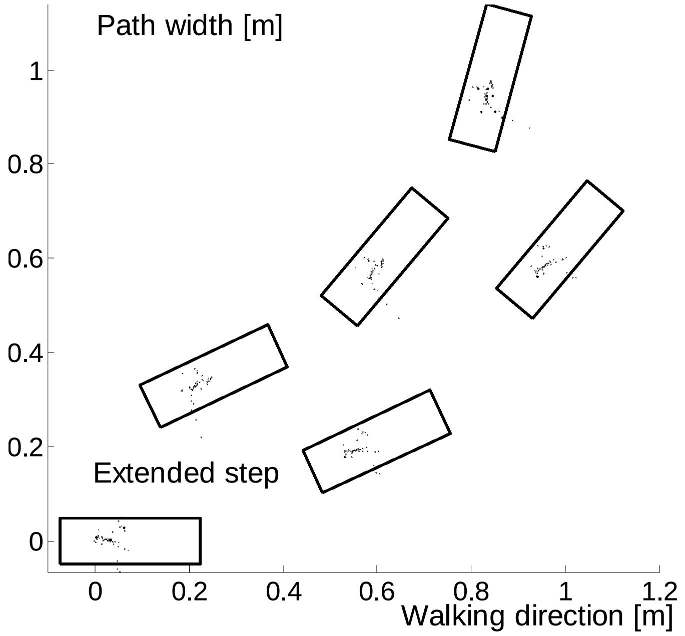


Figure 50. Mechanism footprints of extended step in turning left when α is 25°. No compensation applied.

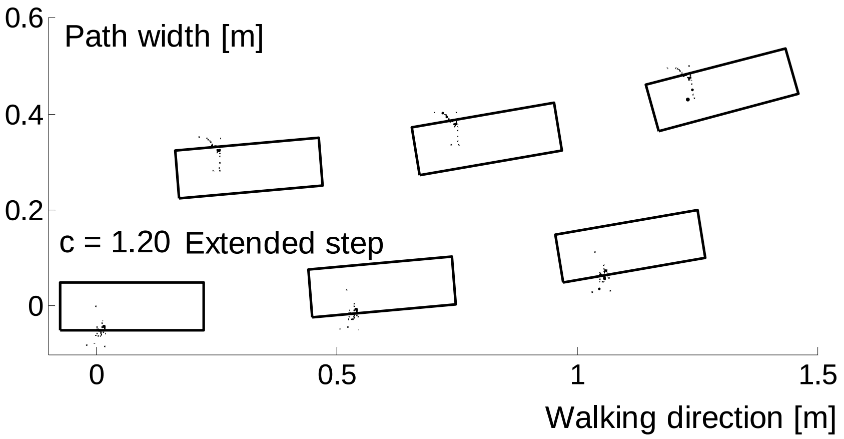


Figure 51. Mechanism footprints of extended step in turning left when α is 5° and accelerated motion for c =1.2. No compensation applied.

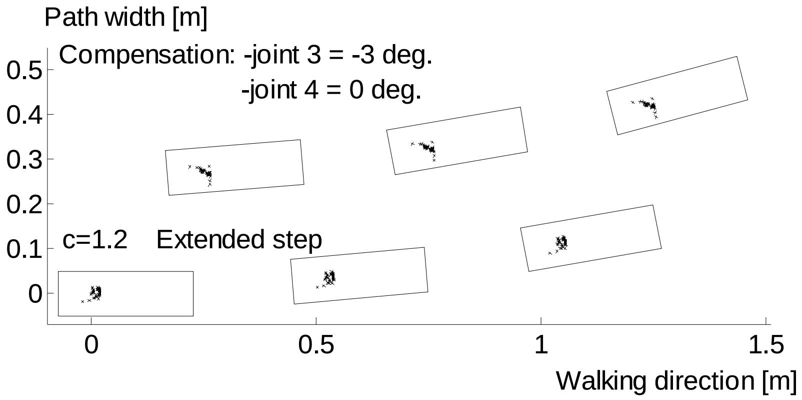


Figure 52. Mechanism footprints of extended step in turning left when α is 5° and accelerated motion for c =1.2. Compensation of –3° applied at joint 3, of 0° at joint 4.

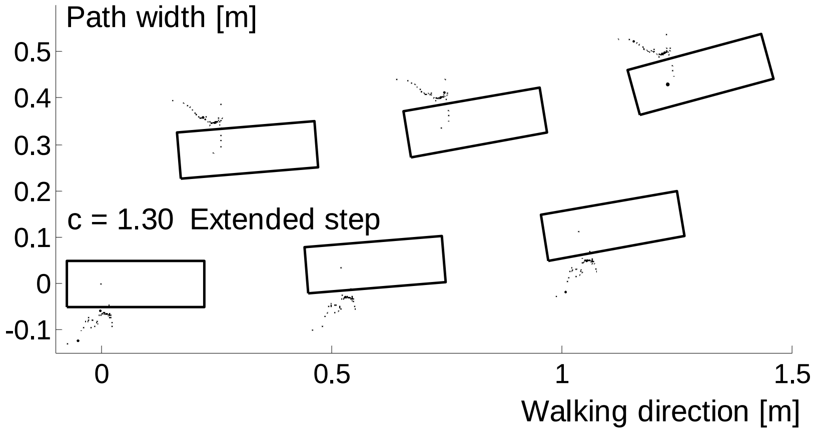


Figure 53. Mechanism footprints of extended step in turning left when α is 5° and accelerated motion for c = 1.3. No compensation applied.

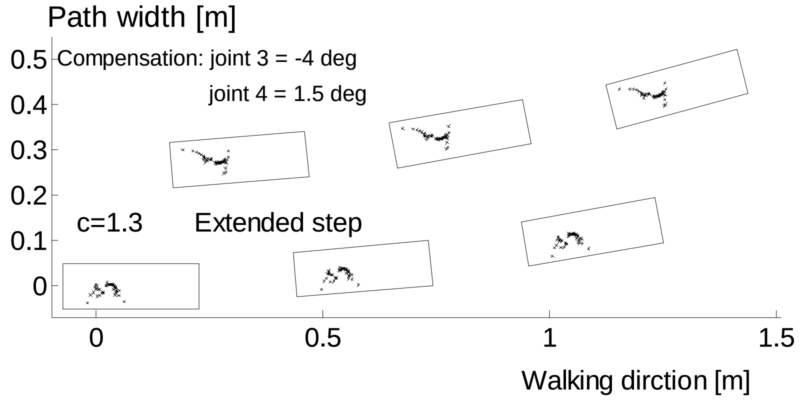


Figure 54. Mechanism footprints of extended step in turning left when α is 5° and acc-elerated motion for c = 1.3. Compensation of –4° applied at joint 3, of 1.5° at joint 4.

In Figs. 52 and 54 is shown the situation when the same compensation strategy was applied again. Ankle joint (joints 3 and 4) was employed for compensation in both the frontal and saggital planes. It is demonstrated again that such simple compensation strategy works well in this case too.

5. Conclusions

The future will bring about the growing application of humanoid robots, which will be increasingly more engaged in the close living and working environment of humans. This fact inevitably leads to sharing their common working environment and the need of "working coexistence" of man and robot. Besides, the human environ-ment, which is highly adapted to humans, is not static but a very dynamic one, so conventional procedures of planning motion trajectories will not be applicable for biped locomotion. Hence, the motion trajectories will have to be determined on the basis of a global plan of motion and instantaneous situation on the scene. Therefore, a new approach to planning biped gait is needed, which will be characterized by the possibility of fast reacting to the newly arisen situations by modifying the planned system motion, while constantly preserving its dynamic balance.

The paper describes a new approach to solving this task. One gait pattern is adopted as the basic one, and all the other gait realizations are obtained by its modification. The basic gait pattern adopted was the dynamically balanced walk (obtained by semi-inverse method) on a flat ground along a straight line, considered as a nominal gait. By appropriate modifications of the nominal gait at particular joints we achieved turning, speed-up, and slow-down motion and a combination of these basic manoeuvres. As a consequence of the modification of the basic pattern, the ZMP position changes so that even the dynamic balance of the overall system is endangered. We showed that the deviations of the ZMP position could be to a large extent compensated for simply by appropriate inclining the body. However, this inclination can jeopardize the anthropomorphism of of the locomotion system, so that it is very important to select the appropriate joint for the modification realization, that is joint that will yield the dynamic balance with the least disturbance of the system anthropomorphism. Hence, this issue has received the special attention in the presented study.

Acknowledgement

This work was funded by the Ministry for Technology and Development of the Republic of Serbia under contract MIS 3.04.0019.A/1.

6. References

Hollerbach, J., M., 1984, "Dynamic Scaling of Manipulator Trajectories", Trans. of the ASME, Vol. 106, pp. 102-106

Vukobratovic M. & Juricic D. (1968), Contribution to the Synthesis of Biped Gait, Proc. of IFAC Symp. on Technical and Biological Problem on Control, Erevan, USSR,

Vukobratovic M. & Juricic D. (1969), Contribution to the Synthesis of Biped Gait, IEEE Transaction on Bio-Medical Engineering, , Vol. 16, No. 1.

Vukobratovic M. & Stepanenko Yu., (1972), "On the Stability of Anthropomorphic Systems", Mathematical Biosciences, Vol. 15, pp.1-37.

Vukobratovic M., Borovac B., Surla D., Stokic D., (1990), Biped Locomotion – Dynamics, Stability, Control and Application, Springer-Verlag, Berlin.

Vukobratovic M., Andric D., Borovac B., "How to Achieve Various Gait Patterns from Single Nominal ", International Journal of Advanced Robotic Systems , Vol. 1., No. 2, Page 99-108, 2004

-IX-

Robot Manipulators

Trajectory Planning of a Constrained Flexible Manipulator

Atef A. Ata & Habib Johar

1. Introduction

Many of today's flexible robots are required to perform tasks that can be considered as constrained motions. A robot is considered to be in constrained maneuvering when its end-effector contacts and/or interacts with the environment as the robot moves. These applications include grinding, deburring, cutting, polishing, and the list continues. For the last two decades many researchers have investigated the constrained motion problem through different techniques such as hybrid position/force control (Raibert, M. & Craig, J., 1981 and Shutter et al., 1996), force control (Su, et al., 1990) , nonlinear modified Corless-Leitman controller (Hu, F. & Ulsoy, G., 1994), two-time scale force position control (Rocco, P. & Book, W., 1996), multi-variable controller (Shi, et al., 1990), parallel force and position control (Siciliano & Villani, L.,2000) , and multi-time-scale fuzzy logic controller (Lin, J., 2003). Luca et al. (Luca, D. & Giovanni, G., 2001) introduced rest-to-rest motion of a two-link robot with a flexible forearm. The basic idea is to design a set of two outputs with respect to which the system has no zero dynamics. Ata and Habib have investigated the open loop performance of a constrained flexible manipulator in contact with a rotary surface (Ata, A. & Habib, J., 2004) and arbitrary shaped fixed surfaces (Ata, A. & Habib, J., 2004). The effect of the contact force on the required joint torques has been simulated for different contact surfaces. Despite the voluminous research done, the study of dynamics and control of the constrained motion of the flexible manipulators remains open for further investigations.

The present study aims at investigating the dynamic performance of a constrained rigid/flexible manipulator. The Cartesian path of the end-effector is to be converted into joint trajectories at intermediate points (knots). The trajectory segments between knots are then interpolated by cubic-splines to ensure the smoothness of the trajectory for each joint. To avoid separation between the end-effector and the constraint surface and to minimize the contact surface as well, these knots are chosen at the contact surface itself. Simulation results have been carried out for three contact surfaces.

2. Dynamic Modeling

Consider the two-link planar manipulator shown in Figure (1). The first link is rigid while the second link is flexible and carries a tip mass m3. This tip mass always moves in contact with a constrained surface. The mass density of the flexible link is ρ_2 and Euler-Bernoulli's beam theory is employed to describe the flexural motion of the flexible link. The manipulator moves in the horizontal plane so that the effect of gravity is not considered. The flexible link is described by the Virtual Link Coordinates System (VLCS) in which the deformation is measured relative to the link that joins the end-points and the angle θ is measured with respect to the horizontal axis (Benati, M. & Morro, A., 1994).

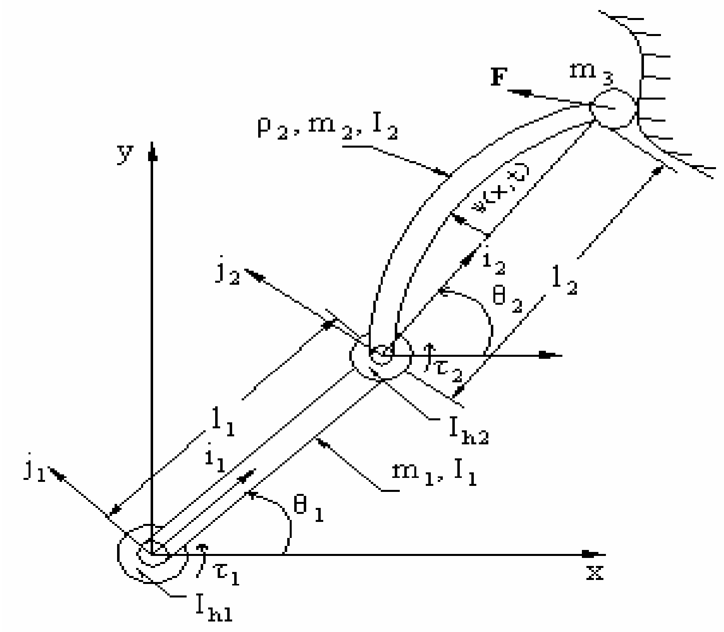


Figure 1. A two link rigid-flexible manipulator with tip mass

In general, the dynamic equation of motion for a rigid-flexible robot can be written as:

$$\tau = M(\theta)\ddot{\theta} + V(\theta, \dot{\theta}) + \tau_r \tag{1}$$

where Θ is *nx1* vector of joint positions; $\dot{\Theta}$ is *nx1* vector of joint velocities; $\ddot{\Theta}$ is *nx1* vector of joint accelerations; *M(*Θ*)* is the *nxn* inertia matrix; *V(*$\Theta, \dot{\Theta}$*)* is the *nx1* Coriolis and centripetal terms of the manipulator; τ is the *nx1* vector of joint torques, *n* is the number of degrees of freedom. Based on the dynamical model of an open chain multi-link flexible manipulator using the extended Hamilton's principle [Ata, A. & Habib, J., 2004 and Benati, M. & Morro, A., 1994], the elements of the inertia and Coriolis matrices of the rigid/flexible manipulator considered here are shown on the appendix .

τ_r is the *nx1* interaction joint torques due to the force exerted at the end-effector and is given by:

$$\tau_r = J^T F = J^T \begin{bmatrix} F_x \\ F_y \end{bmatrix}$$

where J^T is the transpose Jacobian of the system.

F_x and F_y represent the applied forces by the end-effector on the constrained surface and it includes two parts. The first component is due to the difference between the proposed coordinates of the end-effector in free motion state and the corresponding point in the contact surface and can be modeled as mass-spring system (Raibert, M. & Craig, J., 1981). The second part arises from the inertia of the tip mass while maneuvering. Then the components of the contact force can be given as:

$$F_x = K_s(r_x - l_1\cos\theta_1 - l_2\cos\theta_2) + m_3(l_1\ddot{\theta}_1\sin\theta_1 \\ + l_1\dot{\theta}_1^2\cos\theta_1 + l_2\ddot{\theta}_2\sin\theta_2 + l_2\dot{\theta}_2^2\cos\theta_2) \tag{2}$$

$$F_y = K_s(r_y - l_1\sin\theta_1 - l_2\sin\theta_2) + m_3(l_1\dot{\theta}_1^2\sin\theta_1 \\ - l_1\ddot{\theta}_1\cos\theta_1 + l_2\dot{\theta}_2^2\sin\theta_2 - l_2\ddot{\theta}_2\cos\theta_2) \tag{3}$$

Where K_S is the spring stiffness (N/m), r_x, and r_y are the coordinates of the contact point. For the second link, the equation due to the flexibility effect can be written as:

$$\rho_2\{w\dot{\theta}_2^2 - \ddot{w} - x\ddot{\theta}_2 - l_1[\dot{\theta}_1^2\sin(\theta_2 - \theta_1) + \ddot{\theta}_1\cos(\theta_2 - \theta_1)]\} - E_2I_2w'''' = 0 \tag{4}$$

where E_2I_2 is the flexural stiffness of the flexible link. Equation (4) describes the vibration of the flexible link subject to four boundary conditions:

$$w(0,t) = 0 \tag{5a}$$

$$w(l_2,t) = 0 \tag{5b}$$

$$w''(0,t) = -\frac{\tau_2(t)}{E_2I_2} \tag{5c}$$

$$w''(l_2,t) = 0 \tag{5d}$$

3. Inverse Dynamics Analysis

By introducing a new variable *y(x,t)* which represents the total deflection of the flexible link as:

$$y(x,t) = w(x,t) + x\theta_2(t) \tag{6}$$

By ignoring the first nonlinear term of equation (4), since its effect is only obvious at very high speed, then equation (4) can be written in terms of *y(x,t)* as:

$$\rho_2\{-\ddot{y} - l_1[\dot{\theta}_1^2\sin(\theta_2 - \theta_1) + \ddot{\theta}_1\cos(\theta_2 - \theta_1)]\} - E_2I_2y'''' = 0 \tag{7}$$

The modified boundary conditions are:

$$y(0,t) = 0 \tag{8a}$$

$$y(l_2,t) = l_2\theta_2(t) \tag{8b}$$

$$y''(0,t) = -\frac{\tau_2(t)}{E_2I_2} \tag{8c}$$

$$y''(l_2,t) = 0 \tag{8d}$$

Solving equation (7) analytically subject to the time-dependent inhomogeneous boundary conditions (8b and c) is quite a difficult task since the flexible torque $\tau_2(t)$ that should be obtained is already included in the boundary conditions for the total deflection *y(x,t)*. To avoid this difficulty, $\tau_2(t)$ is assumed as the rigid torque without any elastic effect. The boundary condition (8c) can then be evaluated and equations (7) and (6) can be solved to get *w(x,t)*. Finally upon substitution into equation (1) one can get the flexible torque of the second link. Using the assumed mode method, the solution for *y(x,t)* can be assumed in the form (Meirovitch, L., 1967 and Low, K., 1989):

$$y_n(x,t) = \overset{\ddagger}{\underset{n=1}{\ddagger}}{}^{\prime}\![\; v_n(x)\zeta_n(t)\;] + g\;(x)e\;(t) + h\;(x)f\;(t) \tag{9}$$

where:

$$e(t) = l_2\theta_2(t) \tag{10a}$$

$$f\;(t) = -\frac{\tau_2(t)}{E_2I_2} \tag{10b}$$

g(x) and *h(x)* are functions of the spatial coordinate alone to satisfy the homogeneous boundary conditions for $v_n(x)$ and they are given by:

$$g\;(x) = x \tag{10d}$$

$$h\;(x) = -\frac{1}{3}l_2x + \frac{1}{2}x^2 - \frac{1}{6l_2}x^3 \tag{10e}$$

and $\zeta_n(t)$ is the time function. The subscript n represents the number of modes taking into considerations. In this study, five modes of vibration of the flexible link have been taken into consideration. The corresponding eigenvalue problem for equation (7) can be written:

$$k_2\frac{d^4v(x)}{dx^4} + \omega^2\rho v(x) = 0 \tag{11}$$

Then $v_n(x)$ can be obtained using modal analysis in the form (Clough, R. & Penzien, J., 1993):

$$v_n(x) = \sqrt{\frac{2}{\rho l_2}}\sin\frac{n\pi}{l_2}x \quad , \quad n = 1,2,3.... \tag{12}$$

The last two nonlinear terms inside the parentheses in equation (6) can be regarded as distributed excitation force with unit density. This effect can be compensated for the time function as (Meirovitch, L., 1967 and Low, K., 1989):

$$\zeta_n(t) = \frac{1}{\omega_n}\int_0^t N_n\tau\sin(t-\tau)d\tau \tag{13}$$

The convolution integral in equation (13) can be evaluated using Duhamel integral method (Clough, R. & Penzien, J., 1993). Then, the rigid and flexible torques incorporated with vibration effects can be obtained from equations (1).

4. Model Verification

To verify the proposed dynamical model a simulation of the joint torques through the solution of the inverse dynamics problem has been carried out for different values of the hub to beam inertia ratio (IR). The inertia ratio is defined as the percentage of the actuator's inertia to the link inertia that, i. e., inertia ratio = (Hub inertia / Link inertia). The joint profiles are assumed as a sine rest-to-rest maneuver for both links as:

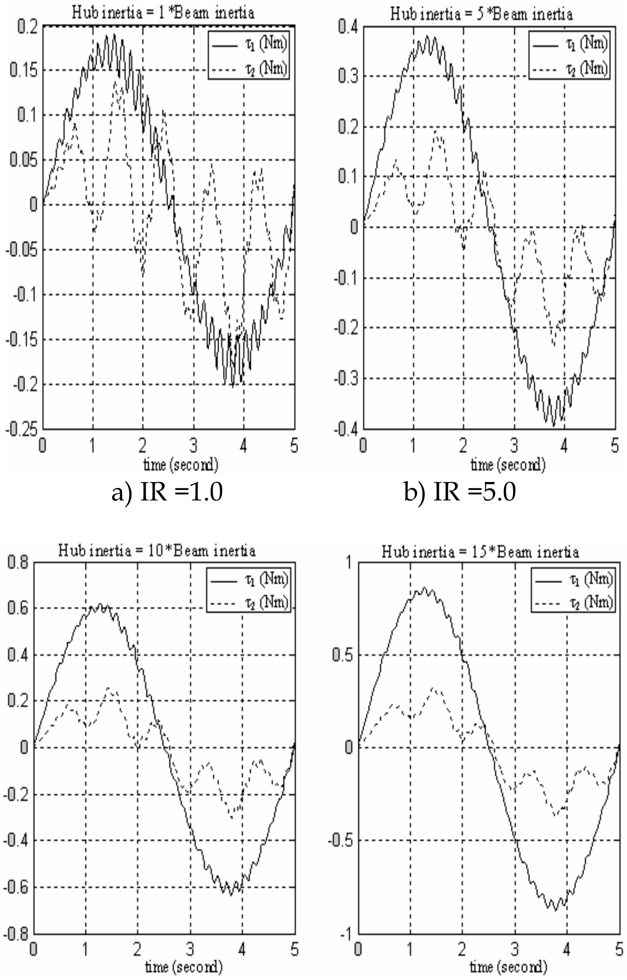
$$\theta_n = \frac{\theta_{nd}}{T}(t - \frac{T}{2\pi}\sin(\frac{2\pi T}{T})),$$

where $\Theta_{nd} = 1$ rad is the desired final angle and T=5 sec. is the duration time. The system parameters used in this study (Luca, D. & Giovanni, G., 2001) are summarized in Table (1).

	Rigid	Flexible
Length (m)	l_1=0.5	l_2=0.75
Mass density (kg/m)		ρ_2=0.5
Mass (kg)	m_1=0.5	$m_2=L_2*\rho_2$
Inertia (kgm^2)	I_1=0.0834	$I_2=(l_2^3*\rho_2/3)$
Hub inertia (kgm^2)	$10I_1$	$10I_2$
Flexural rigidity (Nm^2)		E_2I_2 =2.4507
Tip mass	0.15 Kg	

Table 1. System parameters

Four different inertia ratios have been used to simulate and compare the joint torque profiles as shown in Figure 2(a-d) for unconstrained motion. The end-effector is maneuvering freely in space and thus, no contact force is included in joint torque simulation.



a) IR =1.0 b) IR =5.0

c) IR =10.0 d) IR =15.0

Figure 2(a-d). Joint torque profiles for different inertia ratios

It is clear from Figure 2(a-d) that fluctuations arise in joint torques due to the vibrations of flexible link while the manipulator is maneuvering.

As the inertia ratio increases, the magnitude of the joint torques increases and the amplitudes of fluctuations decrease. It should be noticed that the amplitudes of fluctuations in joint torques resulting from inertia ratio of 10 have significantly diminished as compared to those of inertia ratio of 1 and 5. This implies that vibration effects on joint torques can be damped to certain extent with a good selection of inertia ratio. The situation is even better for an inertia ratio of 15 where the fluctuations are very much significantly reduced but the price is a bigger actuator size. However, since we desire to design light weight manipulator, inertia ratio of 10 is more preferable to that of 15. This conforms with the stability analysis of stable and asymptotically stable flexible manipulator (Ata, et al., 1996).

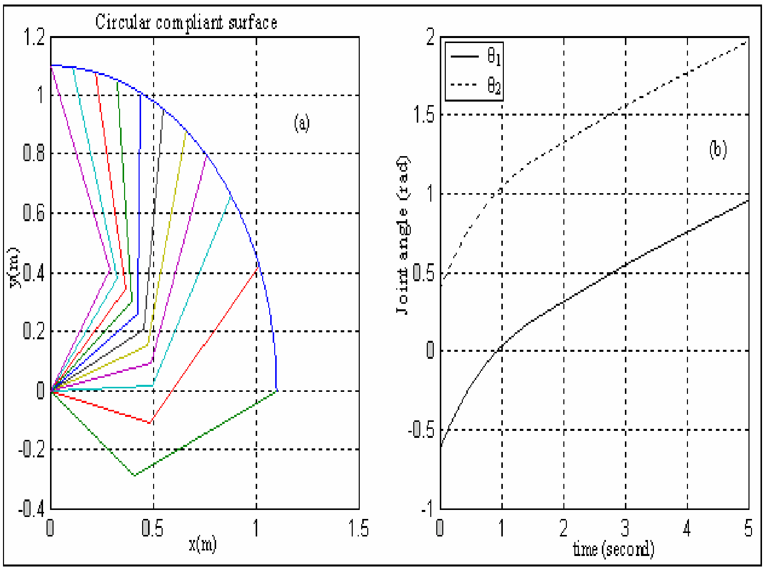
5. Cubic-Spline Trajectory Interpolations

Generally, the trajectory of the robot arm is specified in the task space, while the robot is controlled in the joint space. Using inverse kinematics and Jacobian, the desired path and velocity can be converted to joint trajectories and velocities.

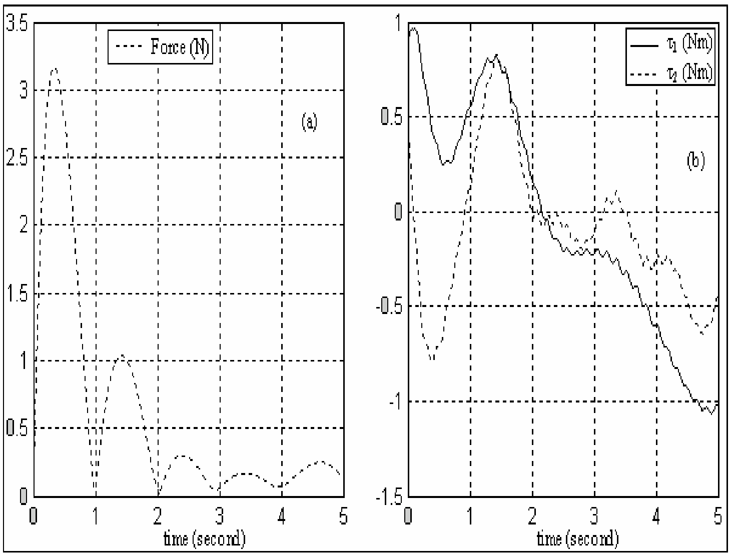
Among the interpolation path planning methods, the piecewise cubic and B-spline polynomial joint paths are often adopted due to their easy and fast mathematic manipulation.

Unfortunately, a B-spline does not go through the knots of the path though they are suitable for on-line computation. In case of cubic-spline trajectory interpolation, joint angles are well calculated beforehand for certain points on the surface as such the end effector strictly remains in contact with the surface at those particular points while maneuvering (Cao, et al., 1994).

Then, the obtained joint angles through the solution of the inverse kinematics problem are interpolated using cubic-spline for entire surface such that the end effector closely maneuvers along the surface. To study the variation of the contact force and its effect on the joint torques, three different compliant surfaces are applied here: circular, inclined, and vertical. For each surface, the force exerted at the end-effector and the required joint torque for both links are plotted in Figures 3,4, and 5 respectively.

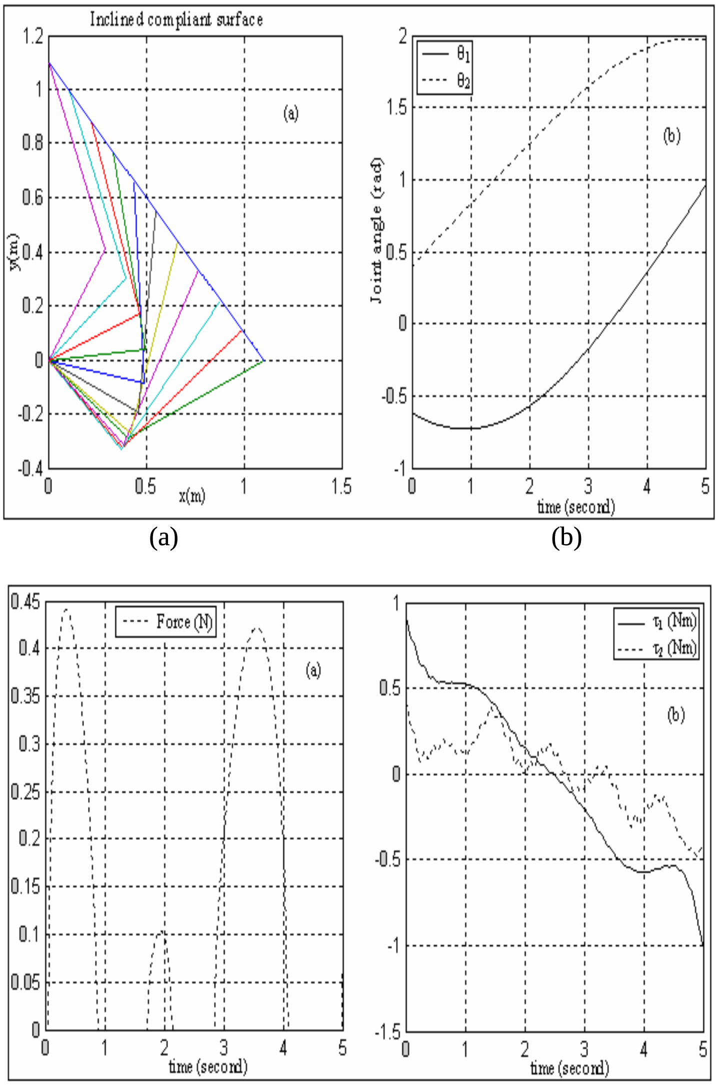


(a) (b)



(c) (d)

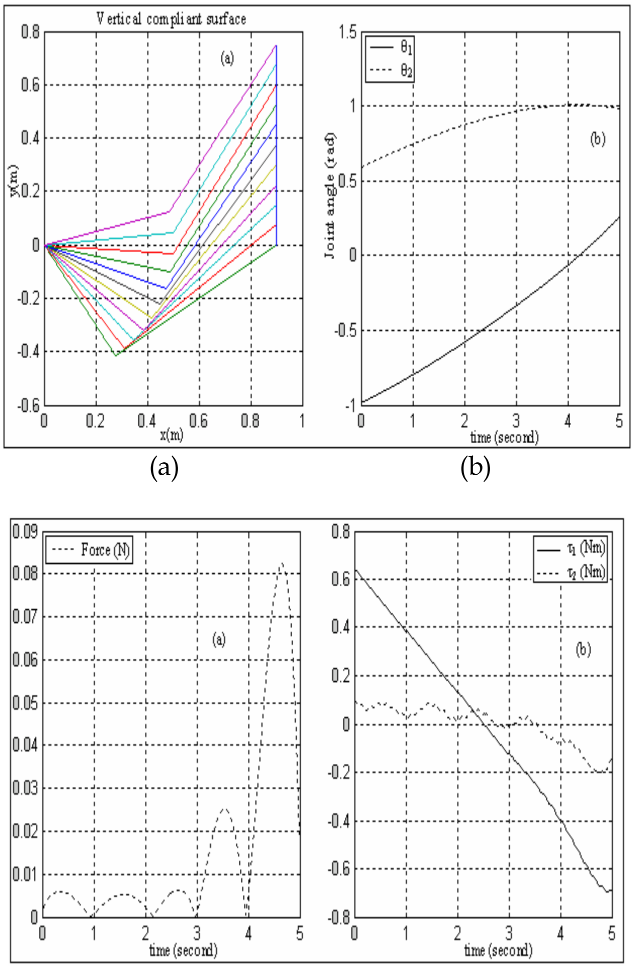
Figure 3. Circular surface: (a) End-effector trajectory, (b) Joint angles (c) Force distribution (d) Joint torques profiles



(a) (b)

(c) (d)

Figure 4. Inclined Surface: (a) End-effector trajectory, (b) Joint angles (c) Force distribution (d) Joint torques profiles



(a) (b)

(c) (d)

Figure 5. Vertical surface: (a) End-effector trajectory, (b) Joint angles (c) Force distribution (d) Joint torques profiles

Simulation of force distribution in Figures 3(c), 4(c) and 5(c) verifies that designing the joint trajectories based on cubic-splines ensures that the end effector closely maneuvers along the desired trajectory. However, in the case of circular contact surface, the initial force distribution in Figure 3(a) is relatively high within the time interval of 0 to 1 second and then gradually reduces as the end effector moves along the surface. This force is expected to be small since the end-effector is controlled to pass closely to the constraint surface. This implies that the cubic-spline interpolation technique may have a good command on the end-effector to maneuver closely along the entire contact surface only for certain shapes of surface. Perhaps for complicated shapes of contact surface, the cubic-spline interpolation technique may not work well to compel the end effector to maneuver closely along the entire region of the contact surface.

The simulation of the joint torque profiles in Figures 3(b), 4(b) and 5(b) shows that irregular fluctuations can appear in the joint torque profiles due to the effect of cubic-spline interpolation.

6. Conclusions

Constrained motion of a planar rigid-flexible manipulator is considered in this study. a dynamic model with zero tip deformation constraint of constrained rigid-flexible manipulator moving in horizontal plane is derived. A systematic approach of analytic solution to the fourth order differential equation with time dependent, nonhomogeneous boundary conditions including the nonlinear terms is presented to compute the elastic deflection w(x,t) of the flexible link while maneuvering. The joint motion profiles have been designed based on the cubic-splines interpolation to ensure that the end-effector will move along the constrained surface without separation. The proper hub to beam inertia ratio of 10 has been selected based on a comparison study

7. Appendix

Equations of motion

$$M_{11} = (I_1 + I_{h1} + m_2 l_1^2 + m_3 l_1^2)$$

$$M_{12} = (m_2 l_1 \bar{x}_2 + m_3 l_1 l_2)\cos(\theta_2 - \theta_1)$$

$$- \int_0^{l_2} \rho l_1 w \sin(\Theta_2 - \Theta_1) dx$$

$$M_{21} = M_{12}$$

$$M_{22} = (I_2 + I_{h2} + I_3 + m_3 l_2^2) + \int_0^{l_2} \rho w^2 dx$$

$$v_{11} = -(m_2 l_1 \bar{x}_2 + m_3 l_1 l_2)\dot{\theta}_2^2 \sin(\theta_2 - \theta_1) + \int_0^{l_2} \rho_2 l_1 [(\ddot{w} - w\dot{\theta}_2^2)\cos(\theta_2 - \theta_1) - (2\dot{w}\dot{\theta}_2)\sin(\theta_2 - \theta_1)] dx$$

$$v_{21} = (m_2 l_1 \bar{x}_2 + m_3 l_1 l_2)\dot{\theta}_1^2 \sin(\theta_2 - \theta_1) + \int_0^{l_2} \rho_2 \{-2w\dot{w}\dot{\theta}_2 + x\ddot{w} + l_1 w \dot{\theta}_1^2 \cos(\theta_2 - \theta_1)\} dx$$

where $m_2 \bar{x}_2 = \int_0^{l_2} \rho_2 x dx$ and I_n and I_{hn} are the inertia of the links and actuators respectively.

$$J^T = \begin{pmatrix} -l_1 \sin\theta_1 - l_2 \sin\theta_2 & l_1 \cos\theta_1 + l_2 \cos\theta_2 \\ -l_2 \sin\theta_2 & l_2 \cos\theta_2 \end{pmatrix}$$

8. References

Atef A. Ata & Habib Johar (2004) " Dynamic Analysis of a Rigid-Flexible Manipulator constrained by Arbitrary Shaped Surfaces", International Journal of Modelling and Simulation, To appear soon.

Atef A. Ata & Habib Johar (2004) "Dynamic simulation of task constrained of a rigid-flexible manipulator", International Journal of Advanced Robotic Systems, Vol. 1, No. 2, June 2004, pp. 61-66.

Atef, A. Ata , Salwa M. Elkhoga, Mohamed A. Shalaby, & Shihab S. Asfour (1996) " Causal inverse dynamics of a flexible hub- arm system through Liapunov second method" Robotica Vol. 14, pp 381-389.

Benati, M. & Morro, A. (1994). Formulation of equations of motion for a chain of flexible links using Hamilton's principle. ASME J. Dynamic System, Measurement and Control , 116, pp. 81-88.

Cao, B., Dodds, G. I., & Irwin, G. W.(1994). Time Optimal and Smooth Constrained Path Planning for Robot Manipulators. Proceeding of IEEE International Conference on Robotics and Automation, pp. 1853-1858.

Clough, R.W. & Penzien, J. (1993). Dynamics of Structures, McGraw-Hill, Inc.,.

Hu, F. L. & Ulsoy, G. (1994). Force and motion control of a constrained flexible robot arm. ASME, J. Dynamic System, Measurement and Control, 116, pp. 336-343.

Lin, J. (2003). Hierarchical fuzzy logic controller for a flexible link robot arm performing constrained motion task, IEE Proceedings: Control Theory and Applications, Vol. 150, No. 4, , pp. 355-364.

Low, K. H.(1989). Solution schemes for the system equations of flexible robots. J. Robotic Systems, 6, pp. 383 -405.

Luca, D. A. & Giovanni, D. G. (2001). Rest-to-Rest motion of a two-link robot with a flexible forearm. IEEE/ASME Int. Conf. on Advanced Intelligent Mechatronics, pp. 929-935.

Meriovitch, L. (1967). Analytical Methods in Vibrations. The Macmillan Co., New York.

Raibert, M. H. & Craig ,J. J. (1981). Hybrid position / force control of manipulator, ASME J. of Dyanmics Systems, Measurement and Control, 102, pp. 126-133.

Rocco P. P & Book, W. J. (1996). Modeling for two-time scale force/position control of flexible arm. IEEE Proc. International conference on Robotics and Automation, pp. 1941-1946.

Schutter , J. D., Torfs, D. & Bruyninckx , H. (1996). Robot force control with an actively damped flexible end-effector. J. Robotics and Autonomous System, 19, pp. 205-214.

Shi, Z. X., Fung, E. H. K., and Li, Y. C. (1999). Dynamic modeling of a rigid-flexible manipulator for constrained motion task control. J. Applied Mathematical Modeling 23, pp. 509-529.

Siciliano & Villani, L. (2000). Parallel force and position control of flexible manipulators. IEE Proc-Control Theory Appl. 147, pp. 605 -612.

Su, H. J., Choi, B. O. & Krishnamurty , K. (1990). Force control of high-speed, lightweight robotic manipulator. Mathl. Comput. Modeling 14, pp. 474 -479.

Position/Force Hybrid Control of a Manipulator with a Flexible Tool Using Visual and Force Information

Jian Huang, Isao Todo & Tetsuro Yabuta

1. Introduction

1.1 Background

Robots used for industrial applications such as welding, painting and object handling have been common for many years. In recent years, the development of domestic robots has become more and more important because of the large and growing population of aged people, especially in Japan. To assist people in their daily lives, a robot must have the ability to deal with not only rigid objects but also the deformable and fragile objects usually encountered in our daily life. Many control algorithms have been developed for the manipulation of rigid objects, and in the recent past many studies related to robotic manipulation of deformable objects have also been reported (Hirai, 1998).

1.2 Recent Researches on Manipulating Deformable Objects

In recent years, the robotic manipulation of flexible objects has been demonstrated in a variety of applications. Manipulation of a deformable tube based on human demonstration (Hirai & Noguchi, 1997), wire manipulation (Nakagaki et al., 1997, Nakagaki, 1998), a study of manipulating a linear objects (Remde et al., 1999, Acker & Henrich, 2003, Schlechter & Henrich, 2002, Schmidt & Henrich, 2001, Yue & Henrich, 2002), one-handed knotting manipulation (Wakamatsu et al., 2002, 2004), handling of fabric objects (Ono, 1998), ticket handling (Itakura, 1998) and contact task on a flexible plate (Wu, et. al., 1997) have been developed,.

In general, deformable objects display a wide range of responses to applied forces because of their different physical properties. Therefore, different control strategies are required for a robot to manipulate different kind of objects. Deformation model analysis is one of the fundamental methods available. Trials have been made of methods based on establishing a deformation model for robot manipulation (Wakamatsu & Wada, 1998, Hisada, 1998), and a state-transition method (Henrich , et. al., 1999, Remde, et al., 1999, Abegg, et al., 2000).

When a human manipulates a deformable object, he will actively combine visual information from his eyes with contact force information obtained by his hands. Vision-based state detection for a linear object (Acker & Henrich, 2003, Abegg, et al., 2000) and visual tracking of a wire deformation (Chen & Zheng, 1992, Nakagaki et al., 1997) have been reported.

Furthermore, a control method based on high-speed visual detection and visual/force sensor fusion was proposed to enable a robot to complete the task of inserting an aluminum peg held by a manipulator into an acrylic hole fixed on a deformable plate (Huang & Todo, 2001). In a continuing study, a stereovision-based compliance control enabling a robot arm to manipulate an unknown deformable beam object (Huang, et al., 2003) has been demonstrated, in which stereovision and force information were combined by an online learning neural network so that the robot could adjust its position and orientation in response to the deformation state of the unknown beam object.

1.3 Description of the Study in this Chapter

A robot with a flexible tool is considered the first choice for manipulating a fragile object. However, in contrast with the extensive literature on robot manipulation of rigid and deformable objects, the study of how to control a robot arm with a flexible tool has been neglected. Therefore, a great need exists to develop fundamental control methods for robots with flexible tools.

When a robot arm with a flexible tool is used to complete a contact task, deformation of the flexible tool always occurs so that the task can not be adequately completed by using solely the type of control methods designed for a robot with a rigid tool. As the position of the tool's tip can not be computed from the robot's kinematics, establishing a deformation model of the flexible tool is generally needed to estimate the position of the flexible tool's tip.

In this chapter, a position/force hybrid control method that incorporates visual information is proposed to enable a robot arm with a flexible tool to complete a contact task. To detect the position of the flexible tool's tip, two CCD cameras are used and a real time image processing algorithm is developed. Furthermore, an online learning neural network is introduced to the position/force hybrid control loop so as to improve the tracing accuracy of the flexible tool. An advantage of the proposed method is that establishing a deformation model of the flexible tool is not necessary.

With the proposed method, a flexible tool held by the end-effector of a 6 DOF robot is used to trace a given curve with a specified vertical pressing force.

2. Overview of the Robot System

The robot control system considered in this chapter includes a manipulator with seven degrees of freedom (PA-10, Mitsubishi Heavy Industry Co.), a force/torque sensor (10/100, B.L. Autotec Ltd.) for force detection, two compact CCD cameras (XC-EI50, Sony Co.) for stereo visual detection and a personal computer (Dimension XPS B733r, Dell Co.) for algorithm calculation. However, the redundant joint of the manipulator is fixed. A schematic diagram of the robot system is shown in Fig.1.

As shown in Fig.1, a cylindrical tube made from polyvinyl chloride (10.7 mm diameter) is used as the tool, and is held by the end-effector of the manipulator. The tip of the flexible tool is wrapped with a light-reflective tape for image detection. A infrared ring type radiators (IRDR-110, Nissin Electronic Co.) is set up in front of each camera's lens.

The required task is for the flexible tool held by the robot to trace a given curve with a specified pressing force. As deformation of the flexible tool always occurs on contact, control of the flexible tool's tip to make it trace a desired trajectory is very difficult.

In order to estimate the tool's tip position, establishing a deformable model is usually considered.

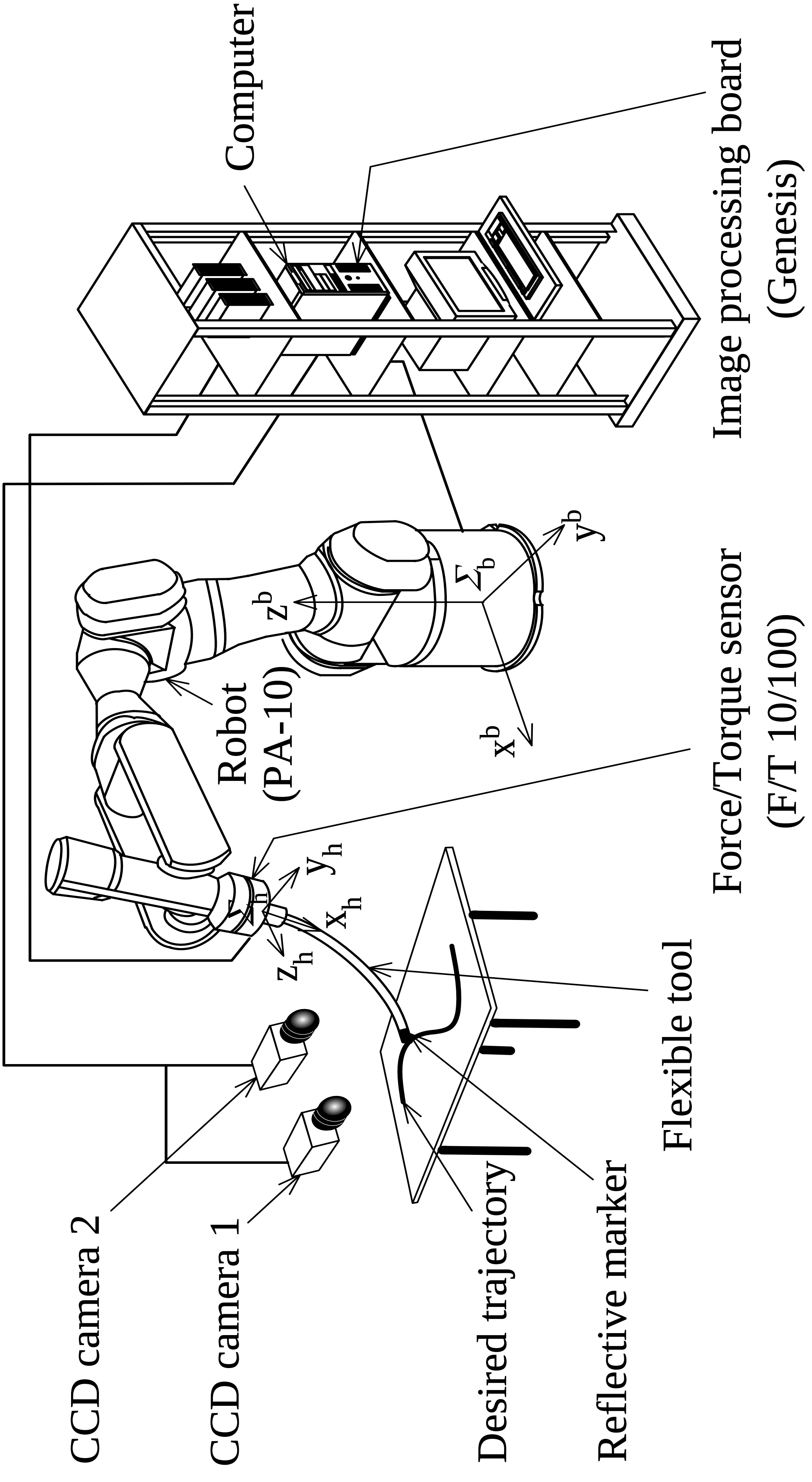


Figure 1.The schematic diagram of the robot system with a flexible tool

Therefore, some physical parameters like the stiffness and Young's modulus of the flexible tool are needed for model analysis. However, such physical parameters are usually unknown for many of the materials encountered in daily life.

3. Method of Image Processing

3.1 Real Time Image Processing

As shown in Fig.1, two synchronized compact cameras are used to construct a stereovision system. Each camera outputs images at the rate of 30 frames per second. For image input and processing, a graphic board with a C80 digital signal processor (GENESIS GEN/F/64/8 /STD, Matrox Co.) is installed in the personal computer.

Generally, real time image processing means that one frame image from a camera must be processed within the camera's frame time so that image input and processing can be carried out simultaneously. Real time image processing is commonly achieved with parallel processing.

In this chapter, we use a method called double buffering to achieve parallel image processing. As shown in Fig.2, images are successively output from cameras 1 and 2. At times n = 0, 2, 4, ···, new images from cameras 1 and 2 are taken into buffers 11 and 21 respectively, while the last images saved in buffers 12 and 22 are processed. At times n = 1, 3, 5, ···, functions of the buffers are exchanged. Images from cameras 1 and 2 are taken into buffers 12 and 22 respectively, while the last images saved in buffers 11 and 21 are processed.

By using the parallel processing, a stereo image can be processed within the camera frame time T_c (33.33 ms). Therefore, the tool tip's position $p_t(n) \in R^{3\times 1}$ (n = 0, 1, 2, ···) in the robot base coordinate Σ_b can be detected at each camera's frame time T_c. Thus, real time image processing is achieved.

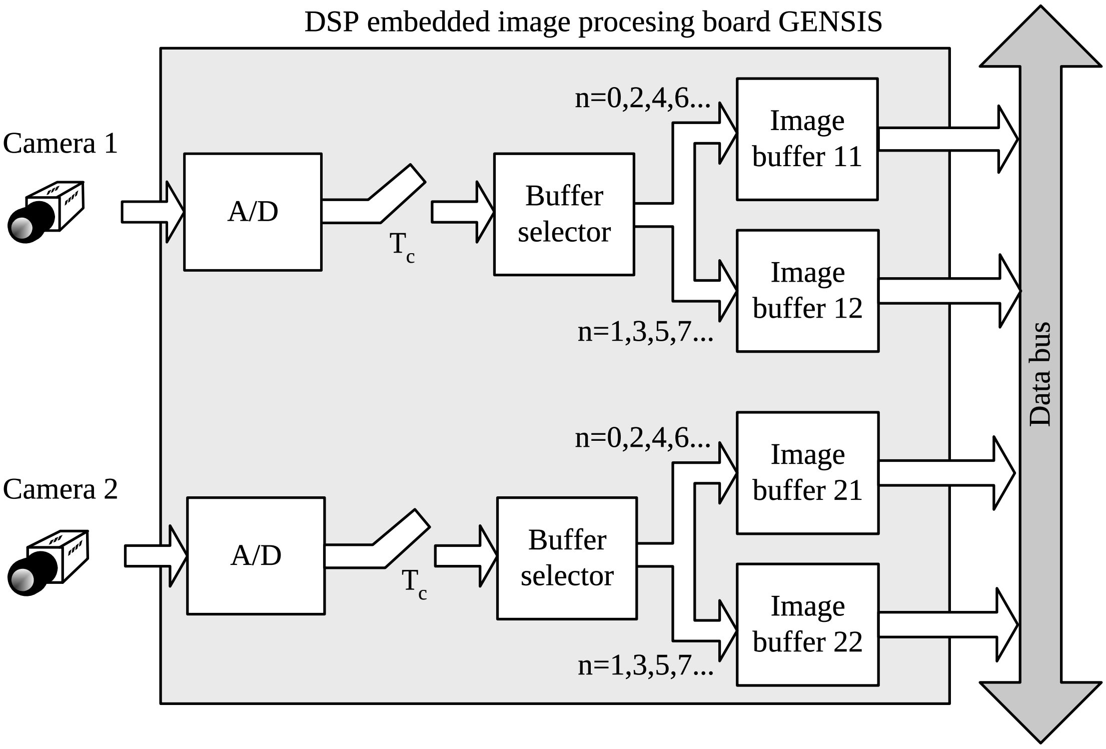


Figure 2. Parallel image processing in GENESIS using double buffering method

3. 2 Interpolation of the Image Detection Results

The robot control sampling period T is 5 milliseconds. At time t = k$_T$, the position of the flexible tool's tip in the robot base coordinate Σ_b is assumed to be $p_t(k) \in R^{3\times1}$. By using the real time image processing method described in section 3.1, the tool's tip position $p_t(n)$ in Σ_b can be detected within the camera frame time T$_c$. However, the image processing sampling period T$_c$ is still longer than the robot control sampling period T. A time chart of image processing and robot control is shown in Fig. 3, where T$_n$ (100 ms) is the least common multiple of T$_c$ and T. Therefore, interpolation is necessary to convert the result $p_t(n)$ of image processing to $p_t(k)$ at every sampling period T for robot control. In the period T$_n$, the position $p_t(k)$ can be calculated by

$$p_t(k) = \begin{cases} p_t(n-1) + (kT - (n-1)T_c)\dfrac{p_t(n-1) - p_t(n-2)}{T_c} \\ \qquad\qquad (nT_c \le kT < (n+1)T_c), \\ p_t(n) + (kT - nT_c)\dfrac{p_t(n) - p_t(n-1)}{T_c} \\ \qquad\qquad ((n+1)T_c < kT < (n+2)T_c), \\ p_t(n+1) + (kT - (n+1)T_c)\dfrac{p_t(n+1) - p_t(n)}{T_c} \\ \qquad\qquad ((n+2)T_c < kT < (n+3)T_c), \\ \cdots\cdots\cdots\cdots\cdots\cdots \\ \qquad\qquad (k = 0,1,2,\cdots),(n = 0,1,2,\cdots) . \end{cases} \tag{1}$$

In equation (1), $p_t(-1)$ and $p_t(-2)$ are set equal to 0 for n = 0.

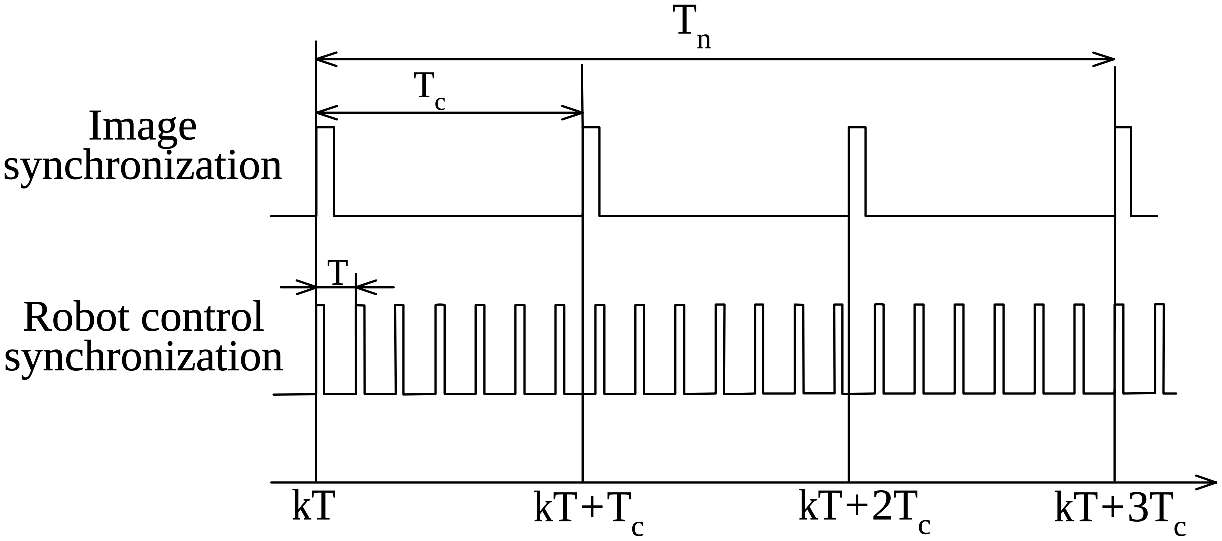


Figure 3. The time chart of image processing and robot control

4. Generation of Desired Trajectory

If a rigid tool held by a manipulator is used to complete the task as shown in Fig.1, the tool will not deform, so the position of the tool's tip can easily be calculated from the kinematics of the manipulator. However, when a flexible tool is used, it deforms on contact so that the position of the tool's tip cannot be directly computed from the kinematics of the manipulator. To explain the principle of the generation of the desired trajectory, a model is assumed as shown in Figs.4 and 5, where the tool's tip moves along the x axis with a constant pressing force in the z direction.

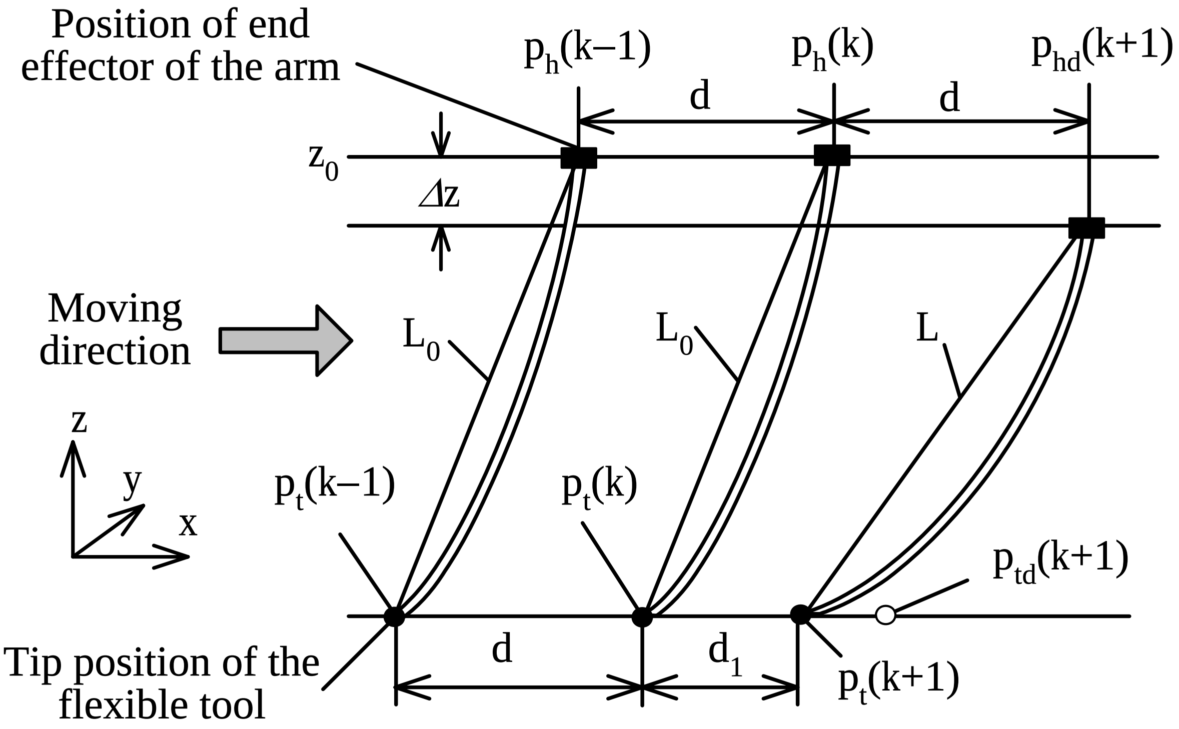


Figure 4. Moving end-effector of the robot forward with a distance d

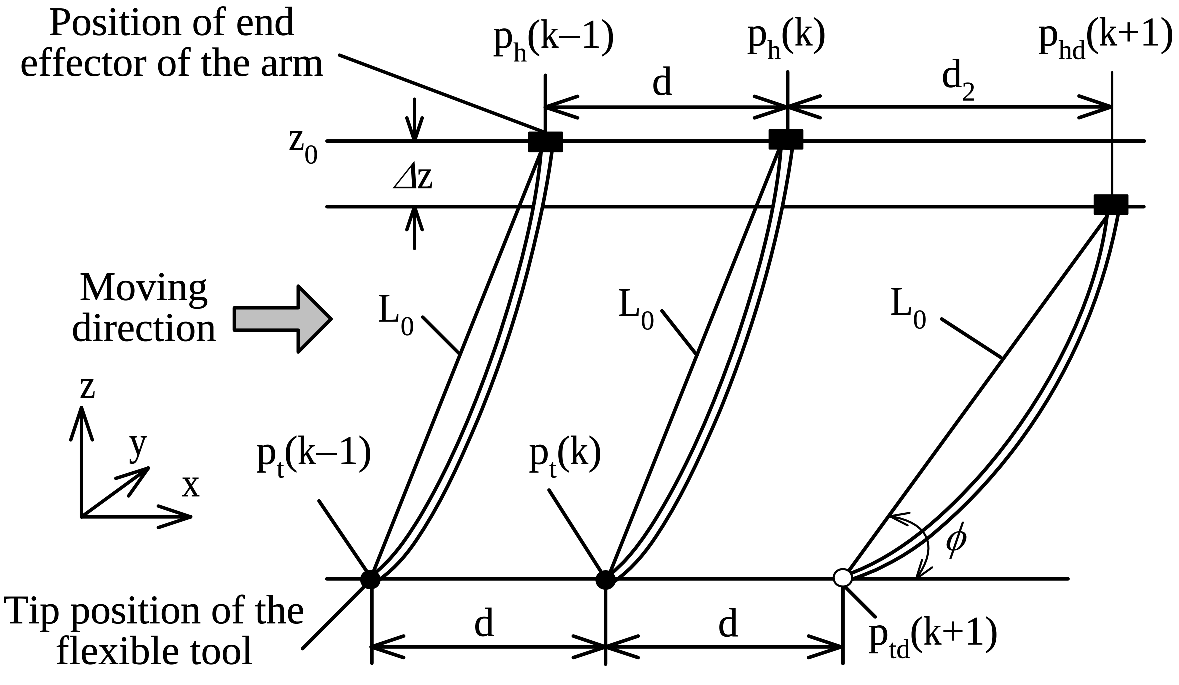


Figure 5. Moving the flexible tool forward with a distance d

Further assumptions of the task are the following:

- The desired trajectory traced by the flexible tool is in a given plane.
- The orientation around the y axis of the end-effector shown in Figs.4 and 5 is kept to be unchanged at its initial value until the task is completed.

4.1 Desired Trajectory Generation of the Robot's End-Effector

At time t = k_T, when the end-effector of the robot moves by a distance d from position $p_{hd}(k) \in R^{3\times1}$ to position $p_{hd}(k+1)$ as shown in Fig.4, the variation Δz along the z axis of the end-effctor caused by deformation will lead to the result that the flexible tool's tip moves by the distance d1 from position $p_t(k) \in R^{3\times1}$ to position $p_t(k+1)$ rather than to the desired position $p_{td}(k+1)$.

As shown in Fig.5, in order to move the tool's tip to the desired position ptd(k+1) ahead of position $p_t(k)$ by the distance d, the end-effector of the arm should be moved by the distance d2 from position ph(k) to position $p_{hd}(k+1)$ to compensate for the position error caused by the variation Δz of the robot end-effector.

Here, we assume that the distance between the tool tip and the end-effector is L0 when the tool presses against the contact surface with a constant force fzd. At time t = kT, the desired position of the tool's tip is expressed as $p_{td}(k+1)=[x_{td}(k+1),\ y_{td}(k+1),\ z_{td}(k+1)]^T$, and the angle between line L_0 and the x-y plane is ϕ. Thus, the moving direction vector $\tilde{p}_{td}(k)$ of the tool's tip can be computed by

$$\tilde{p}_{td}(k) = \frac{p_{td}(k+1) - p_{td}(k)}{d}, \tag{2}$$

where d is given by

$$d = \sqrt{(x_{td}(k+1) - x_{td}(k))^2 + (y_{td}(k+1) - y_{td}(k))^2}. \tag{3}$$

The projection vector $\varDelta p_L(k) \in R^{3\times1}$ of the tool on the x-y plane is

$$\varDelta p_L(k) = L_0 \cos(\phi) \cdot \tilde{p}_{td}(k), \tag{4}$$

where angle Φ can be calculated by

$$\phi = \sin^{-1}(\frac{z_0 - \varDelta z}{L_0}) \tag{5}$$

If the position error $\varDelta p_t(k)$ of the tool's tip is considered, the end-effector's desired position in the x-y plane is given by

$$p_{hd}^{xy}(k) = p_{td}(k+1) + \varDelta p_L(k) + \varDelta p_t(k), \tag{6}$$

Therefore, the desired position phd(k) of the end-effector is generated by

$$p_{hd}(k+1) = S_1 \cdot p_{hd}^{xy}(k) + z_d \cdot s_1, \tag{7}$$

where zd is equal to the initial coordinate z0, and matrix S1 and vector s1 are given by

$$S_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \tag{8}$$

$$s_1 = [0 \quad 0 \quad 1]^T. \tag{9}$$

4.2 Desired Orientation of the Arm's End-Effector

At time t = kT, the desired orientation of the end-effector is expressed as

$$r_{hd}(k) = [\alpha_d(k) \quad \beta_d(k) \quad \gamma_d(k)]^T , \tag{10}$$

where αd(k), βd(k) and γ d(k) are the rotation angle of the end-effector around the xb axis, yb axis and zb axis respectively, defined by the x-y-z fixed-angle method. If the end-effector pushes the tool to move along the desired curve as shown in Fig.6, a twisting moment will cause the rotation of the tool so that the tool's tip position will be uncontrollable.

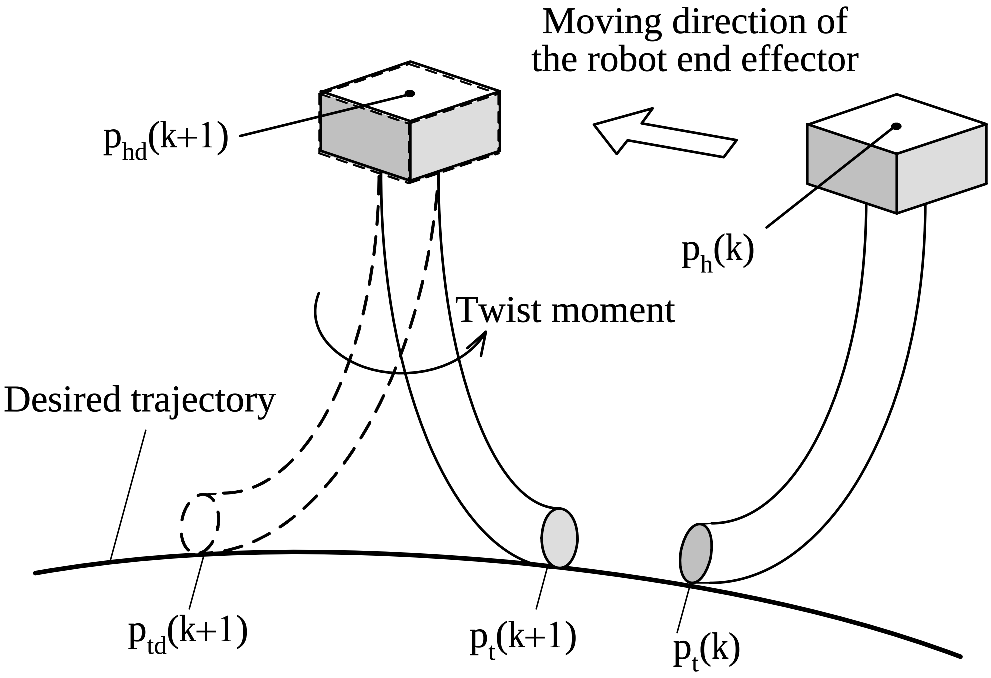


Figure 6. Rotation of the flexible tool caused by twist moment

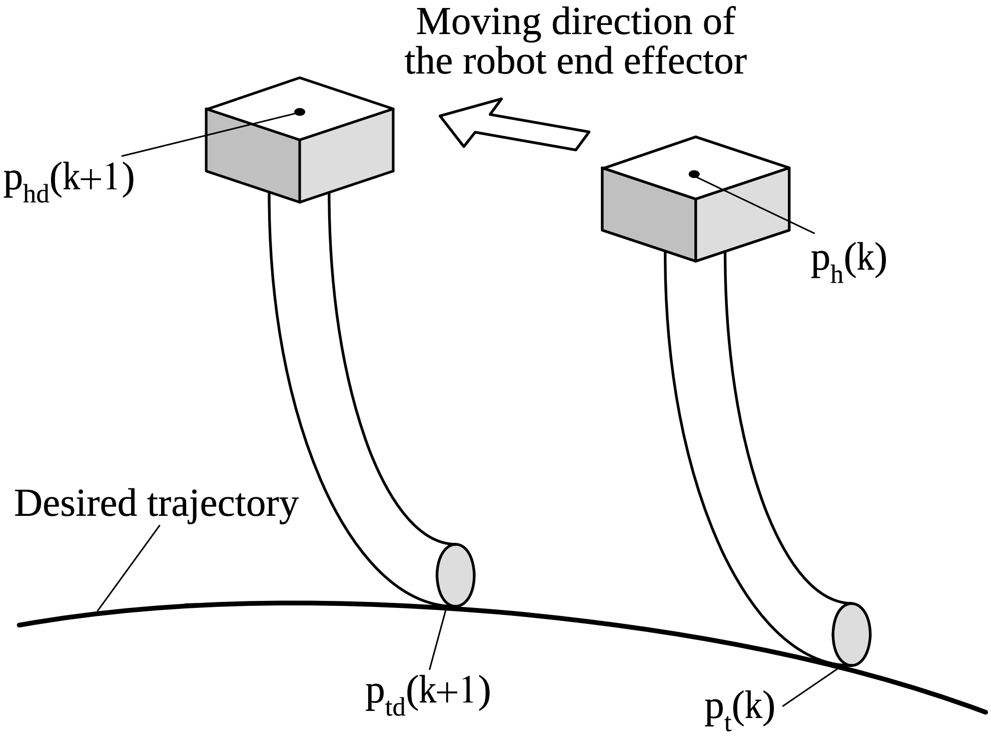


Figure 7. Rotation avoidance by pulling the flexible tool

In order to avoid rotation of the tool, a method of pulling the flexible tool with the end-effector is determined as shown in Fig.7. Therefore, the desired orientation of the end-effector is computed by

$$r_{hd}(k)=[\alpha_d \quad \beta_d \quad \tan^{-1}(\frac{y_{td}(k)}{x_{td}(k)})]^T \tag{11}$$

Then, by equations (7) and (11), the desired position and orientation prd(k+1)□□R6×1 of the end-effector can be calculated by

$$p_{rd}(k+1)=S_p \cdot p_{hd}(k)+S_r \cdot r_{hd}(k) \text{ ,} \tag{12}$$

where

$$S_p=\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\\ 0 & 0 & 0\\ 0 & 0 & 0\\ 0 & 0 & 0\end{bmatrix}, \qquad S_r=\begin{bmatrix}0 & 0 & 0\\ 0 & 0 & 0\\ 0 & 0 & 0\\ 1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix}.$$

4.3 Desired Pressing Force

As a constant force is required for the tip of the flexible tool to press the contact surface, the desired force f(k) is determined by

$$f(k)=\begin{bmatrix}0 & 0 & f_{zd}\end{bmatrix}^T \text{ ,} \tag{12}$$

where fzd is the constant force along the zb axis. If the end-effector pushes the tool as shown in Fig.6, a large pressing force acting between the tool's tip and the contact surface will increase friction so as to easily bring about rotation of the tool caused by the twist moment.

5. Hybrid Control Algorithm

5.1 Online Learning Neural Network

To complete the required task, position control in the xb-yb plane and force control along the zb axis are considered. However, unspecified factors such as friction at the tool's tip and deformation of the tool will impair the tracing accuracy of the tool.

To improve the tracing accuracy of the tool's tip in the xb-yb plane, two online learning neural networks are introduced, one each for the x and y directions as shown in Fig.8. For the neural network NNx, the desired position xhd(k), desired velocity $\dot{x}_{hd}(k)$ and desired acceleration $\ddot{x}_{hd}(k)$ are used as the input quantities. Just as in the NNx, the desired position yhd(k), desired velocity $\dot{y}_{hd}(k)$ and desired acceleration $\ddot{y}_{hd}(k)$ are used as the input quantities for the neural network NNy. The position errors of Δxr(k) on the xb axis and Δyr(k) on the yb axis are used as error signals for the neural network's learning.

The neural networks NNx and NNy have same 3-layer structure. The neurons in the input layer, hidden layer and output layer are designated NA, NB and NC respectively. The sigmoid function F(x) is given as

$$F(x) = \frac{1 - e^{-x/\mu}}{1 + e^{-x/\mu}}, \qquad (13)$$

where μ is the annealing parameter. Learning of the weighting coefficients is obtained with back propagation method [Cichocki & Unbehauen, 1993].

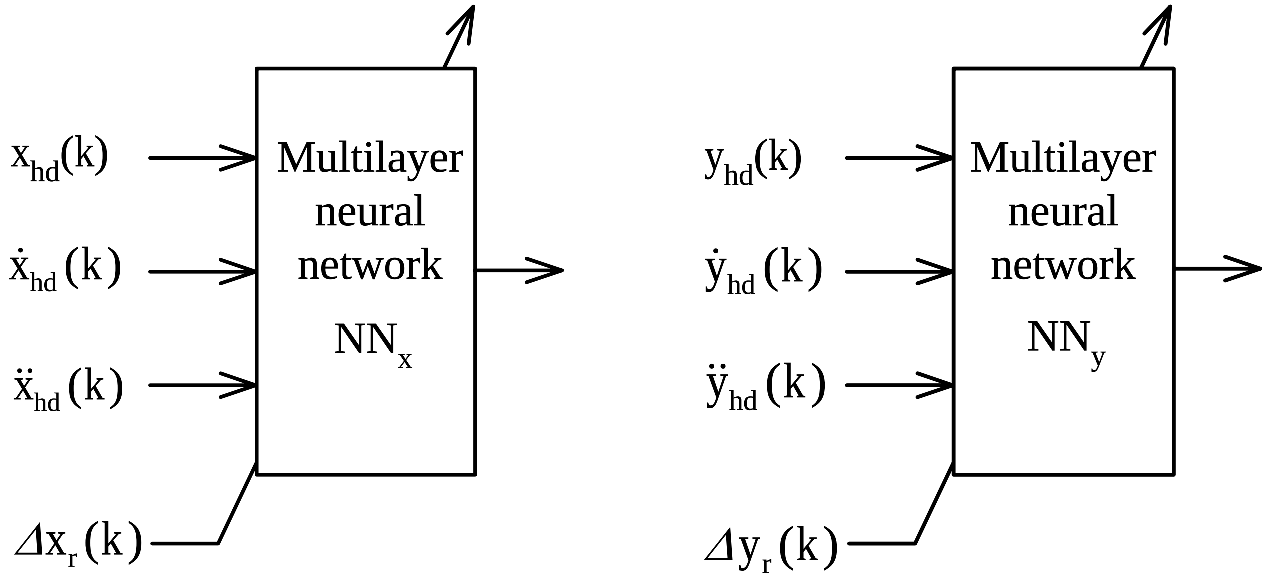


Figure 8. The proposed online learning neural networks

5.2 The Proposed Hybrid Control Method

A position/force hybrid control method using two online learning neural networks is proposed to enable a robot with a flexible tool to trace a given curve. The control block diagram is shown in Fig.9, where Λ is the kinematics, J is the Jacobian and Rf is the rotation matrix for calculating fz from fenv(k) detected by the force/torque sensor.

Matrix S2 and vector sf are given as

$$S_2 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}, \qquad s_f = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}^T$$

The PID controller Gp(z) of the position loop is networks

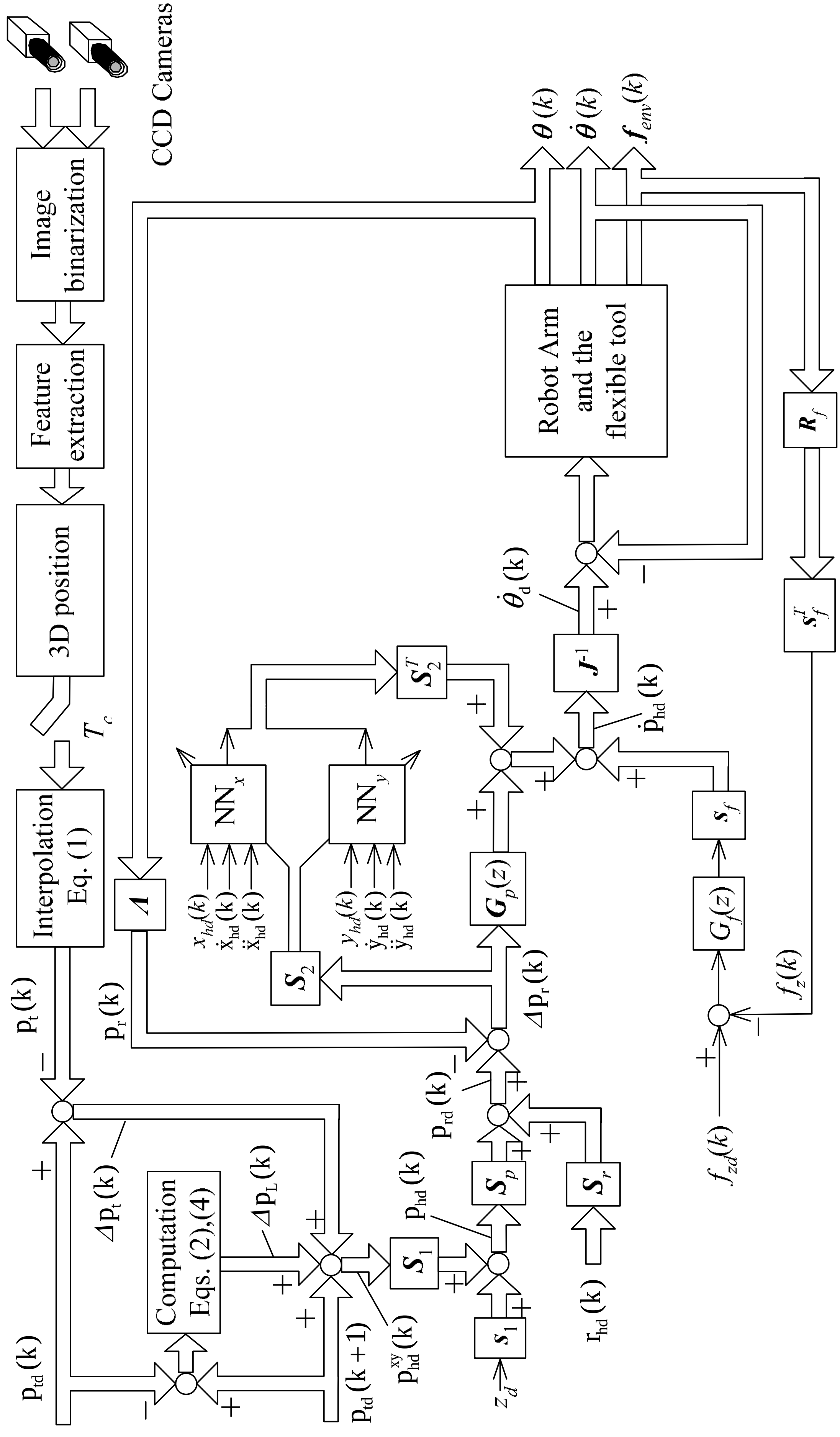
$$G_p(z) = K_P^p + K_I^p \frac{z}{z-1} + K_D^p(1 - z^{-1}), \qquad (14)$$

and the PID controller Gf(z) of the force loop is

$$G_f(z) = K_P^f + K_I^f \frac{z}{z-1} + K_D^f(1 - z^{-1}) \qquad (15)$$

For the position control loop in Fig.9, at time t = kT, interpolation is made by equation (1) for the image detection result p(n−1) so as to obtain the tool tip's position pt(k). After the projection of the vector ΔpL(k) on the x-y plane is calculated with equation (4), the desired position phd(k) can be generated by equation (7). Thus, the desired position and orientation prd(k) are computed by equation (12).

Position errors between the desired position prd(k) and the present position pr(k) calculated from the kinematics are used as error signals for the neural network's learning.



CCD Cameras
Image binarization
Feature extraction
3D position
T_c
Interpolation Eq. (1)
Computation Eqs. (2),(4)
Robot Arm and the flexible tool
$\boldsymbol{\theta}(k)$
$\dot{\boldsymbol{\theta}}(k)$
$\boldsymbol{f}_{env}(k)$
$\mathrm{p_t}(\mathrm{k})$
$\mathrm{p_r}(\mathrm{k})$
$\mathrm{p_{td}}(\mathrm{k})$
$\mathrm{p_{td}}(\mathrm{k}+1)$
$\Delta\mathrm{p_t}(\mathrm{k})$
$\Delta\mathrm{p_L}(\mathrm{k})$
$\mathrm{p_{hd}^{xy}}(\mathrm{k})$
$\mathrm{p_{hd}}(\mathrm{k})$
$\mathrm{p_{rd}}(\mathrm{k})$
$\Delta\mathrm{p_r}(\mathrm{k})$
$\dot{\mathrm{p}}_{\mathrm{hd}}(\mathrm{k})$
$\dot{\boldsymbol{\theta}}_{\mathrm{d}}(\mathrm{k})$
$\mathrm{r_{hd}}(\mathrm{k})$
z_d
$\boldsymbol{S}_1$
$\boldsymbol{s}_1$
$\boldsymbol{S}_p$
$\boldsymbol{S}_r$
$\boldsymbol{S}_2$
$\boldsymbol{S}_2^T$
$\boldsymbol{\Lambda}$
$\boldsymbol{G}_p(z)$
$G_f(z)$
$\boldsymbol{s}_f$
$\boldsymbol{s}_f^T$
$\boldsymbol{R}_f$
$\boldsymbol{J}^{-1}$
NN_x
NN_y
$x_{hd}(k)$
$\dot{\mathrm{x}}_{\mathrm{hd}}(\mathrm{k})$
$\ddot{\mathrm{x}}_{\mathrm{hd}}(\mathrm{k})$
$y_{hd}(k)$
$\dot{\mathrm{y}}_{\mathrm{hd}}(\mathrm{k})$
$\ddot{\mathrm{y}}_{\mathrm{hd}}(\mathrm{k})$
$f_{zd}(k)$
$f_z(k)$
+
−

Figure 9. The proposed position/force hybrid control algorithm with using the online learning

For the force control loop, the desired force fzd and the contact force fz in the z directrion are compared, and the force error is input to the PID controller Gf(z).

Then, the desired joint velocity $\dot{\theta}_{\mathrm{d}}(\mathrm{k})$ is computed by using the inverse of Jacobian J and is input to the servo driver of the manipulator.

6. Experiments and Discussion

Several experiments have been performed to prove the effectiveness of the proposed method as shown in Fig.9. The control parameters used in the experiments are given in Table 1.

$\mathrm{K}_{\mathrm{P}}^{\mathrm{p}} = \mathrm{diag}[\mathrm{K}_{\mathrm{P1}}^{\mathrm{p}} \quad \mathrm{K}_{\mathrm{P1}}^{\mathrm{p}} \quad 0 \quad \mathrm{K}_{\mathrm{P2}}^{\mathrm{p}} \quad \mathrm{K}_{\mathrm{P2}}^{\mathrm{p}} \quad \mathrm{K}_{\mathrm{P2}}^{\mathrm{p}}]$
$\mathrm{K}_{\mathrm{P1}}^{\mathrm{p}}=5.01/\mathrm{s}$, $\mathrm{K}_{\mathrm{P2}}^{\mathrm{p}}=2.01/\mathrm{s}$
$\mathrm{K}_{\mathrm{I}}^{\mathrm{p}} = \mathrm{diag}[\mathrm{K}_{\mathrm{I1}}^{\mathrm{p}} \quad \mathrm{K}_{\mathrm{I1}}^{\mathrm{p}} \quad 0 \quad \mathrm{K}_{\mathrm{I2}}^{\mathrm{p}} \quad \mathrm{K}_{\mathrm{I2}}^{\mathrm{p}} \quad \mathrm{K}_{\mathrm{I2}}^{\mathrm{p}}]$
$\mathrm{K}_{\mathrm{I1}}^{\mathrm{p}}=0.51/\mathrm{s}$, $\mathrm{K}_{\mathrm{I2}}^{\mathrm{p}}=0.21/\mathrm{s}$
$\mathrm{K}_{\mathrm{D}}^{\mathrm{p}} = \mathrm{diag}[\mathrm{K}_{\mathrm{D1}}^{\mathrm{p}} \quad \mathrm{K}_{\mathrm{D1}}^{\mathrm{p}} \quad 0 \quad \mathrm{K}_{\mathrm{D2}}^{\mathrm{p}} \quad \mathrm{K}_{\mathrm{D2}}^{\mathrm{p}} \quad \mathrm{K}_{\mathrm{D2}}^{\mathrm{p}}]$
$\mathrm{K}_{\mathrm{D1}}^{\mathrm{p}}=0.11/\mathrm{s}$, $\mathrm{K}_{\mathrm{D2}}^{\mathrm{p}}=0.11/\mathrm{s}$
$\mathrm{K}_{\mathrm{P}}^{\mathrm{f}}=0.001\ \mathrm{m/(s \cdot N)}$, $\mathrm{K}_{\mathrm{I}}^{\mathrm{f}}=0.0005\ \mathrm{m/(s \cdot N)}$, $\mathrm{K}_{\mathrm{D}}^{\mathrm{f}}=0.0001\ \mathrm{m/(s \cdot N)}$

Table 1. Control parameters used in the experiments

6.1 Influences on Tracing Accuracy of the Tool Caused by Twist Moment (Without Applying Neural Networks and Orientation Control)

A circular trajectory is given in Fig.10. The radius R of the circle is 0.1 meters. In order to investigate the influence of twist moments on the flexible tool, position control is imposed on the end-effector, while the orientation of the end-effector is fixed at its initial value. In this experiment, the stereovision detection and neural networks described in Fig.9 are not used.

Other parameters used are as follows.

For position/orientation control:z_d=0.750,αd=0, β_d=1.222 rad, γ_d=0.

For force control:fzd = 2 N.

Results for the end-effector's position ph(k) and the tool's tip position pt(k) are shown in Fig.11. Because the tool's deformation is not considered, a position error in the starting point caused by deformation of the flexible tool is observed in Fig.11. On the path through points p_0, p_1 and p_2 as shown in Fig.11, the tool's tip follows the movement of the end-effector but a large position error is generated. Rotation of the flexible tool caused by a twist moment is not encountered on this route because the flexible tool is pulled by the end-effector. However, on the path along points p_2, p_3 and p_0, the end-effector pushes the flexible tool, so that a twist moment causes rotation of the tool. As a result, the tool's tip is uncontrollable. The above results demonstrate that the tool's deformation and the effect of the twist moment must be considered.

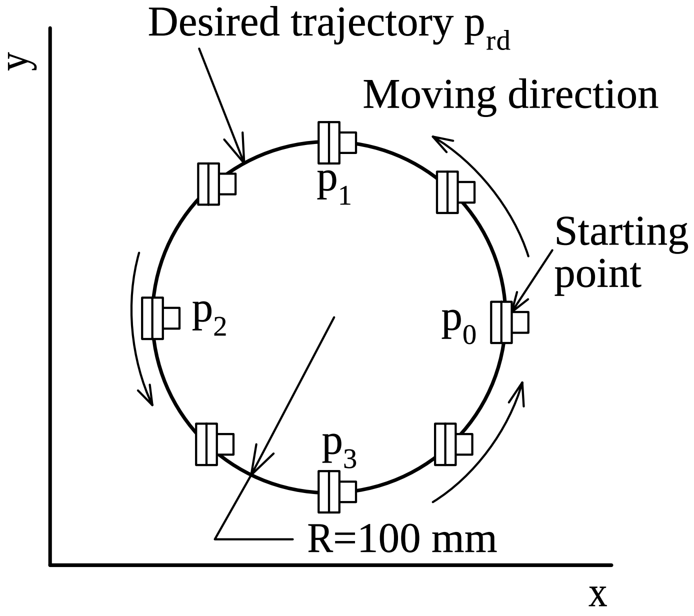


Figure 10. Desired trajectory of the tool's tip

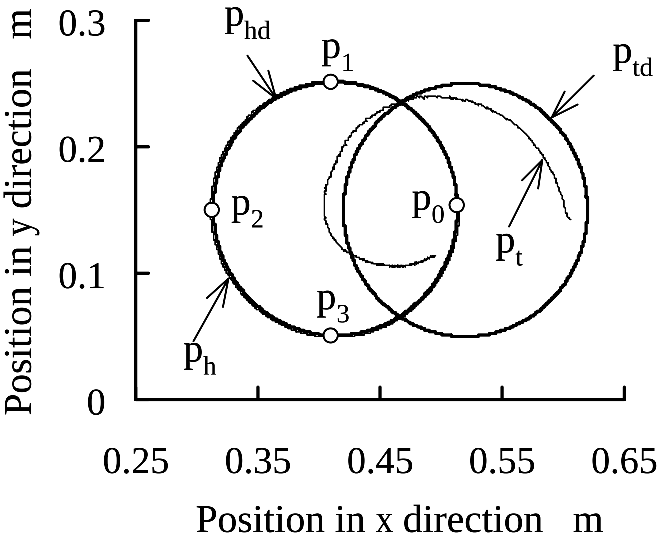


Figure 11. The tool's tip position p_t and end-effector's position p_h

6.2 Tracing Experiment (Without Using Neural Networks)

In order to avoid twist-moment-induced rotation of the flexible tool, it is pulled by the end-effector to trace the desired trajectory as shown in Fig.7. However, due to the arm's mechanical structure, the end-effector can not rotate 360 degrees to trace the circular trajectory shown in Fig.10. Therefore, a sinusoidal curve is given as the desired trajectory in Fig.12. In this experiment, stereovision detection by real time image processing is used, but the neural networks described in Fig.9 are not included in position control. Other parameters are used as follows.

For position/orientation control:zd=0.690,αd=0, βd=1.047 rad.

For force control:fzd = 0.5 N.

The end-effector's position ph(k) and the tool's tip position pt(k) are shown in Fig.13. Compared with the results shown in Fig.11, the tool's tip traces the desired sinusoidal curve with the movement of the end-effector. Furthermore, rotation of the flexible tool caused by the twist moment is successfully avoided by the proposed motion control of the end-effector as shown in Fig.7. However, in Fig.13, the maximum position error between tool tip's position pt(k) and the desired trajectory ptd(k) along the x axis is 8 mm, and the maximum position error between ptd(k) and pt(k) along the y axis is 12 mm.

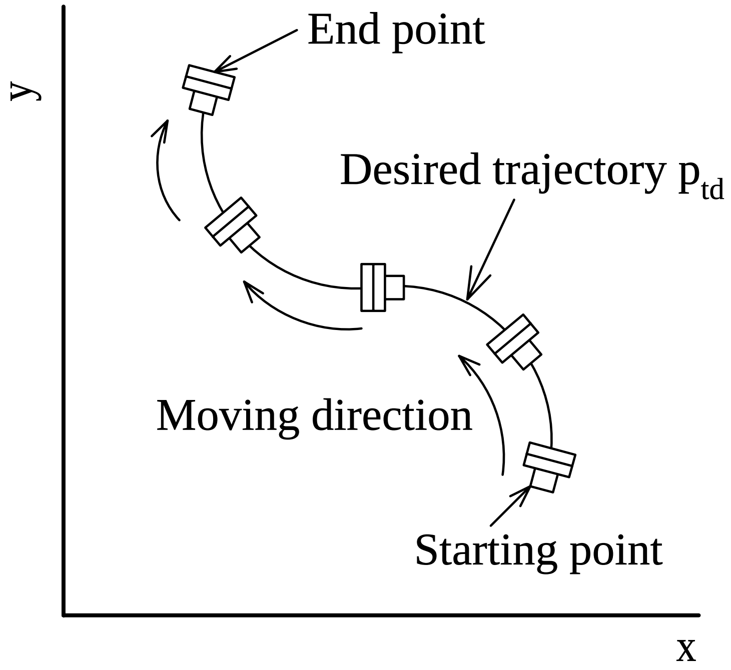


Figure 12. Desired trajectory of the tool's tip

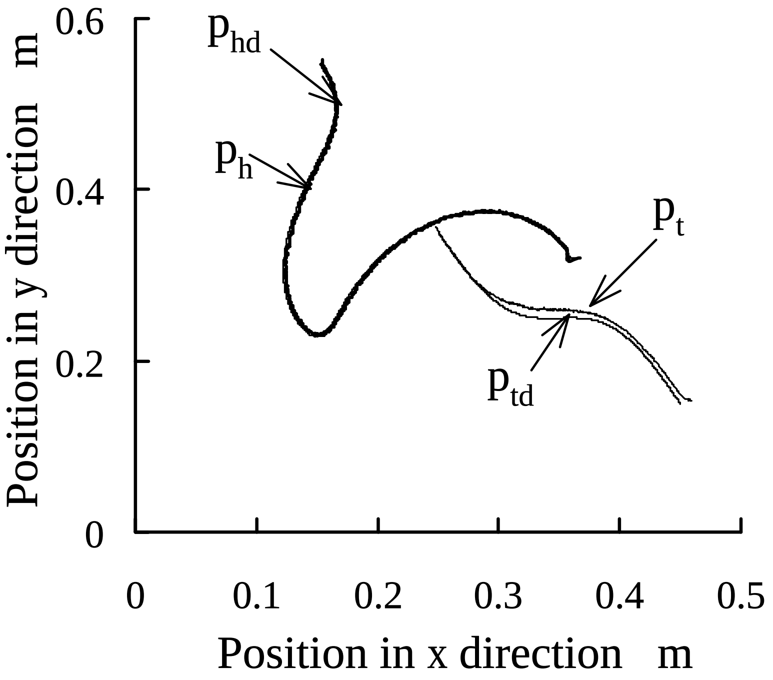


Figure 13. The tool tip's position pt and end effector's position p_h

6.3 Tracing Experiment Using the Proposed Method

In order to improve the tracing accuracy of the flexible tool, the proposed control method using online neural network learning is applied. The desired trajectory is same sinusoidal curve as shown in Fig.12. Other parameters are as follows.

For position/orientation control:zd=0.690,αd=0, βd=1.047 rad.
For force control:fzd = 0.5 N.
The following parameters are used for the proposed neural networks.
Neuron number in the input layer NA= 3.
Neuron number in the hidden layer NB= 6.
Neuron number in the output layer NA= 1.
Annealing rate μ = 1.0.
Learning rate η = 0.85.
The initial values of weighting coefficients of the neural networks are set randomly, and all weighting coefficients are saved in a file for continued learning in the next tracing. It is known that a number of trials are generally needed before the learning error converges [Liu & Todo, 1991]. Results of the error signals for the neural network's learning are shown in Fig.14, where the learning errors settled near zero at the end of the first tracing after 10,000 trials. Furthermore, compared with the error obtained in the first tracing, the tracing accuracy is obviously improved in the second tracing.

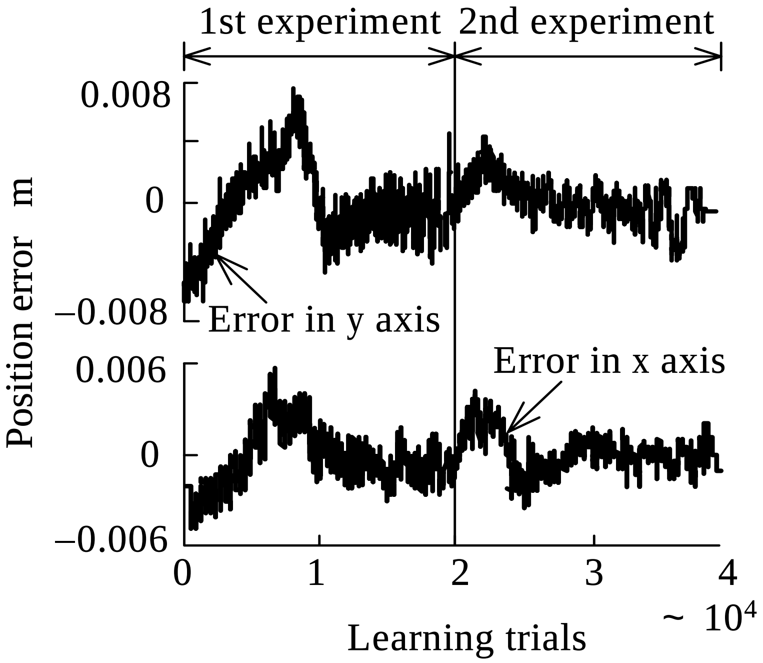


Figure 14. The error signals for neural network learning

The tracing trajectories of the tool's tip and the end-effector's position in the x-y plane are shown in Fig.15, and result of the forces fz is shown in Fig.16. Compared with the results shown in Fig.12, the tracing accuracy of the tool's tip is greatly improved. The maximum position errors between ptd(k) and pt(k) on both the x and y axes are decreased to 4 mm, which is almost half of the error in the experiment without neural networks. Furthermore, the contact force between the tool's tip and the surface is accurately controlled at the desired force fzd.

7. Conclusions and Future Work

For robotic manipulation of a fragile object, using a flexible tool fixed to a robot arm is the obvious choice. However, a flexible tool deforms on contact and control of the flexible tool is difficult because the position of the tool's tip cannot be calculated from the kinematics of the robot arm. We have developed a new approach that is not based on establishing a deformation model to calculate the tool tip's position, but that uses real time image processing with stereovision. Furthermore, an interpolation algorithm is proposed to convert the image-processing results detected in each camera frame time to the results used at every sampling period for robot control.

Visual detection is a convenient and effective way to obtain information on a deformable object with unknown parameters, which is generally required for deformation model analysis.

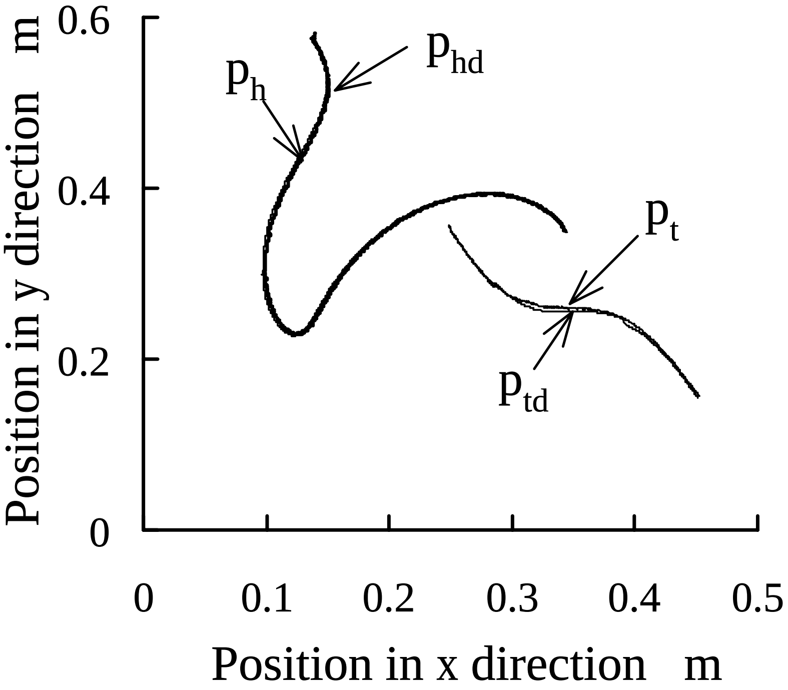


Figure 15. The tool's tip position pt and end effector's position p_h

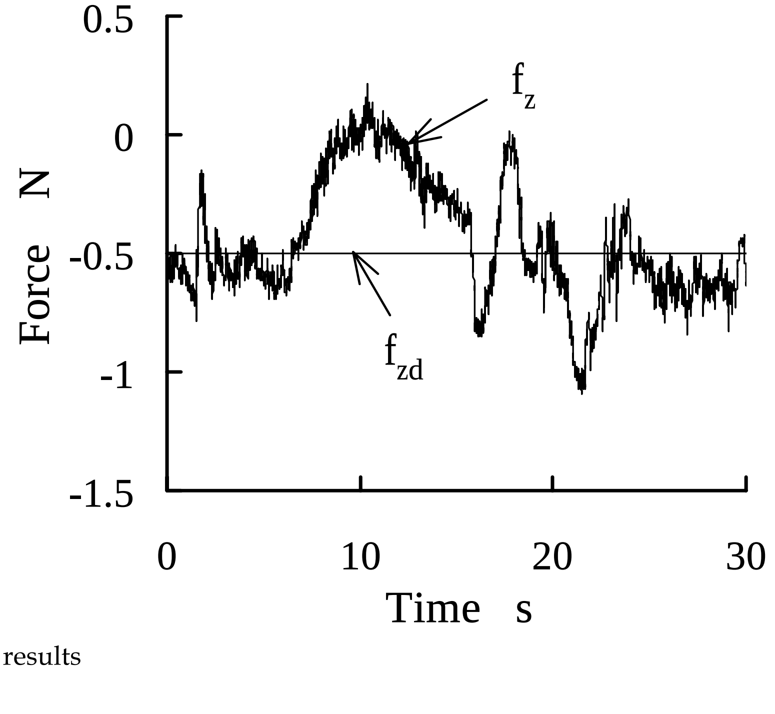


Figure 16. The force results

As shown in Fig.6, control of the deformable tool is difficult because twist moment will cause a rotation of the tool's body so that position of the tool is uncontrollable. Therefore, pulling the flexible tool by the end-effector is considered. In this chapter, a position and force hybrid control method using visual and force information is proposed to enable a flexible tool held by a manipulator to trace a specified curve in a plane. To improve tracing accuracy, online learning neural networks are introduced to construct a feed-forward controller so that the position error of the tool's tip is decreased. The proposed method is used for a manipulator with a flexible tool to trace a sinusoidal curve and it effectiveness is experimentally demonstrated.

For future work, the influence of the pressing force and the physical characteristics of the contact surface will be investigated. In this chapter, a method of avoiding tool rotation caused by the twist moment is proposed. In practice, because the friction coefficients of the contact surfaces vary greatly, any torque can easily cause rotation of the flexible tool. Therefore, a control strategy should be developed to adjust the end-effector's orientation when rotation of the tool occurs. To achieve this control, the visual detection will first be improved by using a number of feature points on the tool body so as to obtain its deformation state. A control algorithm for the end-effector of the arm is also needed for adequate adjustment of the flexible tool.

8. References

Acker, J. & Henrich, D. (2003). Manipulating deformable linear objects: characteristic features for vision-based detection of contact state transitions, Proc. IEEE Int. Symp. on Assembly and Task Planning, pp.204–209, July 10-11, 2003

Abegg, F.; Remde, A. & Henrich, D. (2000). Force and vision based detection of contact state transition, Robot manipulation of deformable objects, Springer, ISBN 1852332506

Cichocki, A. & Unbehauen, R. (1993). Neural networks for optimization and signalprocessing, John Wiley & Sons ISBN 0471930105,

Chen, C. Y. & Zheng,Y. F. (1992). Deformation identification and estimation of one-dimension objects by vision sensors, Journal of Robotic Systems, Vol.9, No.5, pp.595-612

Henrich, D.; Ogasawara, T. & Worn, H. M. (1999). Manipulating deformable linear objects -Contact states and point contacts-, Proc. IEEE Int. Symp. on Assembly and Task Planning, pp.198 - 204, July 21-24, 1999

Hirai, S. (1998). Deformable object manipulation, J. of the Robotics Society of Japan, Vol.16, No.2, pp.136-139

Hirai, S. & Noguchi, H. (1997). Human-demonstration based approach to the object motion design and the recognition of process state transitions in insertion of deformable tubes, J. of the Robotics Society of Japan, Vol.18, No.8, pp.1172-1179

Hisada, T. (1998). Finite element modeling, J. of the Robotics Society of Japan, Vol.16, No.2, pp.140-144

Huang, J. & Todo, I. (2001). Control of a robot based on fusion of visual and force/torque sensor information (Manipulation of a deformable object), Trans. Japan. Soc. Mech. Eng., Series C, Vol.67, No.660, pp.2616-2623

Huang, J.; Todo, I. & Muramatsu, I. (2003). Neuro-control of a robot using visual and force/torque sensor information (Manipulation of a flexible beam object), Trans. Japan. Soc. Mech. Eng., Series C, Vol.69, No.684, pp.2085-2092

Itakura, O. (1998). Manipulation of paper material – ticket handling in station business machine –, J. of the Robotics Society of Japan, Vol.16, No.2, pp.154-158

Liu, M. H. & Todo, I. (1991). Digital control of servo systems using neural networks (A method of off-line learning), Trans. Japan. Soc. Mech. Eng., Series C, Vol.57, No.539, pp.2256-2262

Nakagaki, H. (1998). Insertion task of a flexible beam or a flexible wire, Journal of the Robotics Society of Japan, Vol.16, No.2, pp.159-162

Nakagaki, H.; Kitagaki, K.; Ogasawara, T. & Tsukune, H. (1997). Estimation of a force acting on a flexible wire by using visual tracking and its application to insertion task, J. of the Robotics Society of Japan, Vol.15, No.3, pp.422-430

Ono, E. (1998). Fabric Manipulation, J. of the Robotics Society of Japan, Vol.16, No.2, pp.149-153

Remde, A.; Henrich, D. & Wom, H. (1999). Manipulating deformable linear objects-contact state transitions and transition conditions, Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems,Vol.3, pp.1450-1455, Oct.17-21,1999

Schlechter, A. & Henrich, D. (2002). Manipulating deformable linear objects: manipulation skill for active damping of oscillations, Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and System, Vol.2, pp.1541-1546, Sept.30-Oct.5, 2002

Schmidt, T. W. & Henrich, D. (2001). Manipulating deformable linear objects: robot motions in single and multiple contact points, Proc. IEEE Int. Symp. on Assembly and Task Planning, pp.435-441, May 28-29, 2001

Wakamatsu, H. & Wada, T. (1998). Modeling of string object for their manipulation, J. of the Robotics Society of Japan, Vol.16, No.2, pp.145-148

Wakamatsu, H.; Tanaka, Y.; Tsumaya, A.; Shirase, K. & Arai, E. (2002). Representation and planning of deformable linear object manipulation including knotting, Proc. IEEE Int. Conf. on Industrial Technology, Vol.2, pp.1321-1326, Dec.11-14, 2002

Wakamatsu, H.; Tsumaya, A.; Arai, E. & Hirai, S. (2004). Planning of one-handed knotting/raveling manipulation of linear objects, Proc. IEEE International Conference on Robotics and Automation, Vol.2, pp.1719-1725, April 26-May 1, 2004

Wu, J. Q.; Luo, Z. W.; Yamakita, M. & Ito, K. (1997). Dynamic position/force control of manipulators for contact tasks on unknown flexible plate, Trans. Japan. Soc. Mech. Eng., Vol.63, No.607, pp.937-944

Yue, S. & Henrich, D. (2002). Manipulating deformable linear objects: sensor-based fast manipulation during vibration, Proc. IEEE Int. Conf. on Robotics and Automation, Vol.3, pp.2467-2472, May11-15, 2002

A Novel Parallel Engraving Machine Based on 6-PUS Mechanism and Related Technologies

Kong Ling-fu & Zhang Shi-hui

1. Introduction

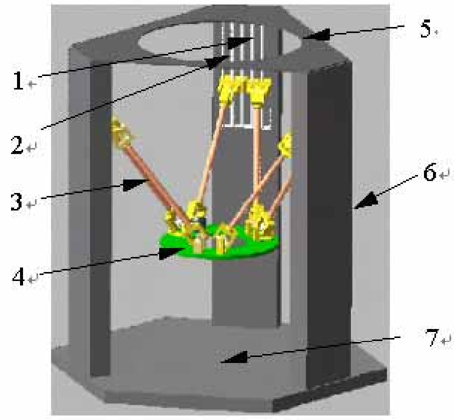
Conventional computer engraving machine has played an important role in industries such as machinery machining, printing and dyeing and entertainment, but it has the inherent disadvantages such as cutting tool can be fed only along the fixed guideway, lower degree-of-freedom(DOF) of cutting tool, lower flexibility and mobility for machining etc. Parallel mechanism has the merits such as high mechanical stiffness, high load capacity, high precision, good dynamic performance etc (Huang,1997). According to the characteristics of parallel mechanism, it has been a hot research topic to apply parallel mechanism to the domain of future machining. By applying parallel mechanism to engraving domain, its inherent advantages can be fully exerted and the disadvantages of conventional engraving machine can be overcome or compensated. But as the special structure of parallel mechanism, the related theory and technology during its engraving is very different from that of conventional engraving machine, and it is a undeveloped research topic by now. In addition, with the development of computer network technology, the new concept and method such as network machining and manufacturing has become hot research topic(Huang & Mak,2001; Taylor & Dalton,2000; Yao & Lu,1999). A novel parallel engraving machine with six-axis linkage is proposed in this paper, which uses the 6-PUS parallel mechanism with 6-DOF as the prototype, and some key technologies such as size design, tool path planning, engraving force control and teleoperation are studied on this basis.

2. Confirming of Mechanism Type and Engraving Machine's Size

2.1 Selection of Mechanism Type and Coordinate System

The selection of mechanism type is the first step for designing novel engraving machine, the following reasons make us select the 6-PUS parallel mechanism for designing our engraving machine. Comparing with traditional mechanism, 6-PUS parallel mechanism uses base platform, three uprights layout and high rigidity framework structure and has the merits such as high modularization, high accuracy and low cost. Its model is shown in Fig.1.

As shown in Fig.1, 6-PUS parallel mechanism consists of base platform, dynamic platform and 6 branch chains with same structure, every branch joins with base platform through prismatic pairs (P), slider of prismatic pairs joins with up end of the fixed length link through universal joint (U) down end of the fixed length link joins with dynamic platform through sphere hinge (S), so it is called 6-PUS parallel mechanism.



1. The first lead screw 2. The second lead screw
3. Fixed length link 4. Dynamic platform
5. Up plane of base platform 6. Upright of base platform
7. Down plane of base platform

Figure 1. The model of 6-PUS parallel mechanism

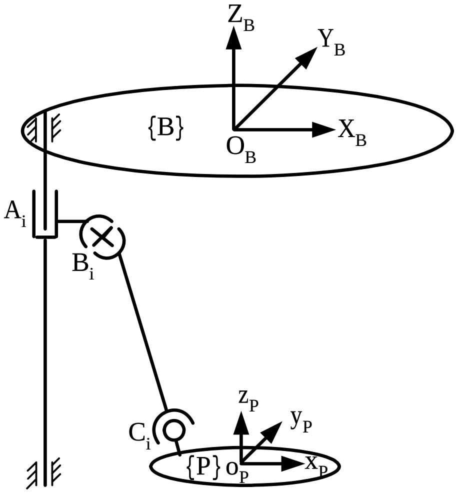


Figure 2. Coordinate system of 6-PUS parallel engraving mechanism

The coordinate system of 6-PUS parallel engraving mechanism is shown in Fig. 2.
In Fig.2, the geometry centers of base platform and dynamic platform plane are supposed as O_B and o_p respectively. In every branch, the centers of prismatic pairs, universal joint and sphere hinge are marked with Ai, Bi, and Ci (i=1,2,…,6) respectively. Coordinate system $O_B\text{-}X_BY_BZ_B$ is fixed on base platform, taking {B} as briefly. The origin of {B} lies on geometry center of base platform's up plane, axis Z_B is vertical with base platform and directs to up, axis Y_B directs to angle bisector of the first and second branch lead screw center line, and axis X_B can be determined with right-hand rule. Supposing the coordinate system set on dynamic platform is $o_p - x_p y_p z_p$, taking {P} as briefly, its origin lies on geometry center of dynamic platform, the initial state of dynamic platform system is consistent with that of base platform system completely. Supposing the coordinate of o_p is (00Z) in {B}, this configuration without relative rotation to every axis is the initial configuration of this mechanism, and Z changing with mechanism's size. On the basis of

coordinate system mentioned, we use influence coefficient theory and the actual parameters of this mechanism to calculate the first and the second order influence coefficient matrix of every branch under different configuration. Then, we can get the first and the second order integrated influence coefficient matrix $\left[G_q^H\right]_{6x6}$ and $\left[H_q^H\right]_{6x6x6}$ of the whole mechanism. The significance and detailed solution process for influence coefficient matrix is omitted here, for more information please refer (Huang et al., 1997).

2.2 Mechanism Performance Analysis Based on Influence Coefficient Matrix

The performance of engraving machine will change with its size. To find out the better size for all the performance indices of both kinematics and dynamics, we obtain a group of mechanisms by changing its parameters. These mechanisms' length of fixed length links (L) range between 45cm and 55cm (step is 1cm), radius of dynamic platform (R) range between 10cm and 20cm (step is 1cm). Other parameters of the mechanism are unchanging, so we get 121 mechanisms totally.

Taking these mechanisms as research object, we confirm the sample point for every mechanism in its workspace with algorithm PerformanceAnalysis, then calculate the first and the second order influence coefficient matrix in every point. Furthermore, calculate all the performance indices in every sample point and draw all the global performance atlas of 121 mechanisms ultimately. To describe conveniently, we abbreviate the first and the second order integrated influence coefficient matrix $\left[G_q^H\right]_{6x6}$ and $\left[H_q^H\right]_{6x6x6}$ to G and H, and use $G\omega$, $H\omega$ and $G\upsilon$, $H\upsilon$ as the angular velocity submatrix and linear velocity submatrix of the first and the second order integrated influence coefficient matrix respectively, namely, $G = \left(\frac{G_\omega}{G_v}\right)$ and $H = \left(\frac{H_\omega}{H_v}\right)$.

Description for algorithm PerformanceAnalysis:

```
PerformanceAnalysis
    Begin
        For L=45 To 55   //scope of fixed length link
            For R=10 To 20   //scope of radius of dynamic platform
                SamplePointNumber=0 // initialization sample point number is zero for every mechanism
                For x= -Maximum To +Maximum moving along Axis X     Step 4cm
                    For y=-Maximum To +Maximum moving along Axis Y   Step 4cm
                       For z= -Maximum To +Maximum moving along Axis Z   Step 4cm
                            For $\alpha$= -Maximum To +Maximum rotating around Axis X   Step 12°
                                For $\beta$=-Maximum To +Maximum rotating around Axis Y   Step 12°
                                    For $\gamma$=-Maximum To +Maximum rotating around Axis Z   Step 12°
                                      If sample point(x,y,z,$\alpha$,$\beta$,$\gamma$)reachable point of mechanism's workspace
                                             Calculating the first order influence coefficient matrix and
                                             its Frobenius norm at current point ;
                                           If   The first order influence coefficient matrix is not singular
                                               SamplePointNumber= SamplePointNumber+1;
                                               Calculating the second order influence coefficient matrix
                                                and its Frobenius norm
                                               calculating condition number at this point with formula
                                                and accumulating sum of performance indices;
                                               //detailed formula is given in the following of this section
                                           Endif
                                       Endif
                                   Endfor
                               Endfor
                           Endfor
                       Endfor
                    Endfor
                Endfor
                Calculating all the performance indices of the mechanism at current size and append the results
                to corresponding data files for different performance index;
                //performance index of the mechanism =
                (accumulating sum of performance indices at all sample points)/ SamplePointNumber
                //There are six data files for performance indices totally: angular velocity, linear velocity,
                  angular acceleration, linear acceleration, force and moment, inertia force
            Endfor
        Endfor
        Drawing all the global performance atlas of 6-PUS mechanism by all the index data files
        (Every data file includes the information of 121 mechanisms);
        //There are six performances atlas totally: angular velocity, linear velocity, angular acceleration, linear acceleration,
        force and moment, inertia force
    End
```

We can change mechanism's parameters and adjust variable's step in the algorithm PerformanceAnalysis to meet actual analysis.

The algorithm is programmed with MATLAB and the global performance atlases of 6-PUS mechanism are drawn (see Fig. 3 to Fig. 8), then the mechanism's performance is analyzed using the atlas.

Table 1 shows the results of sample point number (abbr. to SPN) for 121 mechanisms respectively, the fixed link length of mechanism with sequence number (abbr. to SN) 1 is 45cm, its radius of dynamic platform is 10cm, the fixed link length of mechanism with SN 121 is 55cm, its radium of dynamic platform is 20cm, the rest may be deduced by analogy.

SN	SPN	SN	SPN	SN	SPN	SN	SPN	SN	SPN
1	30962	15	26874	29	20492	43	13852	57	52794
2	28074	16	23906	30	17530	44	11516	58	47350
3	25848	17	21026	31	14848	45	52222	59	42390
4	23252	18	18252	32	12466	46	47554	60	37410
5	20816	19	15784	33	10422	47	43064	61	32446
6	18368	20	13376	34	46608	48	38090	62	27818
7	16168	21	11304	35	42654	49	33780	63	23586
8	13824	22	9464	36	38406	50	29516	64	19902
9	11936	23	41324	37	34386	51	25254	65	16442
10	10106	24	37446	38	30214	52	21500	66	13696
11	8490	25	34194	39	26674	53	18098	67	63950
12	35554	26	30464	40	22830	54	15154	68	58034
13	32872	27	27038	41	19510	55	12570	69	52506
14	29628	28	23648	42	16410	56	58068	70	46696

SN	SPN	SN	SPN	SN	SPN	SN	SPN
71	41040	85	27942	99	17136	113	72098
72	35562	86	23204	100	81766	114	63828
73	30592	87	19196	101	74616	115	56058
74	25994	88	15858	102	66882	116	48546
75	21628	89	75848	103	59488	117	41388
76	17898	90	68936	104	52150	118	34892
77	14718	91	62052	105	45198	119	28942
78	69740	92	55164	106	38610	120	23998
79	63616	93	48462	107	32468	121	19828
80	57254	94	41880	108	27122		
81	50898	95	35792	109	22328		
82	44428	96	30158	110	18462		
83	38696	97	25188	111	88256		
84	33070	98	20736	112	79990		

Table 1. The SPN of 121 mechanisms in experiment

In addition, table 2 gives the performance indices of some mechanism only, where the mean of SN is same as in table 1.

SN	SPN	six performance indices					
		angular velocity	linear velocity	angular acceleration	linear acceleration	Force and moment	inertia force
1	30962	0.17276	0.17442	0.06236	0.11315	0.01521	0.37454
2	28074	0.18248	0.18171	0.08075	0.13276	0.01456	0.40421
3	25848	0.19128	0.18836	0.09932	0.15184	0.01396	0.43136
4	23252	0.20087	0.19545	0.11897	0.17225	0.01348	0.46030
…	…	…	…	…	…	…	…
59	42390	0.21105	0.18995	0.10050	0.15604	0.01304	0.40233
60	37410	0.21915	0.19537	0.11308	0.17355	0.01257	0.42606
61	32446	0.22717	0.20041	0.12312	0.19230	0.01216	0.44929
…	…	…	…	…	…	…	…
119	28942	0.25779	0.20680	0.12265	0.22596	0.01064	0.47030
120	23998	0.26786	0.21185	0.12116	0.24139	0.01041	0.49500
121	19828	0.27714	0.21610	0.11399	0.25527	0.01017	0.51745

Table 2. Six performance indices of some mechanism

2.2.1 Analysis of Kinematics Performance Indices

2.2.1.1 Global Performance Indices of Angular Velocity and Linear Velocity

As the influence coefficient G of engraving mechanism is not a constant matrix it makes the measuring index for parallel mechanism based on G not to be a constant matrix also, so we can't utilize one value to measure the good or bad of the dexterity, isotropy and controlling accuracy(Guo,2002). Here, we define parallel mechanism global performance indices of angular velocity and linear velocity as following respectively

$$\eta_{G_\omega} = \frac{\int_W \frac{1}{K_{G_\omega}} dW}{\int_W dW} \quad \text{and} \quad \eta_{G_v} = \frac{\int_W \frac{1}{K_{G_v}} dW}{\int_W dW} \tag{1}$$

Where W is the reachable workspace of mechanism, $K_{G_\omega} = \|G_\omega\| \|G_\omega^+\|$ and $K_{G_\upsilon} = \|G_\upsilon\| \|G_\upsilon^+\|$ denote the condition numbers for angular velocity and linear velocity respectively (Where ||·|| denotes Frobenius norm of matrix, superscript '+' denotes generalized inverse matrix, the same mean as following).We can get the performance indices' value of the angular velocity and linear velocity according to the condition numbers of every mechanism's sample points. Replacing the underlined part in algorithm PerformanceAnalysis with two formulas in (1) respectively, we can draw the performance atlas for angular velocity and linear velocity as shown in Fig.3 and fig.4 based on 121 mechanisms' indices values of angular velocity and linear velocity. According to the rule that the bigger $\eta_J(J \in \{G_\omega, G_v\})$,the higher dexterity and controlling accuracy of the mechanism, from Fig.3 we can see that the mechanism performance index of angular velocity is not changing with the link length when the changing range of R is not big, but it has the trend that the bigger R, the better performance index of angular velocity, furthermore, the index of mechanism angular velocity is better when L=46.5cm~49.5cm and R=19.5cm, namely, the output error of angular velocity is smaller. Similarly, from Fig.4 we know that the mechanism index of linear velocity is better when L=45cm~48cm and R=19cm, that is to say, the output error of linear velocity is smaller.

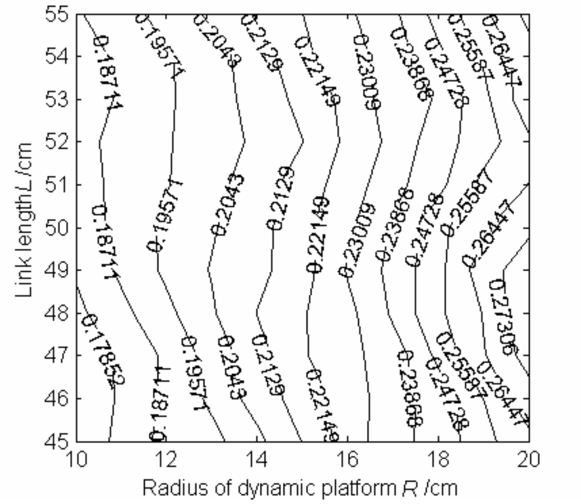


Figure 3. Atlas of angular velocity global performance

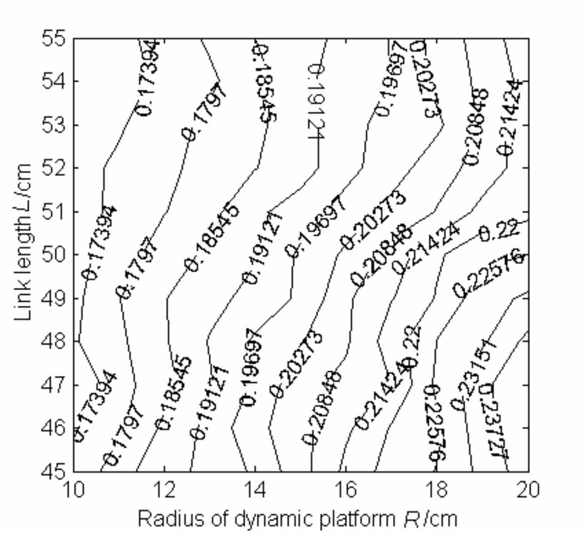


Figure 4. Atlas of linear velocity global performance

2.2.1.2 Global Performance Indices of Angular Acceleration and Linear Acceleration

Considering the influences on acceleration of both the first and the second order influence coefficient matrix, the condition numbers of angular acceleration and linear acceleration for 6-DOF parallel mechanism are(Guo,2002; Guo & Huang,2002)

$$K_{G_\omega + H_\omega} = b\|G_\omega\|\|G_\omega^+\| + (a^2 + 2a)\|H_\omega\|\|H_\omega^+\| \tag{2}$$

$$K_{G_\upsilon + H_\upsilon} = b\|G_\upsilon\|\|G_\upsilon^+\| + (a^2 + 2a)\|H_\upsilon\|\|H_\upsilon^+\| \tag{3}$$

Where, a and b is error coefficient.

So the global performance indices of angular acceleration and linear acceleration for parallel engraving mechanism can be defined as

$$\eta_J = \frac{\int_W \frac{1}{K_J} dW}{\int_W dW} \tag{4}$$

Where $J \in \{G_\omega + H_\omega £\text{-} G_\upsilon + H_\upsilon\}$.

Supposed the mechanism error is smaller than 2%(that is, a=b=0.02), replacing the underlined part in algorithm PerformanceAnalysis with formula (4), we can draw the performance atlas for angular acceleration and linear acceleration as shown in Fig.5 and Fig.6. As same as the evaluating method for velocity performance index, from Fig. 5 we can see that the angle acceleration performance of mechanism is better when nearly L=45cm~47cm and R=16cm~20cm, output error is smaller accordingly. Among the 121 mechanism we studied, its maximum is 0.16399. By observing Fig.6 carefully, we know that performance of linear acceleration is better when nearly L=45cm~48cm and R=19.5cm, accordingly, output error should be smaller.

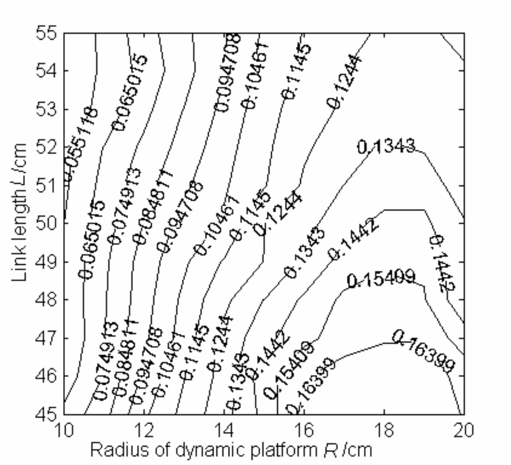


Figure 5. Atlas of angular acceleration global performance

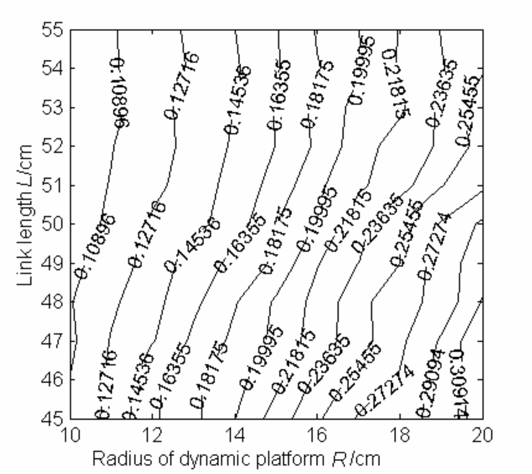


Figure 6. Atlas of linear acceleration global performance

From above analysis, we know that mechanism size with good indices for linear velocity and linear acceleration is coincidence in some degree among the 121 mechanisms we studied, but performance index of angular velocity and angular acceleration may not the best in the same size, so it can't get coincidence. Thus, our analysis will be helpful for designing and choosing mechanism by actual needs. Similarly, analyzing method of kinematics performance indices is the same when other parameters of the mechanism are changed.

2.2.2 Analysis of Dynamics Performance Indices

2.2.2.1 Analysis of Power and Moment Performance Index

The condition number of power and moment performance index based on the first order influence coefficient matrix of power G^F for 6-DOF parallel mechanism can be defined as

(Guo,2002)

$$K_{G^F} = \left\|G^F\right\|\left\|G^{-F}\right\| \tag{5}$$

Similarly, we define global performance index of power and moment for 6-DOF parallel mechanism as

$$\eta_{G^F} = \frac{\int_F \frac{1}{K_{G^F}} dW}{\int_F dW} \tag{6}$$

We suppose that power and moment of parallel mechanism is isotropy when η_J=1. With formula (5) as condition number, replacing the underlined part in algorithm PerformanceAnalysis with formula (6), we can draw the performance atlas for power and moment as shown in Fig.7. From Fig. 7 we can see in the size range of our experiment the performance index for power and moment would have little change with the link length when the radius of dynamic platform is less then 14cm. The performance index for mechanism's power and moment will be bigger when L=45cm~46cm and radius of dynamic platform R=10cm,here, performance of power and moment will be better.

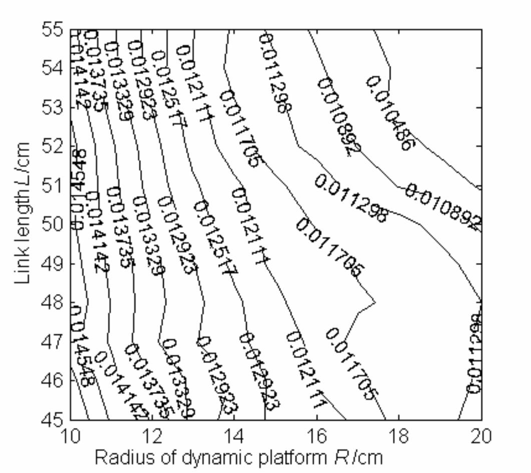


Figure 7. Atlas of global performance of force and moment

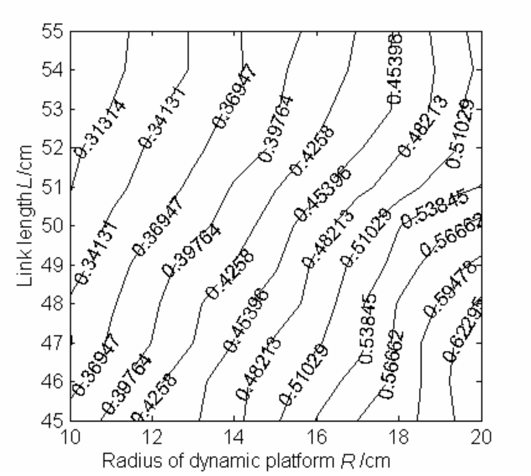


Figure 8. Atlas of global performance of inertia force

2.2.2.2 Analysis of Inertia Force Performance Index

Considering both the first and the second order influence coefficient matrix, the condition number of inertia force for 6-DOF parallel mechanism is defined as(Guo,2002)

$$K_{G+H} = \sqrt{6}((\|G\| + \|H\| + (\|[G_\omega]_1\| + \|[G_\omega]_2\| + \|[G_\omega]_3\|)\|G_\omega\|)) \tag{7}$$

Where $[G_{\omega}]_i$ is the ith column of matrix G_{ω}, i=1,2,3.
Then global performance index of engraving mechanism's inertia force can be defined as

$$\eta_{G+H} = \frac{\int_W \frac{1}{K_{G+H}} dW}{\int_W dW} \tag{8}$$

Obviously, the bigger value of η_{G+H}, the smaller inertia force of mechanism and the higher controlling accuracy. Replacing the underlined part in algorithm PerformanceAnalysis with formula(8), we can draw the performance atlas for inertia force as shown in Fig.8. According to the rule that the bigger value of η_{G+H}, the smaller inertia force of mechanism and the higher controlling accuracy, by observing Fig. 8 carefully, we can see that the inertia force performance index of mechanism is getting better when the link length is getting longer and radius of dynamic platform is getting bigger. Furthermore, the inertia force performance index of mechanism will be the best when nearly L=45cm~48cm and R=19.5cm, that is to say, the inertia force performance of mechanism is the best, inertia force is the smallest, sensitivity is the best and dexterity is the highest.
According to discusses above, we draw a conclusion that mechanism size with good performance index for power and moment and inertia force is coincidence not in all the time. This result indicates that the traditional designing and choosing mechanism method based on only the first order influence coefficient exists some restriction, we have to choose mechanism size on the basis of our analysis and actual demands.

2.3 The Results of Size Design and Summary

Summarizing previous analysis, we know that 6-PUS robot mechanism's all performance indices are better except force and moment when L=45cm~47cm and R=19cm, the actual size of mechanism with this type owned by laboratory is at the above-mentioned scope. We also find that its performances are same with the results of theory analysis by running the mechanism in deed, so prove the correctness of our theory. To validate our theory analysis further, we also do lots of simulations for this mechanism with other sizes. The results are same with those of theory analysis, so we can draw the conclusion that, in a generally way, there is not a mechanism whose all indices are better at the same time for both the kinematics and dynamics. In fact, we can only select the mechanism with relative better performance for all the indices to meet our need. On the basis for considering all the performance indices, the final sizes of novel parallel engraving machine that we designed are following:
The length of fixed link L is 46cm, the radius of dynamic platform R is 19cm, the radius of base platform is 38cm, $\{\varphi_{P1},\varphi_{P2},\varphi_{P3},\varphi_{P4},\varphi_{P5},\varphi_{P6}\}=\{45^{\circ},135^{\circ},165^{\circ},255^{\circ},285^{\circ},15^{\circ}\}$, $\{\phi_{B1},\phi_{B2},\phi_{B3},\phi_{B4},\phi_{B5},\phi_{B6}\}=\{82^{\circ},97^{\circ},202^{\circ},217^{\circ},322^{\circ},337^{\circ}\}$, $\varphi_C=30^{\circ}$, $\phi_A=15^{\circ}$. Where φ_{Pi} (i=1,2,…6)is the angle between tieline from dynamic platform's center o_p to C_i and the axis x_p's positive direction of dynamic platform's coordinate system, ϕ_{Bi} (i=1,2,…6)is the angle between tieline from base platform's center O_B to A_i and the axis X_B's positive direction of base platform's coordinate system. φ_C and ϕ_A is the smaller central angle of hexagon made by C_i and A_i (i=1,2,…6) respectively.

3. Research on Path Planning of Engraving

A series of engraving paths can be obtained after the image of workpiece to be engraved is dealt with image processing technology and tool path generating algorithm(Lo,1998), these paths consist of a series of straight line segments(exclusive for direction-parallel machining). When engraving on plane, the control of engraving path has the linear mapping relation to the cutting tool path that has been planned. But when engraving on spacial curved surface, the engraving path on plane should be mapped into the path on curved surface, this mapping is more complex than the former. In this section, we will pay attention to cutting tool path's mapping method from plane to spacial curved surface and the planning algorithm of cutting tool's posture.

3.1 Mapping Method of Cutting Tool Path

To describe conveniently, supposing the curved surface S to be engraved is

$$S = S(u,v) = \{x(u,v), y(u,v), z(u,v)\} \quad (u,v) \in D \tag{9}$$

Where D is one region on the uv plane.

Curve C on curved surface S can be expressed as

$$C = C(t) = S(u(t), v(t)) = \{x(t), y(t), z(t)\} \quad t \in [t_0, t_n] \tag{10}$$

Then arbitrary point Ci on curve C is

$$C_i = C(t_i) = \{x(t_i), y(t_i), z(t_i)\} = \{x_i, y_i, z_i\} \; (i = 0,1,\cdots,n) \tag{11}$$

In addition, supposing the image plane of workpiece to be engraved is superposition with coordinate system xOy that describes the curved surface S, then the plane equation describing workpiece image is

$$z = 0 \tag{12}$$

For one cutting tool path on plane with length of d, start and end point is Ps(x_1,y_1,0) and Pe(x_2,y_2,0) respectively, which should be mapped into a curve segment when engraving on spacial curved surface. Furthermore, the vertical projection's length of chord length corresponding to curved segment should equal with the straight line segment's length d. The line equation of plane cutting tool path's straight line segment can be expressed as

$$\begin{cases} x = x_1 + (x_2 - x_1)t \\ y = y_1 + (y_2 - y_1)t(-\infty < t < \infty) \\ z = 0 \end{cases} \tag{13}$$

The equation of plane P that passes plane cutting tool path's straight line segment and is vertical with image plane is

$$(y_2 - y_1)x - (x_2 - x_1)y - x_1 y_2 + x_2 y_1 = 0 \tag{14}$$

Then the intersecting line of plane P and curved surface S to be engraved is the curve C, on which is cutting tool path on curved surface crresponding to that on plane. By this, the start and end point's coordinates of cutting tool path on curved surface, which corresponding to that on plane, are Ps' (x_1,y_1,z_1) and Pe' (x_2,y_2,z_2) respectively, as x_1,y_1,x_2,y_2 are known, z_1 and z_2 can be deduced with equation (10). It is easy to validate that the cutting tool path's mapping method metioned above can suit for arbitrary quadrant of the coordinate system and engraving path in arbitrary direction.

3.2 Interpolation Computing of Spacial Curve

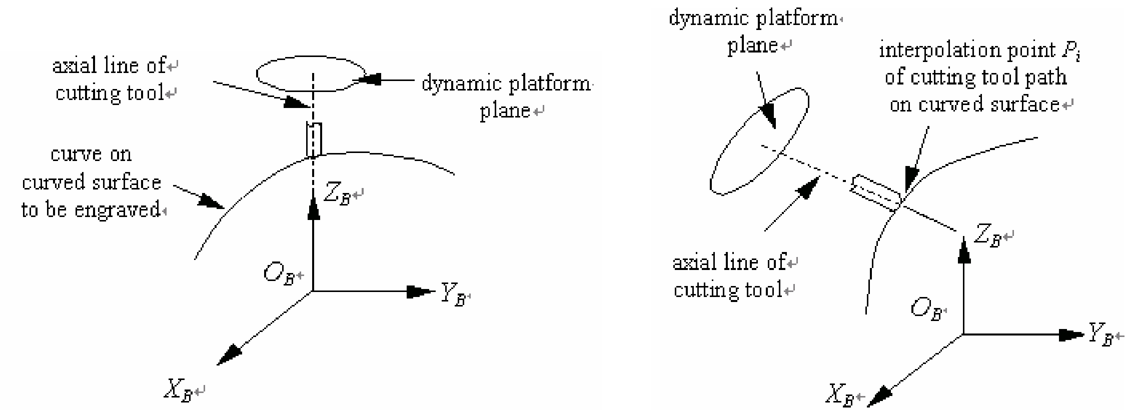
Real-time interpolation computing is necessary to engrave spacial curve segment, whose coordinates of start and end point are Ps' (x_1,y_1,z_1) and Pe' (x_2,y_2,z_2) respectively, with parallel engraving machine. Firstly, the interpolation number N should be calculated by interpolation precision, then the coordinates of ith interpolation point can be expressed as

(x_i, y_i, z_i), where $x_i = x_1 + \frac{x_2 - x_1}{N} i,\ y_i = y_1 + \frac{y_2 - y_1}{N} i. (i = 1, 2, \cdots, N)$, z_i can be deduced with equation (10). When i=N, it shows the terminal Pe′ of the curve segment. According to actual needs, many interpolation algorithms can be selected during engraving (Yang,2002).

3.3 Posture Planning of Cutting Tool

In view of the special structure of parallel mechanism and ensuring effect for engraving, the posture of cutting tool should be considered during engraving(Lee,1997), namely, the posture of cutting tool for every interpolation point should also be calculated besides its position during the control process. For parallel engraving machine with 6-DOF, the rotation around axis Z_B is not used in generally, the rotation angle α and β around axis X_B and Y_B will be solved in following.

Supposing the base coordinate system of parallel engraving mechanism in initial state is same with that of curved surface(curve), that is, the axial line of cutting tool of cutting tool is vertical with dynamic platform plane of engraving machine and fixed on it, then the rotation angles of dynamic platform plane around X_B and Y_B both are zero, namely, the angle between axial line of cutting tool and X_B and Y_B both are π/2, as shown in Fig.9(a).


axial line of cutting tool
dynamic platform plane
curve on curved surface to be engraved
Z_B
O_B
Y_B
X_B
dynamic platform plane
interpolation point P_i of cutting tool path on curved surface
axial line of cutting tool
Z_B
O_B
Y_B
X_B

(a) initial state (b) during engraving

Figure 9. The sketch map of cutting tool's posture for initial state and during engraving

Without losing generality, now taking arbitrary interpolation point P_i (x_i,y_i,z_i) on curve as sample, as shown in Fig.9(b), then the tangent plane (parallel with the dynamic platform plane of engraving machine) equation of curved surface through point P_i can be expressed as

$$\begin{vmatrix} x - x_i & y - y_i & z - z_i \\ x_u' & y_u' & z_u' \\ x_v' & y_v' & z_v' \end{vmatrix} = 0 \qquad (15)$$

Translating the general form as

$$Ax + By + Cz - (Ax_i + By_i + Cz_i) = 0 \qquad (16)$$

Where

$A = y_u' z_v' - y_v' z_u', B = z_u' x_v' - z_v' x_u', C = x_u' y_v' - x_v' y_u'$

In addition, the line equation of axis X_B and Y is as following respectively

$$y=z=0 \text{ and } z=x=0 \qquad (17)$$

According to the relation between spacial plane and line, the angle θ_x and θ_y, representing the angle between dyanmic platform plane of engraving machine and axis X_B and Y_B respectively, satisfy

$$\sin\theta_x = \frac{|Al_1 + Bm_1 + Cn_1|}{\sqrt{A^2 + B^2 + C^2} \cdot \sqrt{l_1^2 + m_1^2 + n_1^2}} = \frac{|A|}{\sqrt{A^2 + B^2 + C^2}} \tag{18}$$

$$\sin\theta_y = \frac{|Al_2 + Bm_2 + Cn_2|}{\sqrt{A^2 + B^2 + C^2} \cdot \sqrt{l_2^2 + m_2^2 + n_2^2}} = \frac{|B|}{\sqrt{A^2 + B^2 + C^2}} \tag{19}$$

In formula (18) and (19), l1;m1;n1 and l2;m2;n2 represent the parameters of symmetrical line eqution of axis X_B and Y_B respectively. Then we can get

$$\theta_x = \arcsin\frac{|A|}{\sqrt{A^2 + B^2 + C^2}}\text{;when A≥0 is }\theta_x\text{;otherwise }-\theta_x.$$

$$\theta_y = \arcsin\frac{|B|}{\sqrt{A^2 + B^2 + C^2}}\text{;when B≥0 is }\theta_y\text{;otherwise }-\theta_y.$$

Here, θ_x and θ_y represent the angle β and α, β and α is the rotation angle of dynamic platform plane rotating around axis Y_B and X_B respectively. Positive value denotes counterclockwise rotation, negative value denotes clockwise rotation. Thus, the posture parameter (α,β,0) of arbitrary interpolation point can be obtained, coupled with position parameter above, the position and posture of arbitray interpolation point on curve can be expressed as $P_i(x_i,y_i,z_i,\alpha,\beta,0)$. So it can be ensured that the axial line of cutting tool at arbitrary interpolation point will be vertical with the tangent plane of engraving point all the time.

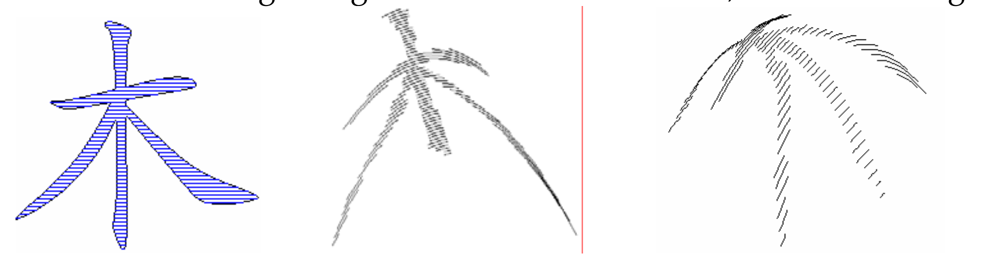
3.4 Summary of Tool Path Planning

On the basis of preceeding work, table 3 gives the corresponding relationship between interpolation points of straight line cutting tool path on plane and that of curved surface in space.

		start point	interpolation point i	end point
cutting tool path on plane		$(x_1,y_1,0,0,0,0)$	$(x_i,y_i,0,0,0,0)$	$(x_2,y_2,0,0,0,0)$
cutting tool path on curved surface	tangent plane	$A(x-x_1)+B(y-y_1)+C(z-z_1)=0$	$A(x-x_i)+B(y-y_i)+C(z-z_i)=0$	$A(x-x_2)+B(y-y_2)+C(z-z_2)=0$
	position and posture	$(x_1,y_1,z_1,\alpha_1,\beta_1,0)$	$(x_i,y_i,z_i,\alpha_i,\beta_i,0)$	$(x_2,y_2,z_2,\alpha_2,\beta_2,0)$

Table 3. The corresponding relationship between cutting tool path on plane and curved surface

Based on the theory mentioned above, we simulate the cutting tool path on plane and spacial curved surface when engraving Chinese characters “mu”, as shown in Fig.10



(a)tool path on plane (b)tool path on parabolic cylinder (c)tool path on sphere

Figure 10. The sketch map of cutting tool path

This section mainly resolves some questions about parallel engraving machine engraves on arbitrary spacial curved surface which is expressed with parameter equation, then realizes the mapping of cutting tool path from plane to spacial curve surface in the workspace of engraving machine. In addition, questions about interference and error analysis are not discussed in this section, the reason is that these questions will be involved only considering the radius of cutting tool, these works will be done in the future.

4. Research on Force Control Technology of Engraving Machine

Control technology for engraving machine can be classified two types: free motion control and constrained motion control. The control when the cutting tool of engraving machine does not contact with the workpiece is called free motion control. It is called constrained motion control when there is interaction between the cutting tool and the workpiece. Position control can meet demands when cutting tool of engraving machine moves in free space. Not only position but also contact force between cutting tool and environment should be controlled during engraving job. The aim of force control is to realize precise operation, but the first location in large scope is achieved by position control. As position control is relative simple and mature, this section will pay attention to the research on the technology of force control. At present, force control technology often used can be classified two types of basic and advanced. Basic force control technology also has two schemes: impedance control(Hogan,1985) and hybrid force/position control (Raibert & Craig,1981). Where, impedance is one force control technology developed in early stage, which does not control the force between machine and environment directly but control the dynamic relationship between force and position (or velocity) to implement force control. It is called impedance control because this dynamic relationship is similar to the concept of impedance in circuit. Hybrid force/position control is a scheme that implements force control and position control in “force subspace” and “position subspace” respectively, although its theory is definite, it is difficult to be realized. Basic control method of engraving force is simple and feasiable, it also can achive better control effect when the enviroment parameters are invariable. The precision of basic force control technology depends on the mathematical model of the engrving machine totally as this method’s patameters are invariable. We have deduced the accurate dynamics model of engraving machinism in (Kong et al., 2004) as follow

$$\mathrm{SF=K} \tag{20}$$

Where

$$S=\begin{bmatrix} {}^{P}s^{1} & {}^{P}s^{2} & {}^{P}s^{3} & {}^{P}s^{4} & {}^{P}s^{5} & {}^{P}s^{6} \\ {}^{P}r^{1}\times{}^{P}s^{1} & {}^{P}r^{2}\times{}^{P}s^{2} & {}^{P}r^{3}\times{}^{P}s^{3} & {}^{P}r^{4}\times{}^{P}s^{4} & {}^{P}r^{5}\times{}^{P}s^{5} & {}^{P}r^{6}\times{}^{P}s^{6} \end{bmatrix}$$

$$F=\left(\left\|\frac{{}^{P}F_{d1}^{1}}{\left({}^{P}s^{1}\right)^{T}\,{}^{P}b^{1}}\right\| \quad \left\|\frac{{}^{P}F_{d1}^{2}}{\left({}^{P}s^{2}\right)^{T}\,{}^{P}b^{2}}\right\| \quad \left\|\frac{{}^{P}F_{d1}^{3}}{\left({}^{P}s^{3}\right)^{T}\,{}^{P}b^{3}}\right\| \quad \left\|\frac{{}^{P}F_{d1}^{4}}{\left({}^{P}s^{4}\right)^{T}\,{}^{P}b^{4}}\right\| \quad \left\|\frac{{}^{P}F_{d1}^{5}}{\left({}^{P}s^{5}\right)^{T}\,{}^{P}b^{5}}\right\| \quad \left\|\frac{{}^{P}F_{d1}^{6}}{\left({}^{P}s^{6}\right)^{T}\,{}^{P}b^{6}}\right\|\right)^{T}$$

$$K=\begin{pmatrix} m\,{}^{P}\dot{v}+m\,{}^{P}g-{}^{P}F_{E}+6\sum_{n=1}^{3}m_{n}\,{}^{P}g+\sum_{i=1}^{6}\sum_{n=1}^{3}m_{n}\,{}^{P}\dot{v}_{cn}^{i} \\ {}^{P}I_{P}\,{}^{P}\dot{\omega}_{P}+{}^{P}\omega_{P}\times{}^{P}I_{P}\,{}^{P}\omega_{P}+\sum_{i=1}^{6}\left[{}^{P}r^{i}\times\sum_{n=1}^{3}m_{n}\left({}^{P}\dot{v}_{cn}^{i}+{}^{P}g\right)\right]-{}^{P}r_{E}\times{}^{P}F_{E}-{}^{P}M_{E} \end{pmatrix}$$

Please refer (Kong et al., 2004) for the meanings of all variables in formula (20). According to formula (20), the driving forces for appointed task can be solved when the kinematics parameters and engraving force and moment are known. But as the uncertainty of environment during actual engraving process, conventional method can’t be adjusted effectively when parameters are changed, so good control effect can't be obtained, then it is not suitable for the control of engraving machine.Along with the development of control technology, many researchers introduce advanced control technology to robot force control methods, which causes the appearance of advanced force control technology such as adaptive fuzzy control and neural network(NN) control etc. Fuzzy logic is similar to the thinking approach of human and more suitable for dealing with the uncertain and non-linear issues. Neural network is suitable for parallel computing and has the function of adaptive learning etc. Combining the merits of fuzzy control and neural network technology, fuzzy neural network(FNN) control makes the fuzzy control parameters and rules to be adjusted, modified and perfected automatically during control process, then the control performances of system can be improved constantly and the best control effect can be achieved finally. The control method of engraving force based on FNN technology proposed by us will be discussed in the following.

4. 1 The Structure and Learning Algorithm of FNN

We adopt five-layer NN to implement the fuzzy reasoning rules. The five layers are input layer, fuzzified layer, fuzzy association layer, fuzzy postassociation layer and output layer. In the five layers, the nodes of input layer is connected directly to the components of input vector and transports the input values to the next layer. Fuzzified layer completes the calculation of membership function and realizes the fuzzification of input variables, so every node of this layer corresponds to one linguistic variable. Every node of fuzzy association layer represents one fuzzy rule, which is used to match the antecedent of fuzzy rule and calculate the applicable degree of every rule. To given input, the value of linguistic variable only near the input point has bigger membership degree. Fuzzy postassociation layer completes normalized calculation, which prepares for the reasoning process of next layer. Output layer accomplishes defuzzification and outputs concrete value of the FNN. Generally speaking, NN should to be trained before practical application. For engraving machine, its engraving environment is not fixed commonly, the learning result of network for a certain environment is not suitable for other environment usually, so it needs retraining to adapt the new system. Therefore, online learning and adaptive ability is very important to the control of engraving force. Only the output layer has weight in our FNN, and only the local connect weights, which are connected with the activated neurons, are adjusted. The supervised Hebb learning rule(it is a learning algorithm without tutor) is adopted in this paper, which uses the deviation between desired force and actual force as supervised signal, and adjust the position by self-organizing and self-learning to environment information through associated searching, then realizes the tracking of desired engraving force finally.

4.2 The Design of Force Controller

Control framework of engraving force for parallel engraving machine based on FNN is shown in Fig.11, where the concrete structure of FNN is 2-14-49-49-1.

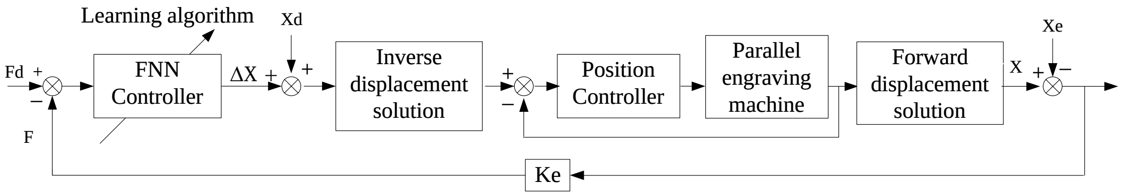

Learning algorithm
Xd
Xe
Fd
+
−
F
FNN Controller
ΔX
+
+
Inverse displacement solution
+
−
Position Controller
Parallel engraving machine
Forward displacement solution
X
+
−
Ke

Figure 11. Control framework of engraving force for parallel engraving machine based on FNN

In Fig.11, FNN controller uses force error F_e and error changing rate $\dot{F}_e$ as input value, position adjustment ΔX as output value where $F_e \in [-1000N,1000N], \dot{F}_e \in [-10000N/s,10000N/s], \Delta X \in [-0.01m,0.01m]$. F_d is desired force trajectory, F is the actual engraving force. K_e is stiffness matrix of engraving enviroment, which is commonly a positive diagonal matrix, that is to say, the eviroment is decoupled in all directions. X_d is desired position trajectory. X is the actual position of engraving machine's cutting tool. X_e is the position when the engraving enviroment is not deformed. Fuzzy subsets about input and output linguistic variable of fuzzy controller both are negtive large(NL), negtive middle(NM), negtive small(NS), zero(ZE), positive small(PS), positive middle(PM) and postive large(PL), the following bell function is used as membership function

$$\mu_{ij} = \exp\left\{-\frac{(x_i - \sigma_{ij})^2}{\delta_{ij}^2}\right\}, \quad (i = 1,2 \quad j = 1,2,\cdots,7) \tag{21}$$

Where σ_{ij} and δ_{ij} is the center and width of membership function respectively. Fuzzy control rule is shown in Table 4.

F_e $\dot{F}_e$	NL	NM	NS	ZE	PS	PM	PL
NL	NL	NL	NM	NL	NS	ZE	PS
NM	NL	NL	NM	NL	NS	PS	PM
NS	NL	NL	NM	NL	PS	PM	PL
ZE	NL	NL	NS	ZE	PM	PL	PL
PS	NL	NM	NS	PL	PL	PL	PL
PM	NM	NS	ZE	PL	PM	PL	PL
PL	NS	NS	PS	PL	PL	PL	PL

Table 4. Fuzzy control rule

Fig.12 to Fig.19 are the simulation results in one direction of degree-of-freedom (which can represent arbitrary direction) based on the control method mentioned above. Where dashed line shows desired engraving force, solid line shows actual engraving force.

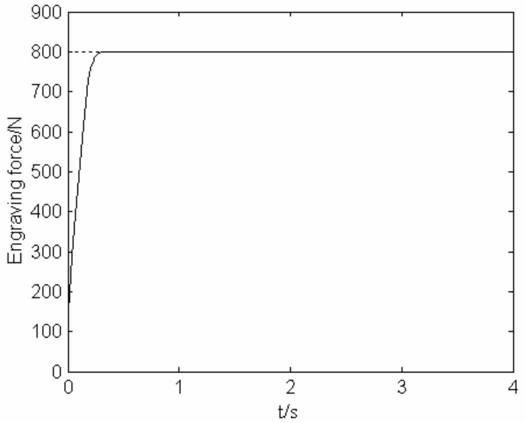


Figure 12. Response of step force with 800N

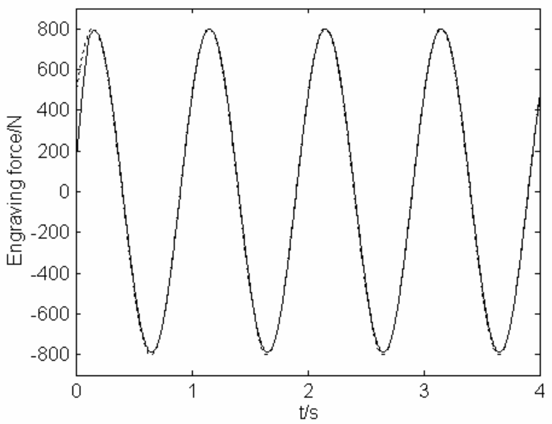


Figure 13. Response of time-changing force with 800sin(2πt+π/5)

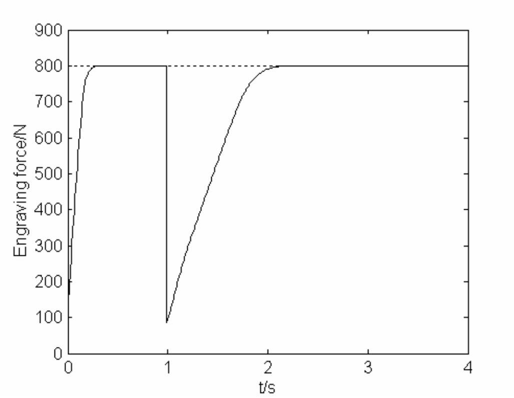


Figure 14. Response of step force under environmental stiffness changing (when t=1, Ke changing from 10000N/m to 1000N/m)

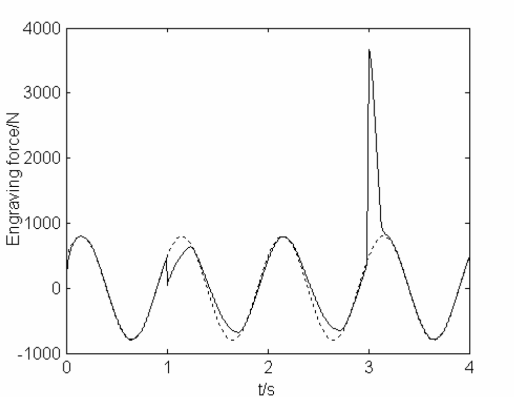


Figure 15. Response of sine force under environmental stiffness changing (when t=1, Ke changing from 10000N/m to 1000N/m, and t=3, Ke changing from 1000N/m to 10000N/m)

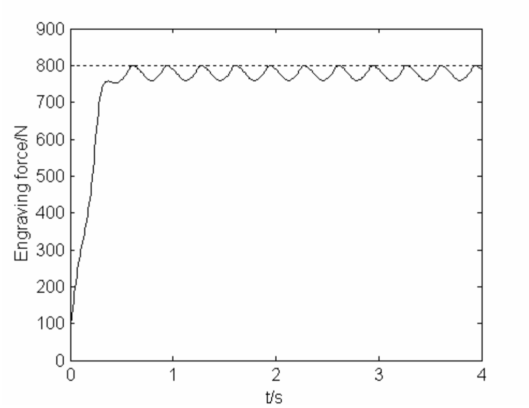


Figure 16. Response of step force under environmental position changing (Xe= $0.005\left|\sin(3\pi t + \frac{\pi}{4})\right|$)

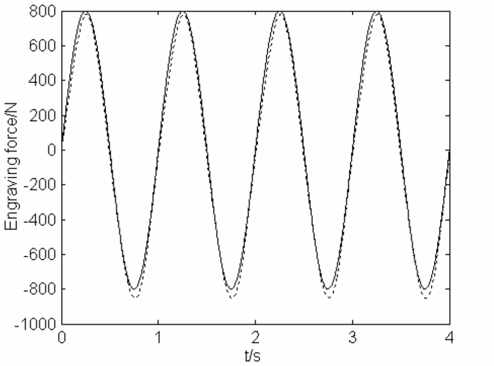


Figure 17. Response of sine force under environmental position changing Xe= $0.005\left|\sin(3\pi t + \frac{\pi}{4})\right|$)

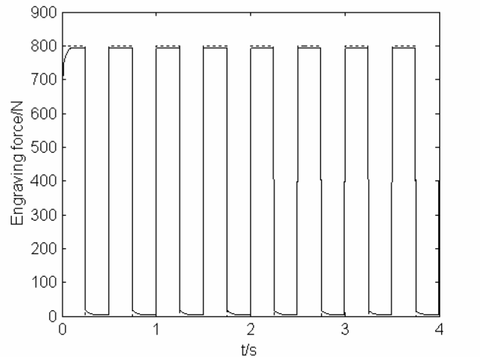


Figure 18. Response of squarewave force without interference

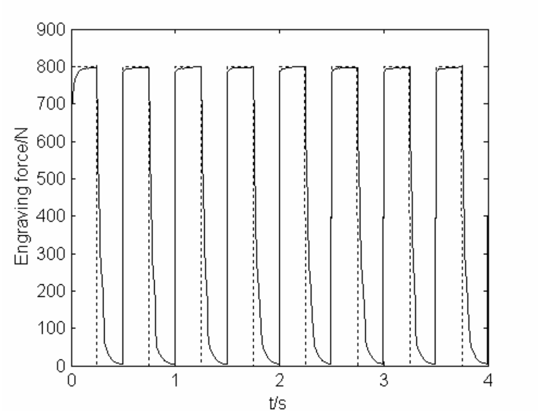


Figure 19. Response of squarewave force with noise(0,10)

From above results, we draw the conclusion that the control technology of engraving force based on FNN can make adjustment rapidly according to the changing of environment parameters, the whole system has merits such as rapid response to new environment, small influence by changing parameters, strong adaptability and robustness.

5. Research on Teleoperation Technology for Parallel Engraving Machine Based on B/S Mode

5.1 Implementation of Teleoperation System

The structure of teleoperation system for parallel engraving machine is shown in Fig.20, which adapts supervised control model(Huang & Mak,2001). This kind of control model makes operator out of the closed loop for control structure, so the influence on the whole system by transmission delay is decreased. Three parts functions, i.e. motion control, engraving control and video feedback, are provided to the use by web server. Motion control mainly implements the motion of engraving machine such as straight line, circular arc, ellipse and pentagon etc. Engraving control implements related task about actual engraving, including input of information to be engraved, contour extraction, tool path generation, machining simulation and actual engraving, which is the control core of parallel engraving machine and decides the engraving effect. The running state can be monitored real-time by starting the function of video feedback during process of both motion control and engraving control. All functions mentioned above can be shared by multiuser except controlling the engraving machine.

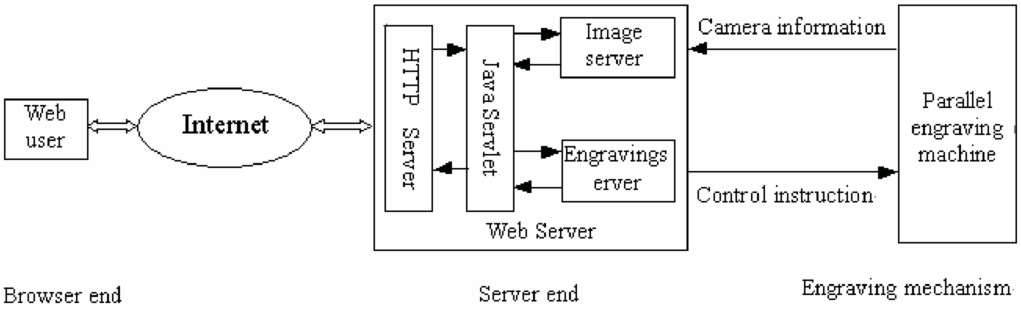


Figure 20. The structure of teleoperation system

The mutual exclusion principle between progress is adopted in our system to avoid sharing engraving machine by multiuser at the same time. The detailed method can be described as follow. A common variable is used as semaphore, every web user will test the state of this variable before actual engraving. “0” represents that the engraving machine can be controlled, “1” represents does not and now the related prompting message such as “system is busy” will be send to user immediately. Once the engraving machine is controlled by one user, the common variable will be assigned to “1” and other users will not use the machine, so the reliability of system can be guaranteed. Engraving task may be canceled by the user for all kinds of reasons during engraving process, which asks the control algorithm for engraving machine have the ability to respond new instruction during control. This question can be resolved by setting a common variable like above, what the control program should do is to test the value of this variable every time before sending control instruction to engraving machine. In addition, to ensure that only the user who submits the job has qualification to terminate engraving task, the system will return a unique ID to the browser end after engraving task is submitted successfully. The system will ask the user to submit a ID when user want to terminate engraving task and will compare the ID provided by user with the ID of task being executed. The engraving job will be terminated if the two IDs are coincident, otherwise the system will warn that the user has not the privilege to terminate the task. The Java Media Framework(JMF) is adopted for real-time transmission of video information. JMF is the application program interface for streaming media developed by SUN, through which the multimedia function is introduced to Java. As Remote Transport Protocol(RTP) has been implemented in JMF, the real-time acquisition and transmission of video information can be realized with JMF easily.

5.2 Experiments for Teleoperation

To verify the feasibility of our system, the engraving of straight line, circle, ellipse and pentagon is implemented successfully firstly, on this basis, the engraving of Chinese characters also successes by visiting the engraving server through local area network. In addition, remote engraving and real-time transmission for video information is obtained success further by visiting server through Internet. Fig.21 shows the system interface for engraving Chinese characters “zi”. Fig.22 shows the parameters to be submitted for straight line motion of engraving machine, here the “engraving control” part on the right side of Fig.21 will change into the content of Fig.22, similar as “circular arc motion” and other functions.

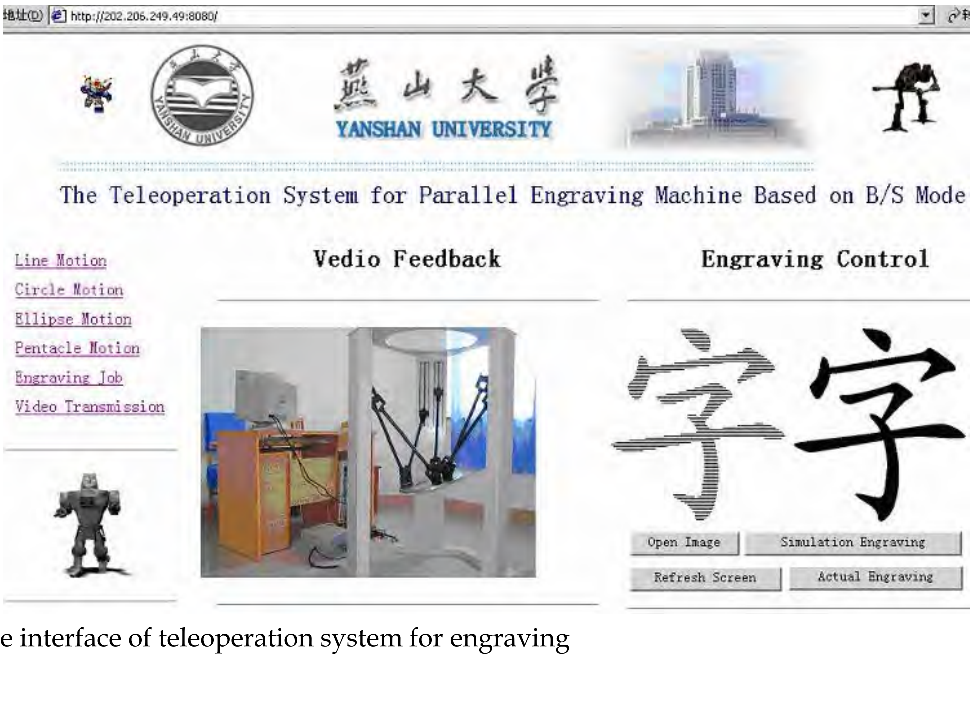
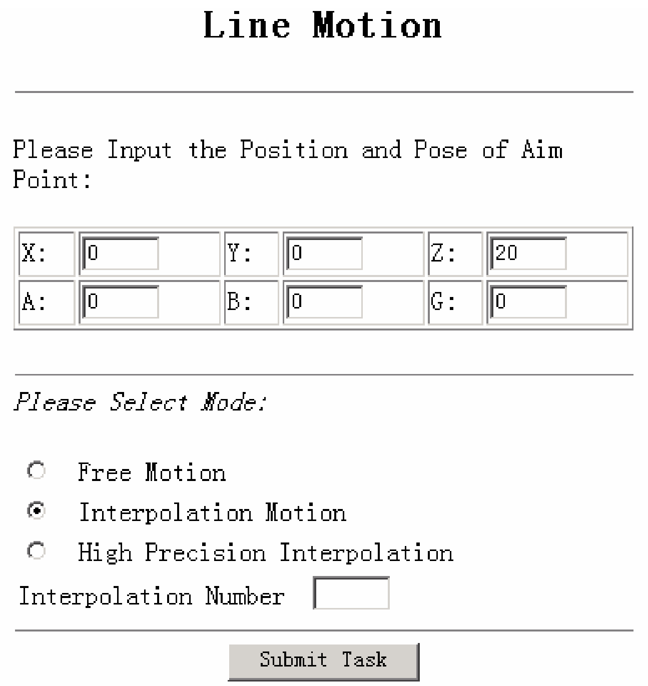


Figure 21. The interface of teleoperation system for engraving



Line Motion

Please Input the Position and Pose of Aim Point:

X:	0	Y:	0	Z:	20
A:	0	B:	0	G:	0

Please Select Mode:

○ Free Motion

● Interpolation Motion

○ High Precision Interpolation

Interpolation Number □

Figure 22. The submitted parameters for straight line motion

6. Conclusions

A novel parallel engraving machine is proposed and its some key technologies are studied in this paper. Based on the confirming of mechanism type, a group of mechanisms are obtained by changing the sizes of engraving machine. Performance indices are analyzed by considering both the first and the second order influence coefficient matrix of different sample point in every mechanism's workspace, then mechanism's sizes better for both kinematics and dynamics are achieved, so the theory basis for designing the size of novel engraving machine is established. In addition, method for tool path planning and control technology for engraving force is also studied in the paper. The proposed algorithm for tool path planning on curved surface solves path's mapping method from plane to spacial curved surface, which can be applied to arbitrary curved surface in theory. Control technology for engraving force based on FNN also has strong adaptability to the unknown environment. The implementation of teleoperation for parallel engraving machine based on B/S mode can meet the demands of distance engraving for ordinary web user, which also can be expanded to other machining domain besides engraving, so long distance and network machining can be realized. Simulation experiments are done based on the proposed model of engraving machine and related methods, simulation results show the feasibility and validity of the proposed methods. Beneficial exploration has been done for further design and development of future novel engraving machine in the paper.

7. References

Guo,X.J.(2002). Research on Dynamical Basal Theory of Parallel Robotic Mechanism. Dissertation for the Doctoral Degree in Engineering of Yanshan University,China

Guo, X.J. & Huang,Z.(2002). Analysis for Acceleration Performance Indices of Parallel Robots. China Mechanical Engineering,13(24): 2087-2091,ISSN 1004-132X

Huang,G.Q.&Mak, K.L.(2001).Web-integrated Manufacturing: Recent Developments and Emerging Issues. International Journal of Computer Integrated

Manufacturing,14(1):3-13,ISSN 0951-192X

Huang,Z.; Kong,L.F. & Fang, Y. F.(1997). Mechanics Theory and Control of Parallel Robot. Publishing House of Machinery Industry, ISBN 7-111-05812-7, Beijing, China

Hogan ,N.(1985). Impedance Control: An Approach to Manipulation: Part I-theory. Part II-Implementation. Part III-Applications. ASME Journal of Dynamic Systems, Measurement, and Control, 107:1-24, ISSN 0022-0434

Kong, L.F.; Zhang,S.H.; Xiao,W.H..; Li, C.Y.& Huang, Z.(2004). Rigid Body Dynamics Model of the 6-PUS Parallel Mechanism Based on Newton-Euler Method. Robot, 2004,26(5):395-399,ISSN 1002-0446

Lee, Y.S.(1997). Admissible Tool Orientation Control of Gouging Avoidance for 5-Axis Complex Surface Machining. Computer-Aided Design, 29(7): 507-521,ISSN 0010-4485

Lo,C.C.(1998)A New Approach to CNC Tool Path Generation. Computer-Aided Design, 30(8): 649-655,ISSN 0010-4485

Raibert, M.H. & Craig, J.J. (1981). Hybrid Position/Force Control of Manipulators. ASME Journal of Dynamic Systems, Measurement, and Control,102: 126-133, ISSN 0022-0434

Taylor,K.& Dalton,B.(2000). Internet Robots: a New Robotics Niche. IEEE Robotics & Automation Magazine,7(1):27-34, ISSN 1070-9932

Yang, X.N.(2002). Efficient Circular Arc Interpolation Based on Active Tolerance Control. Computer-Aided Design, 34(13): 1037-1046, ISSN 0010-4485

Yao, Y.X. & Lu Y.(1999). Research on Remote Working Condition Information Integration Based on Computer Network. Aeronautical Manufacturing Technology,42(1):13-17,ISSN 1000-8756

Pose Estimating the Human Arm Using Kinematics and the Sequential Monte Carlo Framework

Thomas Moeslund

1. Introduction

Human Motion Capture (MoCap) has for several years been an increasing activity around the world. The primary reason being the number of applications where MoCap is a necessity. Most well know is perhaps the application areas of Human Computer Interaction (HCI), surveillance, diagnostics of orthopaedic patients, analysis of athletes, and computer graphics animations such as the motion of the dinosaurs in ″Jurassic Park″ and Gollum in ″Lord of the Rings″. Different MoCap technologies exist but marker-free computer vision-based MoCap is of special interest as it can provide a ″touch-free″ and one-sensor (monocular) technology, see (Moeslund & Granum, 2001) for a survey.

Computer visions-based MoCap systems use different number of cameras, where, generally speaking, the fewer cameras the more complex the task. When more cameras are applied one of two overall techniques are applied. Either the position of the individual body parts are found in each image and combined into 3D-positions using triangulation (Azoz *et al.*, 1998); (Wren *et al.*, 2000), or a disparity map of the image is produced and compared with a 3D model of the human (Fua *et al.*, 1998);(Plankers *et al.*, 1999). A hybrid approach is to have multiple cameras but only use the one with the 'best view' to process the current image (Ong & Gong, 1999).

When just one camera is present a geometric model of the human is used to solve the ill-posed problem of estimating a 3D pose based on a sequence of 2D images. The model is either used directly in the estimation or it is used indirectly. By the latter is meant that the model guides the estimation process, as opposed to the former where the model is an integrated part of the estimation process. For example, in the work by (Segawa *et al.*, 2000) a particular pose of an arm is estimated by filtering past and future image measurements of the arm. The filtering is done using an Extended Kalman Filter wherein the joint angle limits of the model are incorporated. The model is therefore not used directly but rather indirectly to constrain the possible solutions.

In the case of direct model use, the model is synthesised into the image where it is compared to the image data. This process is known as the analysis-by-synthesis (AbS) approach. The type of data used in the matching differ between systems, but usually it is: edges (Sminchisescu, 2002);(Wu *et al.*, 2003), texture (Sidenbladh *et al.*, 2000);(Ben-Arie *et al.*, 2002), contours (Ong & Gong, 1999);(Delamarre & Faugeras, 2001), silhouettes (Moeslund & Granum, 2000);(Ogaki *et al.*, 2001), or motion (Bregler & Malik, 1998);(Howe *et al.*, 2000).

The framework used to decide which synthesised data to match with the current image data also differ. Usually a very high number of different model poses exist and an exhaustive matching is seldom realistic. Different approaches are therefore applied. One approach is to synthesise just one model pose and let the difference between the synthesised data and the image measurements be used as an error signal to correct the state of the model (Goncalves *et al.*, 1995); (Wren *et al.*, 2000). An excellent framework for this type of approach is the Kalman Filter.

Another approach is to formulate the matching as a function of the different parameters in the model. This function is then optimised, i.e., different model poses are synthesised until the synthesised data and image data is close with respect to some metric. Due to the high dimensionality of the problem an analytic solution is not possible and a numeric iterative approach is generally used (Gavrila & Davis, 1996);(Wachter & Nagel 1999). Within the last five years or so stochastic approaches have been the preferred ways of handling the dimensionality problem. Especially, the Sequential Monte Carlo (SMC) framework has been used successfully (Isard & Blake, 1998);(Deutscher *et al.*, 2000);(Mitchelson & Hilton, 2003);(Lee & Cohen, 2004). The SMC framework is basically a Bayesian approach where the current state of the model is represented by the posterior information inferred from predicted the most likely states from the previous frame and validating them using the current image measurements.

Independent of how the pose estimation problem is formulated the AbS-approach results in a huge solution-space. Therefore kinematic constraints are often applied in order to prune the solution-space, e.g., the bending of the elbow is between 0 and 145$^{\text{o}}$. This may be used directly to partition the solution-space into legal and illegal regions, as in (Segawa *et al.*, 2000), or the constraints may be defined as forces acting on an unconstrained state phase (Wren *et al.*, 2000). The fact that two human body parts cannot pass through each other also introduces constraints. Another approach to reduce the number of possible model poses is to assume a known motion pattern - especially cyclic motion such as walking and running. In the work by (Rohr, 1997) gait parallel to the image plane is considered. Using a cyclic motion model of gait all pose parameters are estimated by just one parameter, which specifies the current phase of the cycle. This is perhaps the most efficient pruning ever applied! (Ong & Gong, 1999) map training data into the solution-space and use PCA to find a linear subspace where the training data can be compactly represented without loosing too much information. (Pavlovic *et al.*, 1999) take this idea a step farther by learning the possible or rather likely trajectories in the state space from training data, i.e., dynamic models are learned.

1.1 The Content of this Paper

From a HCI point of view the primary interest regarding MoCap is a reliable way of pose estimating the arms over time. The focus of this work is therefore on pose estimating a human arm using monocular computer vision. The approach we take is twofold. First of all we want to apply the image measurements from the current frame in order to 1) derive a more compact representation of the solution-space and 2) improve the SMC framework. Secondly, we want to exploit the kinematic constraints more thoroughly than what is usually the case. The hypothesis behind the approach is that the fewer possible solutions (the result of a compact solution-space and thorough constraints) and the better the search (SMC) through these, the higher the likelihood of actually finding the correct pose in a particular frame.

The structure of the paper is as follows. In section 2 a new way of representing the human arm is presented. In section 3 this model is pruned by exploiting the kinematic constraints. In section 4 the model of the arm is applied in a SMC framework in order to pose estimate an arm. In section 5 results are presented and in section 6 a conclusion is given.

2. Modelling the Arm

When modelling the pose of the arm it is necessary to understand the anatomic features, which controls the movement of the arm, hence the bones and the joints connecting them. A complete description of the interacting between the joints and bones and the DoF this results in, leads to a comprehensive model that again results in a huge solution-space. This is not desirable and we therefore focus on the large-scale motion of the arm meaning that only the most significant DoF in the arm are modelled. Furthermore, we assume that a geometric ideal joint can model each anatomic joint. Another consequence of the focus on large-scale motion is that we assume the hand to be a part of the lower arm.

Overall the arm consists of the upper- and lower arm; those lengths are assumed known as A_u and A_l, respectively. The length of the lower arm is defined as the distance from the elbow to the centre of the hand. As for the rest of the movable rigid entities in the human body, the arm consists of bones that are moved by muscles following the design of the joints connecting the bones. The ligaments, muscles, joints, and other bones impose the limits of the movements, but in this work we simply refer to the limits of a joint independent of the origin.

The lower arm consists of two bones; the ulna and the radius (Morrey, 1985), see figure 2. They are connected to each other with a pivot joint at the elbow and with a pivot joint at the wrist. These two joints allow the hand to rotate around the long axis of the lower arm by rotating the radius around the ulna. The pivot joint in the elbow is part of the elbow joint that also connects the two bones with the upper arm bone, the humerus. The ulna is connected in a hinge joint and the radius in a ball-and-socket joint. Overall the primary DoF at the elbow is modelled very well using one hinge joint and one pivot joint even though the elbow motion is more complex (An & Morrey, 1985). Since we ignore the motion of the hand we can ignore the pivot rotations of the radius around the ulna, hence the DoF in the elbow is modelled by one hinge joint.

The upper arm consists of one bone, the humerus, and is connected to the shoulder in the gleno-humeral joint (GH-joint), see figure 2. Even though the GH-joint contains two sliding DoF (Dvir & Berme, 1978) it is well modelled as a ball-and-socket joint since the surfaces of the joint is more than 99% spherical (Soslowsky *et al.*, 1999).

The "socket" part of the joint is a part of the shoulder and called the glenoid. Its motion with respect to the torso, or more precisely the thorax (Zatsiorsky, 1998), originates from the complex structure of the shoulder, known as the shoulder complex. The shoulder complex provides mobility beyond any other joint in the entire human body (Zatsiorsky, 1998). The shoulder complex contains up to 11 DoF (Maurel, 1998). However, since the mobility of the shoulder complex can be described as a closed kinematic chain these 11 DoF are not independent and in fact the relative pose of the glenoid is well described by only four independent parameters (Zatsiorsky, 1998).

To model the shoulder complex requires a comprehensive biomechanical model based on knowledge of the bones, joints, ligament, muscles, and their interactions. Such models have been developed, see e.g., (Engin & Tumer, 1989);(Hogfors, 1991); (Maurel, 1998).

We can however not use such models since they contain too many details, hence too many parameters. However, by analysing the outcome of advanced biomechanical models we observe that the primary motion of the glenoid with respect to the torso, hence the four parameters, is: rotations around the z- and y-axes, and vertical and horizontal displacements along the y- and x-axes, see figure 1 for a definition of the axes. The rotations can be governed by the GH-joint by increasing its ranges accordingly whereas two prismatic joints can model the two displacements, each having one DoF.

Altogether six DoF are needed to model the primary DoF of the arm and the shoulder complex. As the prismatic joints have significantly less effect on the pose of the arm compared to the elbow and GH-joints we (for now) ignore the prismatic joints and focus on the remaining four primary DoF.

2.1 Modelling the four DOFs of the Arm

A number of different ways of modelling the four DOF in the arm exist (Moeslund, 2003). The most common model is the one shown in figure 1 where four Euler angles are applied. Since we aim at a compact state-space we derive a new model inspired by the screw axis model.

The parameters in the screw axis model do not directly relate to the anatomic joints. Nevertheless the model has the same ability to represent the different arm configurations as for example the Euler angle model has. The representation is based on Chasles' theorem (Zatsiorsky, 1998) that loosely states that a transformation between two coordinate systems can be expressed as a rotation around an axis, called the screw axis (or helical axis), and a translation parallel to the screw axis. In the context of modelling the human arm the screw axis is defined as the vector spanned by the shoulder and the hand. The position of the elbow is defined as a rotation of an initial elbow position around the screw axis. Since the length of the upper and lower arm are fixed no translation is required parallel to the screw axis and the perpendicular distance from the elbow to the screw axis is independent on the rotation and can be calculated without adding additional parameters. Altogether the representation requires four parameters; three for the position of the hand and one for the rotation around the screw axis, α. In figure 1 the parameters are illustrated.

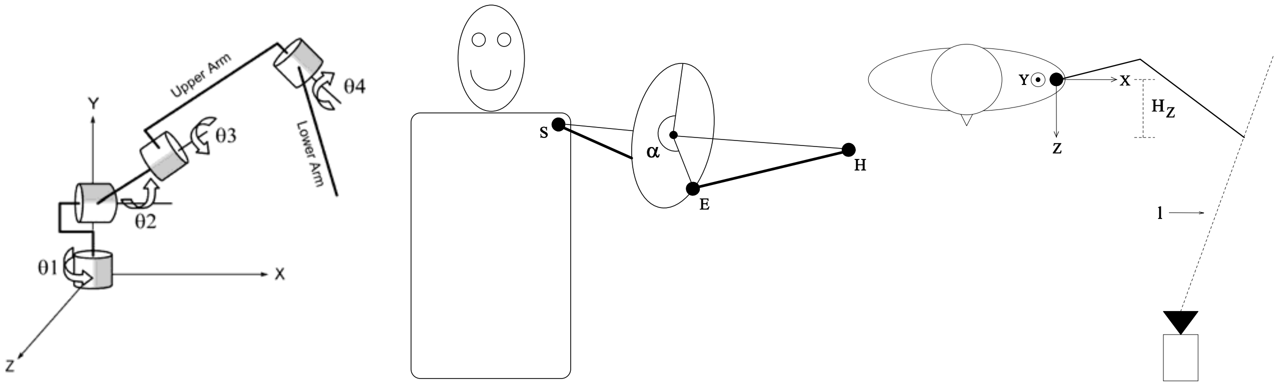


Figure 1. Different arm representations. **Left**: Four Euler angles. **Middle**: Screw axis model. **Right**: Local screw axis model

We now combine the screw axis model with image data in order to obtain a more compact representation of the arm. Colour information is used to segment the hand in an image. Combining this with the camera parameters obtained during calibration the position of the hand in an image can be mapped to a line in space:

$$\vec{H}(t) = \vec{P} + t \cdot \vec{D} \tag{1}$$

where $\vec{P}$ is a point on the line, e.g., the focal point of the camera, and $\vec{D}$ is the unit direction vector of the line. This means that for each value of H_z the two other components H_x and H_y are uniquely determined. Applying this to the screw axis representation we obtain a "local screw axis model" utilising only two parameters, α and H_z, to model the pose of the arm. For each new image a unique instance of the solution-space exist, hence the name "local". In this compact model α□is bounded by one circle-sweep [0^o , 360^o], while H_z is bounded by ± the total length of the arm, A_u+A_l. Using only two parameters to model the arm is indeed a compact representation. Furthermore, having only two parameters also allows for a visualisation of the result when comparing the synthesised data with the image data, hence the solution-space can be directly visualised - a very powerful characteristic!

2.2 Eliminating the Effect of the Prismatic Joints

Previously we suggested using six parameters to model the arm: two for the shoulder complex, three for the GH-joint, and one for the elbow joint. Next we suggested ignoring the prismatic joints by arguing that they have significantly less effect on the pose of the arm compared to the elbow and GH-joints. Even though this is true the prismatic joints should not be ignored altogether. In this section we therefore revise the prismatic joints and include them in our local screw axis model.

Concretely we seek to eliminate the effect of the prismatic joints altogether by estimating the displacements of the glenoid in each image and correcting the shoulder position accordingly. This elimination allows for a more compact model describing the pose of the arm, hence two parameters modelling the six DoF.

The idea is to find a relationship between the displacements of the GH-joint[1] (denoted Δv and Δh) and the angle between the torso and the upper arm, ϕ. For each image ϕ is estimated based on the position of the hand in the image and Δv and Δh are estimated. In section 2.2.1 we show how to estimate the relationship between ϕ and Δv and how to estimate the relationship between the position of the hand and ϕ.

2.2.1 Relating ϕ and Δv

To understand the relationship between ϕ and Δv the shoulder complex is investigated in more detail. The shoulder complex consists of two bones (the shoulder girdle); the scapula and the clavicle, see figure 2. The former is the large bone on the back of the shoulder and the latter is the one from the breastbone to the top of the shoulder tip (Codman, 1934). The clavicle is a long thin bone connected to the breastbone in the sterno-clavicular joint. It functions as a ball-and-socket joint and is denoted the SC-joint, see figure 2. In the other end the clavicle is connected to the scapula in the acromio-clavicular joint that also functions as a ball-and-socket joint. This joint is denoted the AC-joint. The scapula is a large triangular bone, which contains the glenoid below the AC-joint. The scapula is not

[1] The exact location of the GH-joint is defined as the centre of the humerus head that is the point the humerus rotates about, hence its position is fixed with respect to the glenoid. This means that finding the displacement of the GH-joint is equivalent to finding the displacements of the glenoid. The reason for choosing the GH-joint over the glenoid is that the former has a more clearly anatomic definition.

connected directly to the thorax through a joint but instead via muscles. This results in a very flexible "joint" which can both rotate and translate with respect to the thorax.

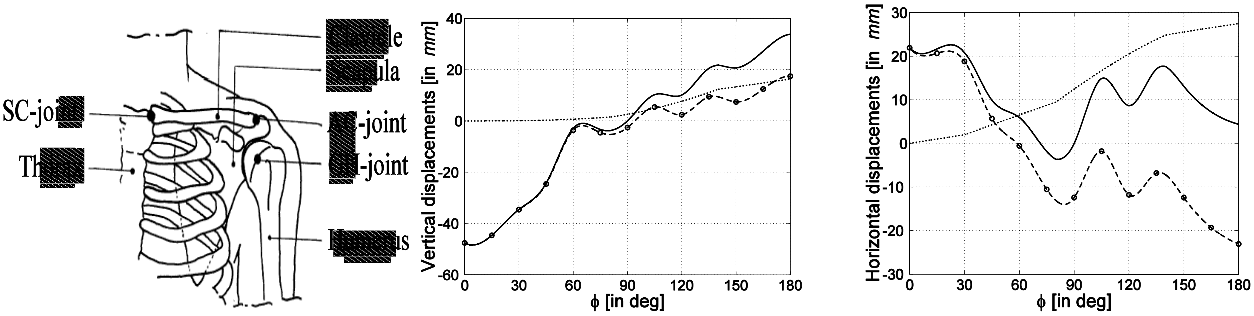


Figure 2. Left: The different bones and joints in the shoulder complex. Figure after (Breteler *et al.*, 1999). Middle and Right: The vertical and horizontal displacements, respectively, as a function of □. The dashed graphs are the displacements of the AC-joint due to the rotation in the SC-joint. The circles are the actual measurements from (Dvir & Berme, 1978) while the graphs are obtained by spline interpolations. The dotted graphs are the additional displacements of the GH-joint due to rotation in the AC-joint. The solid graphs are the total displacements of the GH-joint with respect to the SC-joint normalised to zero at $\phi = 90^{O}$

The value of Δv is the same as the vertical displacement of the GH-joint with respect to the resting position where $\phi = 0^{O}$ and originates from rotation in the SC-joint and the AC-joint. Actually, without the rotation in the SC-joint and AC-joint the elevation of the arm would be limited. When elevating the arm to 180^{O} only 120^{O} comes form the rotation in the GH-joint and the rest comes from the rotation of the scapula (Culham & Peat, 1993), hence in the SC-joint and the AC-joint.

The primary displacement of the GH-joint comes from the elevation of the AC-joint, hence the upward rotation in the SC-joint. In the work by (Dvir & Berme, 1978) measurements describing the displacement of the AC-joint as a function of ϕ□are presented, see the dashed graph in figure 2 (middle). To obtain a complete relationship we need to add Δy that expresses the vertical displacement of the GH-joint originating from the rotation in the AC-joint. We assume that the clavicle is horizontal when $\phi = 0^{O}$ and that the GH-joint is located *r mm* directly below the AC-joint. Δy, can then be calculated as $\Delta y = r\ (\ 1 - \sin(\tau))$ where $\tau = 270^{O} - \alpha - \beta$. α and β are the rotations in the SC-joint and AC-joint, respectively (Moeslund, 2003). *r* is approximately equal to 10.4% of the length of the upper arm (Leva, 1996). That is, $\mathrm{r} = \mathrm{A_U} \cdot 0.104$.

To be able to calculate Δy as a function of ϕ we need to known how α and β depend on ϕ. The relationship is via the rotation of the scapula, which is therefore investigated further. To structure our investigation we utilise the concept of the "shoulder rhythm" (Zatsiorsky, 1998) or "scapulohumeral rhythm" (Culham & Peat, 1993). The shoulder rhythm is defined as the ratio, *R*, of the upwards rotation in the GH-joint, η, and the angle between the scapula and torso, ψ, hence R = η / ψ.. Since $\phi = \eta + \psi$ we can apply the ratios reported in the literature to find the relationship between the rotation of the scapula and ϕ as $\psi = \dfrac{\phi}{\mathrm{R}+1}$

In average the ratio of an 180^{O} elevation of ϕ is 2:1 (Inman *et al.*, 1944). However, the ratio wary a lot during a 180^{O} elevation. Usually it is divided into four phases (Culham & Peat, 1993). From the ratios in these four phases ϕ can be calculated. Hence, the relationship between α and ϕ, and β and ϕ, respectively, can be derived and the following values of □ can be calculated (Moeslund, 2003)

$$\tau = \begin{cases} 90.0^{\circ} - 0.12\phi & \text{if } \phi \in \left[0^{\circ}, 30^{\circ}\right[\\ 94.8^{\circ} - 0.28\phi & \text{if } \phi \in \left[30^{\circ}, 80^{\circ}\right[\\ 119.2^{\circ} - 0.59\phi & \text{if } \phi \in \left[80^{\circ}, 140^{\circ}\right[\\ 68.5^{\circ} - 0.22\phi & \text{if } \phi \in \left[140^{\circ}, 180^{\circ}\right] \end{cases} \tag{2}$$

In figure 2 (middle) $\Delta y(\phi)$ is shown as the dotted graph. The solid graph is the sum of the dashed- and dotted-graphs, hence $\Delta v(\phi)$. The relationship between ϕ□and □Δh which can be found in a similar manner (Moeslund, 2003) is illustrated in figure 2 (right).

After having related the two prismatic joints to ϕ we need to relate ϕ to the parameters of the local screw axis model. For H_Z this is done by projecting the predicted position of the hand onto the camera ray through the hand. For α this is done using prediction and inverse kinematics (Moeslund, 2003).

3. Kinematic Constraints

Even though the local screw axis model is very compact it still has a large solution-space. Therefore constraints are introduced to prune the solution-space.

When a human moves his/her arm two types of constraints ensure a plausible sequence of poses. These are kinematic constraints and dynamic constraints. The former is concerned with the position and all higher order derivatives of the position variable(s). These constraints are defined without regard to the forces that cause the actual motion of the arm, and are measurable in the image(s). The dynamic constraints on the other hand require knowledge about anatomic features that are not directly measurable, for example masses of bodies and strength of muscles. Furthermore, humans first consider the kinematics and then the dynamics when positioning the arm (Kondo, 1991). This suggests that the pruning effect from the kinematic constraints is dominant.

The principal kinematic constraints come from the limits on the joint angles, e.g., that the arm cannot bend backwards at the elbow, and are defined in anatomy. The actual values of these limits are, however, not universal and differ between individuals. In table 1 the limits for the first author is listed and in figure 1 the angles are illustrated.

	θ_1	θ_2	θ_3	θ_4
Minimum	-135°	-135°	0°	45°
Maximum	45°	100°	145°	180°

Table 1. The legal ranges of the different joint angles

Besides angle values also the angle velocity and acceleration yield constraints. Their ranges depend on the activity carried out. Here "normal" movements are considered with a maximum velocity of $400^{\circ}/\mathrm{s}$. It is assumed that a subject can accelerate the upper- and lower arm to their maximum angle velocity within one tenth of a second, i.e., the maximum acceleration is $4000^{\circ}/\mathrm{s}^2$.

In the following the limits on the joint angles and their derivatives together with geometric reasoning are applied to prune H_Z and α. The pruning is done through six constraints: four pruning H_Z and two pruning α, see (Moeslund, 2003) for a more detailed description of the constraints.

3.1 Pruning H_Z Using Static Constraints

First H_Z is pruned through three static constraints. The first constraint states that the distance between the 3D hand position and the shoulder needs to be less than the total length of the arm; hence a sphere limits the hand position. To calculate the allowed interval for H_Z we find where the line in equation 1 intersects the sphere, yielding $P_Z + t_2 \cdot D_Z \leq H_Z \leq P_Z + t_1 \cdot D_Z$, where t_1 and t_2 define the intersection points.

The second constraint is an angle constraint. When the position of the hand in the image is to the left of the shoulder (H_X > 0) H_Z can be pruned using the limits on θ_1. The limitations on the angle mean, among other things, that the upper arm can only rotate 45° backwards. Together with the limitations of the other angles the minimum H_Z positions of the hand is limited by θ_1 in the following way: $\tan(45^\circ) \geq \frac{H_Z}{H_X}$, where the angle is measured from the x-axis. Inserting the parametric equation of the line, equation 1, and isolating t allows us to calculate the smallest value of H_Z as $H_{Z,\min} = P_Z + t \cdot D_Z$

The final static constraint pruning H_Z is an occlusion constraint. When the position of the hand in the image is to the right of the shoulder $H_X \leq 0$ the hand is likely to be in front of the head or torso. If this is the case H_Z can be pruned by finding the intersection between the line, *l*, in equation 1 and a representation of the head and torso, respectively. The torso is modelled using an elliptic cylinder with its semi-axes (*a* and *b*) parallel to the *x* and *z*-coordinate axes shown in figure 1. The head is modelled as an ellipsoid with semi-axes *i, j,* and *k*, where *i*=*k* (Moeslund, 2003). The parametric line in equation 1 is inserted into both the equation of the ellipsoid and elliptic cylinder and solved with respect to *t*. If *t* is real and its *y*-value belongs to one of the shapes, the limits on H_Z is found as

$$H_{Z,\min} = P_Z + t \cdot D_Z$$

3.2 Pruning H_Z Using a Temporal Constraint

The final pruning of H_Z is based on a temporal constraint stating that the displacement of H_Z between two consecutive frames is bounded by the limits on the joint angles[2].

For each image the limits can be tightened using the previously estimated angle values together with the limits on the velocity and acceleration. Altogether a new set of legal intervals for the four angles is obtained, namely Φ_1, Φ_2, Φ_3, and Φ_4. Each interval is calculated as

$$\Phi = [\Phi_{\min}, \Phi_{\max}] = [\max\{\theta_{\min}, \theta^* + \Delta^- \theta\}, \min\{\theta_{\max}, \theta^* + \Delta^+ \theta\}] \qquad (3)$$

Where θ^* is the value of the joint angle in the previous frame, $\theta_{\min}$ and $\theta_{\max}$ are defined in table 1, and $\Delta^- \theta$ and $\Delta^+ \theta$ are defined in figure 3 (Moeslund, 2003).

In some situations, e.g., when $\theta^* = \theta_{\max} \wedge V^* > 200^O / s$ a sudden stop of the movement of the hand is required. Obviously, this is not realistic and a minimum interval wherein the hand is bound to be is defined to be 20^O. Equation 3 is therefore expanded to

[2] This is calculated for a frame-rate of 10Hz

$$\Phi = \begin{cases} \left[\theta_{\max} - 20^{\circ}, \theta_{\max}\right] & \text{if } \left(\Phi_{\max} - \Phi_{\min}\right) < 20^{\circ} \wedge \Phi_{\min} > \theta_{\min} \\ \left[\theta_{\min}, \theta_{\min} + 20^{\circ}\right] & \text{if } \left(\Phi_{\max} - \Phi_{\min}\right) < 20^{\circ} \wedge \Phi_{\max} > \theta_{\max} \\ \left[\Phi_{\min}, \Phi_{\max}\right] & \text{otherwise} \end{cases} \tag{4}$$

The maximum change of H_Z between two frames occurs when the arm moves in the (*y*=*0*)-plane and can be found as $H_Z^* - \delta \leq H_Z \leq H_Z^* + \delta$, where H_Z^* is the value of H_Z in the previous frame and δ is the maximum change of H_Z found by investigating the differential geometry of the arm within the current range of the joint angles Φ_1 and Φ_4 (Moeslund, 2003).

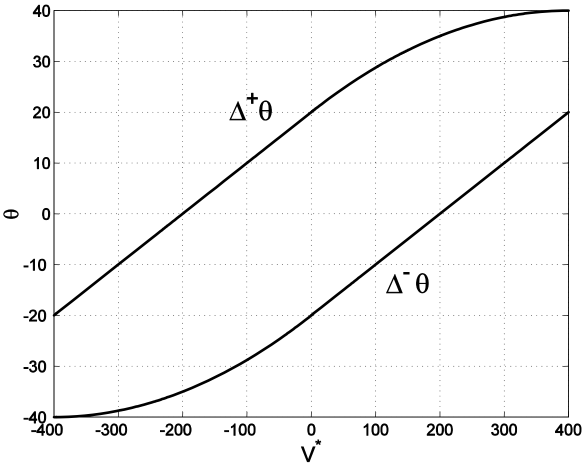


Figure 3. The upper and lower limits on the joint angles as a function of the velocity in the last frame, *V**

3.3 Pruning α Using Joint Angle Constraints

This constraint prunes α by mapping the legal intervals for the joint angles $(\Phi_1, \Phi_2, \Phi_3, \Phi_4)$ to α, Φ_α. To make the following description conceptually easier the arm configuration is viewed as a triangle spanned by the shoulder, elbow, and hand. In terms of angles within the triangle this will make a number of configurations equal. Φ_4 only apply in the interval [0^{O} , 180^{O}] and angles outside this interval will be constrained by Φ_3. Independently of α and the three other angles, θ4 needs to be within Φ_4. If not, the current value of H_Z can be ignored altogether, i.e., the entire range of α is pruned. The current value of□θ4 is calculated using the cosine-relation (Moeslund, 2003).

A relationship between the remaining three angles and α is found by defining a reference triangle, where $\vec{E} = (A_U, 0, 0)$ and $\vec{H} = (A_U + A_L \cdot \sin(\theta_4), -A_L \cdot \cos(\theta_4), 0)$, and explaining how to rotate it into the current triangle spanned by the shoulder, the elbow, and the hand. The rotation is described in two ways. One is by using the three joint angles R_θ and another using R(α). The two rotations will be equal yielding the relationship between the joint angles and α. The three angles, see figure 1, are Y-Z-X Euler angles, hence

$$R_\theta = R(\theta_1, \theta_2, \theta_3) = \begin{bmatrix} c1 \cdot c2 & -c1 \cdot s2 \cdot c3 + s1 \cdot s3 & c1 \cdot s2 \cdot s3 + s1 \cdot c3 \\ s2 & c2 \cdot c3 & -c2 \cdot s3 \\ -s1 \cdot c2 & s1 \cdot s2 \cdot c3 + c1 \cdot s3 & -s1 \cdot s2 \cdot s3 + c1 \cdot c3 \end{bmatrix} \tag{5}$$

where c1=cos(θ_1) and s2=sin(θ_2) etc. The other rotation, R(α), is obtained by performing a number of rotations. First, the shoulder-hand-line of the reference triangle is rotated to be aligned with the x-axis. Next, this line is rotated to be aligned with the current shoulder-hand-line, $\vec{H}$. This is done by first aligning the z-component and then the y-component. The shoulder-hand-line of the reference triangle is now aligned with the shoulder-hand-line of the current configuration. The final rotation is to align the position of the elbow in the reference triangle with the current elbow position. This is done using Rodriques' formula (Craig, 1989), which rotates the reference triangle around, $\vec{H}$ by α degrees. The resulting rotation matrix is quite complicated, but each entry can be expressed as: $a \cdot \sin(\alpha) + b \cdot \cos(\alpha) + c$ where *a, b,* and *c* are constants.

We can now use the fact that $R(\theta_1, \theta_2, \theta_3) = R(a)$ to calculate how the limits on the Euler angles can prune α. First we see how θ_2 prune α. We apply entry (2,1) yielding: $\sin(\theta_2) = a_{21} \cdot \sin(\alpha) + b_{21} \cdot \cos(\alpha) + c_{21}$

Looking at this equation as a function of α we can see that the right-hand side is a sine curve and the left-hand side is two straight lines; corresponding to the minimum and maximum allowed values of θ_2 defined by Φ_2. The equation can thou yield six different results, i.e., six different types of pruning intervals. These are illustrated in figure 4, where the shaded area (from zero to 360) illustrates the legal values for θ_2, and the shaded region(s) on the α-axis illustrates the legal α-values. For details see (Moeslund, 2003).

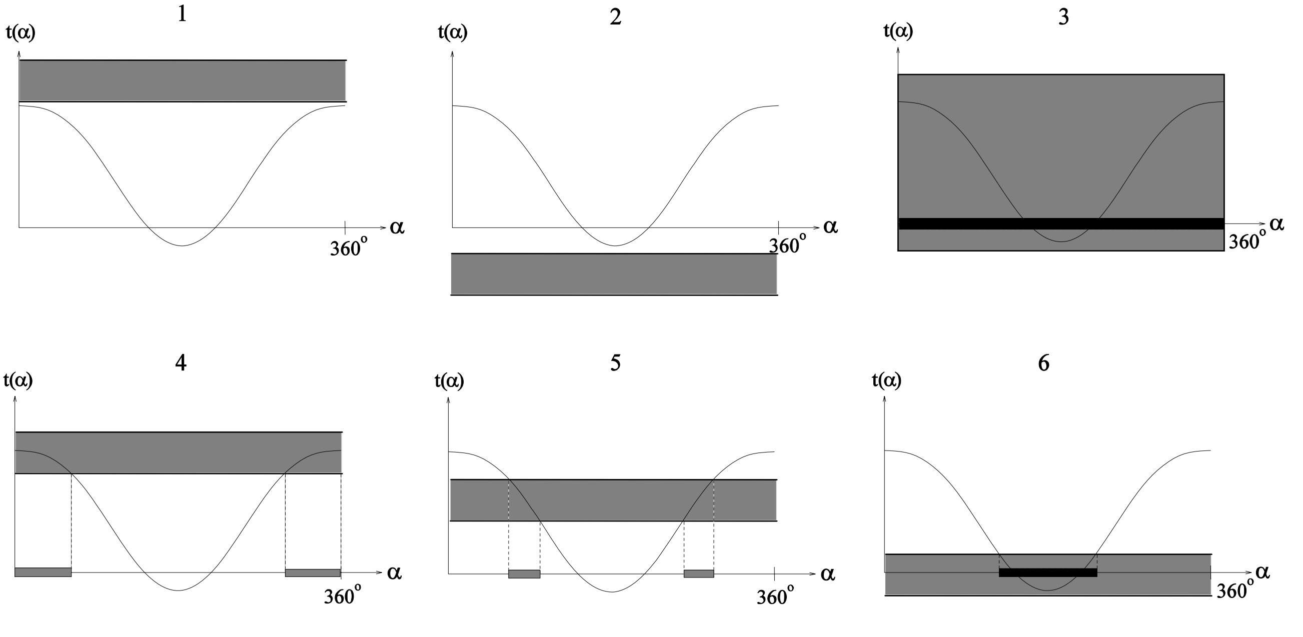


Figure 4. The six possible situations when combining the two rotation matrices

To calculate how θ_1 prunes α and how θ_3 prunes α entry (1,1) and (2,2) are applied, respectively. Since two joint angles are present in these equations, the joined legal intervals for each equation are defined over two variables rather than one, hence a legal region. This can result in more legal intervals for α due to more complex monotonic characteristics, but otherwise the calculations are similar to those relating θ_2 and α For details see (Moeslund, 2003).

3.4 Pruning α Using a Collision Constraint

The fact that two body parts cannot occupy the same space at the same time is the final constraint that prunes α. This constraint is applied for each non-pruned H_Z-value and it is therefore important to have an analytic solution that evaluates all α-values at a time instead of one at the time. To obtain this two simplifications are introduced. First, a collision is defined as a configuration where the elbow is colliding with the torso or head. Second, the torso and head are modelled as one square having sides equal to those used to define the torso in section 3.1, except $-\infty < y < \infty$. The impact of these simplifications is in practice minor and therefore justifiable.

Before applying the model two heuristic rules are introduced. If the current pose contains a collision then

- the distance in the *z*-direction between the hand and torso needs to be less than the length of the lower arm: $H_Z - b < A_l$, or
- the distance in the *x*-direction between the hand and torso needs to be less than the length of the lower arm: $H_x < A_l$

In other words, one of the two rules has to be fulfilled in order to evaluate the collision constraint. If not, the entire α-circle related to this H_Z-value is pruned.

For the remaining H_Z-values the following is done. For each value of H_Z the elbow describes a circle in space and, in general, an ellipse in the x-z-variables. The pruned interval of α is found as $[\alpha_1, \alpha_2]$ where α_1 and α_2 are the intersection points (if any exist) between the ellipse and the rectangle (ignoring the y-component).

4. Pose Estimation

So far we have focused on deriving a compact representation of the arm and shoulder and pruning this representation with respect to impossible configurations. The result is a highly reduced solution-space for each frame. Now we want to apply the result to an actual computer vision system those task is to capture the pose of the arm. That is, given the highly reduced solution-space, how do we find *the* correct pose of the arm? A brute force solution can some times be applied depending on the resolution in the solution-space and the real-time requirements. But in general and in the case of estimating the pose of the entire human the overall solution-space will still be too large even though the solution-space for the arm is limited, i.e., a non-brute force approach is required. Furthermore, the representation derived above is based on the fact that the human hand can always be found in the correct position in the image. Clearly this will not always be the case and one should therefore expect a certain amount of uncertainty and that the hand sometimes will disappear altogether. A solution to these problems is to imbed the tracking algorithm into the probabilistic framework known as Sequential Monte Carlo (SMC). The SMC framework is an efficient solution to the brute force problem and inherently handles uncertainty. We phrase our pose estimating problem as a matter of including auxiliary information (the position of the hand in the image) into the SMC framework. In the rest of this section we show how the SMC operates and how it can be improved by adding auxiliary information. Concretely, we first present the SMC framework. We then show how to include the auxiliary information. Finally we present a method for comparing the arm model with the image data.

4.1 The SMC Framework

The SMC algorithm is defined in terms of Bayes' rule and by using the first order Markov assumption. That is, the posterior probability density function (PDF) is equal to the observation PDF multiplied by the prior PDF, where the prior PDF is the predicted posterior PDF from time *t-1*:

$$p(\vec{x}_t \mid \vec{y}_t) \propto p(\vec{y}_t \mid \vec{x}_t)\, p(\vec{x}_t \mid \vec{y}_{t-1}) \tag{6}$$

where $\vec{x}_t$ is the state and $\vec{y}_t$ contains the image measurements. The predicted posterior PDF is defined as

$$p(\vec{x}_t \mid \vec{y}_{t-1}) = \int p(\vec{x}_t \mid \vec{x}_{t-1})\, p(\vec{x}_{t-1} \mid \vec{y}_{t-1}) d\vec{x}_{t-1} \tag{7}$$

where $p(\vec{x}_t \mid \vec{x}_{t-1})$ is the motion model governing the dynamics of the tracking process, i.e., the prediction, and $p(\vec{x}_{t-1} \mid \vec{y}_{t-1})$ is the posterior PDF from the previous frame. The SMC algorithm estimates $p(\vec{x}_t \mid \vec{y}_t)$ by selecting a number, *N*, of (hopefully) representative states (particles) from $p(\vec{x}_{t-1} \mid \vec{y}_{t-1})$, predicting these using $p(\vec{x}_t \mid \vec{x}_{t-1})$, and finally giving each particle a weight in accordance with the observation PDF. In this work the state vector, $\vec{x}_t$, represents the 3D model of the arm, i.e., (α, H_z).

The observation PDF, $p(\vec{y}_t \mid \vec{x}_t)$, expresses how alike each state and the image measurements are. In this work the image measurements are the probabilities of the orientations of the upper and lower arm in the image, respectively, i.e., $\vec{y}_t = [p_u(y_u), p_l(y_l)]^T$, where $p_u(y_u)$ and $p_l(y_l)$ are the PDFs of the different orientations of the upper and lower arm in the image, respectively. We define the observation PDF as

$$p(\vec{y}_t \mid \vec{x}_t) = \frac{p_u(y_u(\vec{x}_t)) + p_l(y_l(\vec{x}_t))}{2} \tag{8}$$

where $y_u(\vec{x}_t)$ and $y_l(\vec{x}_t)$ map from (α, H_Z) to the orientation of the upper and lower arm in the image, respectively[3].

4.2 Including the Auxiliary Information

In this section we describe how including auxiliary information enhances the SMC algorithm. The auxiliary information is in the form of the position of the hand in the image. Firstly we will describe how the auxiliary information is obtained and related to the SMC algorithm. Secondly we will describe how to apply the auxiliary information to correct the states of the predicted particles.

4.2.1 Obtaining the Auxiliary Information

The hand candidates in an image are detected based on skin colour segmentation. We first convert each image pixel, (R,G,B), into chromaticity, (r,g,b), and make a binary image

[3] These mappings require the camera parameters as well. But to enhance the concept we have left them out of the expression.

based on a Mahalanobis classifier, those mean and covariance are found during training. In the binary image we apply morphology followed by a connected component analysis. This gives us a number, *m*, of skin blobs, b_i, which each could represent the hand. Each skin blob is used to correct a number of particles according to the likelihood of this particular blob being a hand, $p(hand \mid b_i)$. That is, the number of particles, *N*, available at each time instance are divided between the *m* skin blobs so that blob b_i is associated with p_i particles, where

$$p_i = N \frac{p(\text{hand} \mid b_i)}{\sum_{i=1}^{m} p(\text{hand} \mid b_i)} \tag{9}$$

The problem with this approach is that it assumes that the correct hand position always is among the detected skin blobs. When this is not the case the entire system is likely to fail. To overcome this, we adapt the approach taken in (Davis *et al.*, 2000), where only a portion of the *N* particles are predicted and the remaining particles are drawn from a uniform distribution. Similar, we will always let $T \cdot N$ particles be predicted and weighted regardless of the auxiliary information, i.e., *N* is replaced by $N - T \cdot N$. Concretely we say that at least 10% of the particles should be drawn without regard to the auxiliary information and define *T* as

$$T = \begin{cases} 0.1, & \text{if } k > 0.9 \\ 1-k, & \text{else} \end{cases} \tag{10}$$

where *k* is the likelihood of the skin blob most likely to represent the hand, i.e., $k = \arg\max_i \{ p(\text{hand} \mid b_i) \}$

4.2.1.1 Defining the Likelihood of a Hand

We define the likelihood of the hand as

$$p(\text{hand} \mid b_i) = F\left(\prod_{j=1}^{t} w_j \cdot p(\text{hand} \mid f_j, b_i) \right) \tag{11}$$

where $F(\cdot)$ scales the likelihood, *t* is the number of features, w_j is the weight of the *j*th feature, f_j is the *j*th feature, and $p(hand \mid f_j, b_i)$ is the likelihood of the hand given the *j*th feature and the *i*th skin blob. The scaling of the likelihood is necessary as we use this value not only as a relative measure, but also as an absolute measure when defining *T*. In this work *F(x) = 1 - exp(-5x)* was found to work well. We use three equally weighted features, i.e., *t=3* and w_j*=1*. The first feature is based on the number of pixels in the blob. As this feature is dependent on a number of different aspects, such as the distance to the camera, we apply this feature in a very conservative manner

$$p(\text{hand} \mid f_1, b_i) = \begin{cases} 0, & \text{if } A > TH_{max} \\ 0, & \text{if } A < TH_{min} \\ 1, & \text{else} \end{cases} \tag{12}$$

where TH_{min} and TH_{max} define the lower and upper limits, respectively, and A is the area, i.e., the number of pixels.

The second feature is based on the idea that the centre of gravity (CoG) and the centre of the hand should be close to each other. This is evaluated by estimating the centre of the blob (hand) by a distance transform and comparing it with the CoG in the following way

$$p(\text{hand} \mid f_2, b_i) = 1 - \left(\frac{DT_{max} - d(CoG)}{DT_{max}} \right) \tag{13}$$

where $d(CoG)$ is the value found by the distance transform in the position of the CoG and DT_{max} is the maximum value found by the distance transform inside the blob. The last feature is inspired by the fact that an ellipse often can model the shape of a hand. We therefore calculate the semi-axes of the ellipse that corresponds to the area and perimeter of the blob. This ellipse is denoted E_d and compared to the blob to see how well it matches. An ellipse can be described by its area $A = ab\pi$ and perimeter $P = 2\pi\sqrt{\frac{1}{2}\left(a^2 + b^2\right)}$, where $2a$ is the major axis and $2b$ is the minor axis. Expressing the major and minor axes in terms of A and P yields

$$\begin{aligned} 2a &= \sqrt{\frac{P^2}{2\pi^2} + \frac{2A}{\pi}} + \sqrt{\frac{P^2}{2\pi^2} - \frac{2A}{\pi}} \\ 2b &= \sqrt{\frac{P^2}{2\pi^2} + \frac{2A}{\pi}} - \sqrt{\frac{P^2}{2\pi^2} - \frac{2A}{\pi}} \end{aligned} \tag{14}$$

The measured area and perimeter of the blob are used to calculate the axes of E_d, a and b. The centre of E_d is then placed in the CoG and E_d is rotated and compared with the blob. The rotation is done in a coarser-to-finer manner and the comparison is carried out by calculating the intersection divided by the union of the two regions, that is

$$p(\text{hand} \mid f_3, b_i) = \arg\max_{\delta} \left\{ \frac{E_d(\delta) \cap A}{E_d(\delta) + A - E_d(\delta) \cap A} \right\} \tag{15}$$

where δ is the rotation of the ellipse E_d, and A is the area of the blob.

4.2.2 Applying the Auxiliary Information

In this subsection we describe how one particle is corrected based on the auxiliary information. We first convert (α, H_Z) into the 3D position of the elbow, $\vec{E}$, and hand, $\vec{H}$, respectively. We do this conversion for two reasons. Firstly, more smooth trajectories can be expected for these parameters and hence, better motion models can be defined. Secondly, we can directly apply the CoG of the hand to correct the predictions which is not so easy in the (α, H_Z) representation. After this conversion both $\vec{E}$ and $\vec{H}$ are predicted using a linear first order motion model and then kinematic constraints described earlier are applied to ensure a possible configuration.

First we will show how the prediction of the hand, $\vec{H}$, is corrected and hereafter we will show how the predicted position of the elbow, $\vec{E}$, is corrected. In figure 5 the predictions are illustrated using subscript 'p' while the corrected predictions are illustrated using subscript 'c'. As mentioned earlier we assume a calibrated camera and can thou span a line in 3D, l , via the CoG and the camera, see figure 5 and equation 1. We can therefore correct the prediction by projecting the predicted position of the hand, $\vec{H}_P$, to the line, l . The projected prediction is denoted $\vec{H}_1$ and calculated $\vec{H}_1 = \vec{P} + \left((\vec{H}_P - \vec{P}) \cdot \vec{D}\right)\vec{D}$ where $\vec{P}$ and $\vec{D}$ are the line parameters of l, see equation 1.

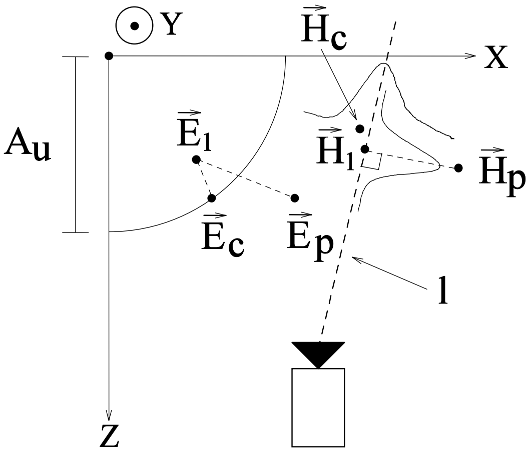


Figure 5. The shoulder coordinate system seen from above. The circle arc illustrates the sphere that defines the possible positions of the elbow. The large dashed line illustrates a camera ray through the hand. See the text for a definition of the parameters

We randomly diffuse $\vec{H}_1$ to account for the uncertainties in the estimate of the CoG. Concretely we first randomly draw a value from a Gaussian distribution with mean in $\vec{H}_1$ and standard deviation as $c \cdot \left\| \vec{H}_P - \vec{H}_1 \right\|$, where c is a predefined constant. This point defines the displacement along the camera ray with respect to $\vec{H}_1$. The resulting point is now rotated around the vector $\overrightarrow{H_1 H_P}$ using Rodriques' formula (Craig, 1989). The number of rotated degrees is determined by randomly sampling from a uniform distribution. The final diffusion of the point is along the vector $\overrightarrow{H_1 H_P}$. The displacement is define with respect to $\vec{H}_1$ and designed so that the maximum probability is at $\vec{H}_1$ and that the tail towards $\vec{H}_P$ is more likely than on the opposite side. This corresponds to a Poisson distribution with its maximum probability located in $\vec{H}_1$. We implement this by combining two Gaussian distributions, G_1 and G_2, each with mean in $\vec{H}_1$. G_1 represents the distribution on the opposite side of $\vec{H}_P$ and its variance is controlled by $p(hand \mid b_i)$. G_2 represents the distribution on the same side as $\vec{H}_P$ and its variance is controlled by both $\left\| \vec{H}_P - \vec{H}_1 \right\|$ and $p(hand \mid b_i)$. In praxis we first choose from which side of the mean we should draw a sample and then draws it from the appropriate (one-sided) Gaussian distribution. After these three diffusions we have the final corrected particle, denoted $\vec{H}_c$. The difference between the predicted and corrected particles yields a measure of the prediction error: $\vec{H}_e = \vec{H}_c - \vec{H}_P$.

The predicted position of the elbow cannot directly be corrected by the auxiliary information. However, we know the elbow is likely to have a prediction error closely related to that of the hand, as the hand and elbow are part of the same open-looped kinematic chain. We therefore calculate the corrected position, $\vec{E}_c$, by first adding the prediction error of the hand to the predicted value of the elbow, yielding $\vec{E}_1 = \vec{E}_P + \vec{H}_e$, and then finding the point closest to $\vec{E}_1$ that results in a legal configuration of the arm. In mathematical terms $\vec{E}_c = \arg\min_{\vec{E}} \left\| \vec{E} - \vec{E}_1 \right\|$ subjected to the constraints $\left\| \vec{E} \right\| = A_U$ and $\left\| \overrightarrow{EH_c} \right\| = A_L$. The solution to this problem can be found in (Moeslund, 2003).

4.3 The Observation PDF

For each blob, b_i, we estimate an observation PDF and use this to weight the particles related to a particular blob. For the $T \cdot N$ particles that are not related to a blob we use the less accurate approach of chamfer matching instead of equation 8. The distance transform is calculated on the edge image in figure 6.

The observation PDF in equation 8 is based on the orientations of the arm segments in the image. We estimate the PDFs of the orientations of the upper arm, $p_u(y_u)$, and lower arm, $p_l(y_l)$, respectively, based on edge pixels. As our input images contain background clutter and non-trivial clothes we utilize temporal edge pixels[4]. That is, we find the edge pixels in the current image using a standard edge detector and AND this result with the difference image achieved by subtracting the current- and the previous image, see figure 6 for an example[5]. Those pixels actually belonging to the arm will be located in four classes, two for the upper arm and two for the lower arm, respectively.

Our system does not impose restrictions on the clothes of the user. The clothes will in general follow gravity, hence the two classes of pixels originating from the upper sides (with respect to gravity) of the upper- and lower arm will model the structure of the arm better, see figure 6. We therefore only consider temporal edge pixels located on the "upper" sides. Concretely we define "upper" and "lower" via two lines described by the position of the shoulder in the image, the CoG of the blob, and the predicted position of the most plausible elbow location found among the most likely in the previous frame.

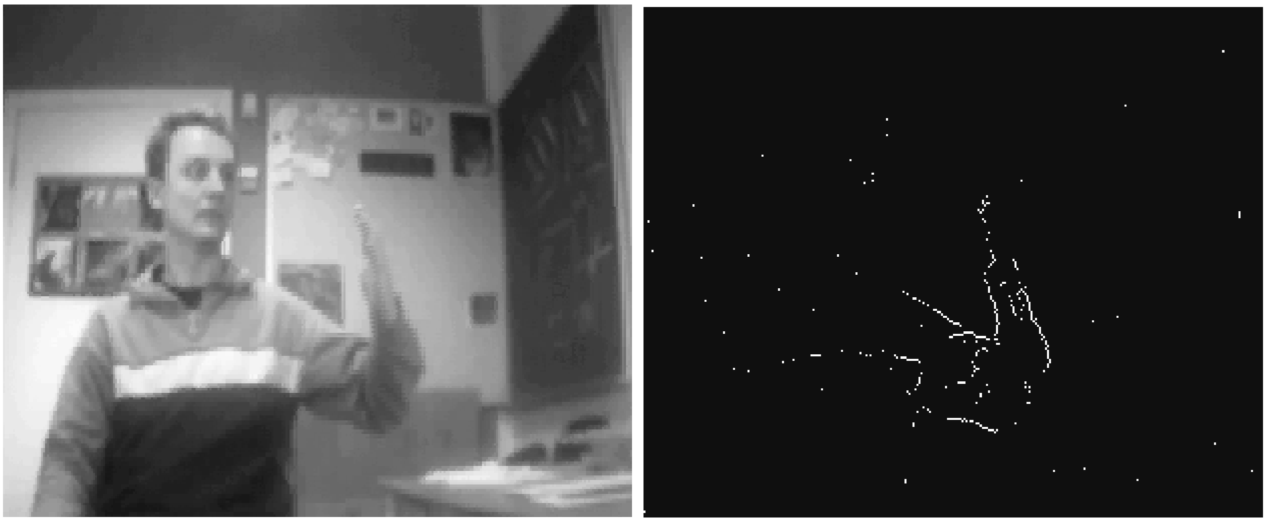


Figure 6. Left: A typical input image Right: The temporal edge pixels

[4] We assume that only ego-motion is present.

[5] When only a few temporal edges can be found we conclude that on motion is present and do not update the state-space parameters, i.e., no processing beyond this point is carried out.

We wish to estimate $p_u(y_u)$ and $p_l(y_l)$ independently and therefore separate the temporal edge pixels into two groups by calculating the perpendicular distance from each pixel to the two lines. As the prediction of the position of the elbow is uncertain we ignore all pixels within a certain distance from the predicted position of the elbow. Furthermore, we ignore all pixels too far away from both lines. When no predictions are available we use chamfer matching (as described above) as the observation PDF.

Estimating the orientation of a straight line from data can be carried out in different ways, e.g., via principal component analysis or linear regression. However, as we will not model the distribution of the orientations via Gaussians we cannot apply these methods. Instead we apply the dynamic Hough transform (DHT). It estimates the likelihood of each possible orientation, hence allowing multiple peaks in the observation PDF. The choice of the DHT is furthermore motivated by the fact that it adapts to the data. The DHT randomly samples two pixels from one group and calculates the orientation of the line spanned by the two pixels. The more times the groups are sampled the better the estimation of the PDFs will be. On the other hand many samplings also lead to large processing time. The sampling of one group is therefore terminated as soon as the variance of the PDF is stable. To evaluate the stability of the variance after *n* samplings the variance of the last *j* variances is calculated as

$$v_{jn}^2 = \frac{1}{j}\sum_{i=n-j}^{n}\left(\sigma_i^2 - \mu_{jn}\right)^2 \tag{16}$$

where σ_i^2 is the variance after *i* samplings and μ_{jn} is the mean of the last *j* variances.

The stop criterion is defined as the number of samplings, *n*, where the last *j* samplings are within the interval $\lfloor \mu_{jn} - \lambda, \mu_{jn} + \lambda \rfloor$. The distribution of the last *j* variances will in general follow a Uniform distribution. The theoretical variance of such a distribution in the given interval can be estimated as $\lambda^2/12$ (Ross, 1987). When the mean of the variances, μ_{jn} is large it indicates large uncertainty in the PDF, which again indicates weak lines in the temporal edge image. A stable variance for such a PDF tends to require a larger value of λ compared to an image with stronger lines. To account for this difference λ is defined with respect to μ_{jn} as

$$\lambda = \frac{\mu_{jn}}{\gamma} \tag{17}$$

where γ is found empirically. Setting the estimated variance equal to the theoretical variance yields $\lambda = v_{jn}\sqrt{12}$. Inserting this result into equation 17 and writing it as an inequality yields

$$v_{jn}^2 \le \frac{\mu_{jn}^2}{12 \cdot \gamma^2} \tag{18}$$

Altogether the stop criterion is found as the smallest *n* for which inequality 18 is true. To speed up the calculations the variance is not recalculated after each new sampling, but rather for every *10th* sampling. Using the above-described procedure we obtain two independent PDFs, one for the upper arm, $p_u(y_u)$ and one for the lower arm, $p_l(y_l)$. Different number of samplings might have been used to estimate the two PDFs. The accumulated probability mass for each PDF is therefore normalized to 1.

5. Results and Discussion

In this section we will evaluate our approach. That is, we will present results and discuss: the local screw axis model, the effect of the constraints, and the improved SMC framework.

5.1 The Arm Model

Modelling the pose of the arm, i.e., the GH-joint and elbow joints, by α and H_Z is a novel approach. Through geometric reasoning it can easily be shown that there exists a one-to-one mapping between our representation and the standard representation via four Euler angles. Since the Euler angles representation is sound, the same must be true for the local screw axis model.

Comparing the size of its solution-space with that of the standard approach namely the four Euler angles carries out a quantitative evaluation of the local screw axis model. In table 2 the sizes of the two representations are listed for different resolutions. The calculations are done for standard arm lengths, A_u = A_l = 30*cm*, and no constraints whatsoever are used, thus yielding the full size solution-space. The Greek letters τ and ρ represent the resolution of the Cartesian coordinates and angle values, measured in cm^{-1} and degrees^{-1}, respectively.

Model name	Parameters	Size of solution-space	$\tau = \rho = 10$	$\tau = \rho = 1$	$\tau = \rho = 0.1$
Euler angles	$\theta_1, \theta_2, \theta_3, \theta_4$	$(\tau \cdot 360)^4$	$1.68 \cdot 10^{14}$	$1.68 \cdot 10^{10}$	$1.68 \cdot 10^{6}$
Local screw axis model	α, H_Z	$\tau \cdot 360 \cdot 2(\rho \mathrm{A}_U + \rho \mathrm{A}_l)$	$4.32 \cdot 10^{6}$	$4.32 \cdot 10^{4}$	$4.32 \cdot 10^{2}$

Table 2. Comparing the local screw axis model with the four Euler angles

From the table it can be seen that a huge reduction in the total number of different arm configurations is achieved. In fact, the reduction factors for the three resolutions are: $3.89 \cdot 10^7$, $3.89 \cdot 10^5$, and $3.89 \cdot 10^3$, respectively.

5.2 Effects of the Constraints

How much a particular constraint prunes the solution-space depends on the current position of the hand and the previous estimated position of the arm, in other words spatial and temporal information[6]. It is therefore not possible to state a general pruning effect but in table 3 the intervals of the pruning effects are shown together with the average effects (Moeslund, 2003).

The first four constraints usually overlap, except for the second and third which are mutually exclusive. They, however, each overlap with the two others. The last two constraints might also overlap each other and it is therefore not possible to calculate a general accumulated effect of the different constraints. Instead the minimum, maximum,

[6] The effect of the three first and the last constraints also depends on the position of the camera, but for simplicity it is assumed that the camera is perpendicular to the torso and infinitively far away.

and average effect can be estimated. For α□ these are 75%, 100%, and 85%, respectively, and for H_Z these are 49%, 100%, and 80%, respectively. Altogether this yields a minimum pruning effect of 87%, a maximum pruning effect of 100%, and an average pruning effect of 97% (Moeslund, 2003).

Parameter	Type of constraint	Minimum	Maximum	Average
H_Z	Distance	0%	100%	48%
H_Z	Angle	0%	50%	25%
H_Z	Occlusion	0%	57%	10%
H_Z	Temporal	51%	92%	77%
□	Joint angle	75%	100%	88%
□	Collision	0%	100%	30%

Table 3. The different constraints and their pruning effects.

A resolution of for example *2cm* for H_Z and 5^o for α results in 4320 distinct configurations. Pruning yields in worst-case 553 non-pruned values and in average 125.

5.3 The Improved SMC Framework

The reason for applying the SMC framework is, as described earlier, that the SMC framework can handle both the uncertainties regarding the segmentation of the hand and at the same time avoid the need for an exhaustive search.

Distributing the particles in accordance with the likelihood of the different skin-colour blobs being a hand is in theory a solid approach. It also turns out to be an applicable solution when implementing a SMC-based tracker. In 7.A an example image from a test sequence is shown where 50 particles are used to track the arm. The circles represent the corrected position of the hand projected into the image and it is evident that the main parts of the particles are located on and around the true hand of the hand[7].

As shown above the SMC framework can improve our modelling approach, but in fact our modelling approach can also improve the SMC framework. To illustrate this point we implemented a standard SMC tracker based on the same observation PDF but using Euler angles to represent the solution-space as opposed to the local screw axis model. For both the standard SMC-tracker and our version 50 particles are applied. After tracking the arm for 100 frames the characteristics of the two algorithms are illustrated in figure 7.

In figure 7 we show the values of the predicted particles in the standard SMC algorithm (figure 7.B) and the values of the corrected particles when our algorithm is applied (figure 7.A). We do not visualize the parameters in the solution-space but rather the 3D position of the hand projected into the image. In figure 7.A the main parts of the particles are located around the segmented skin-coloured blobs and especially around the hand. These more focused particles result in a higher probability of finding the correct pose of the arm - even when using as few as 50 particles. This can also be seen in figure 7.C and 7.D where the three particles with the biggest likelihood are illustrated for the standard SMC algorithm (7.D) and when applying our method (7.C). It can be seen that our method improves the results.

[7] Note that the face blob is eliminated by feature, f_1. The neck region is segmented into a different blob and therefore associated with some particles.

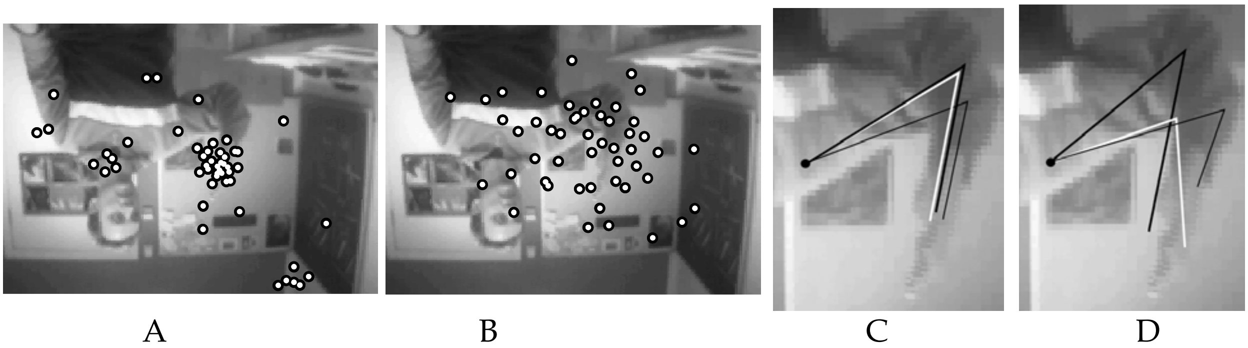


Figure 7. Next we test whether the true state of the arm is among the states found by the SMC algorithm. If this is the case the SMC algorithm can indeed avoid the need for an exhaustive search. In figure 7.C we show the 2D projection of the three particles with the biggest likelihood (highest weight first: black, white, thin black)

In images such as the one in figure 7.A the posterior PDF is in general ambiguous and a "ridge" will be present in the posterior PDF. This means that a number of correct poses, i.e., poses that can explain the current image data, can be found by increasing the distance between the hand and camera, i.e., moving along the ridge. This tendency can be seen in figure 7.D while the standard SMC algorithm fails to capture this tendency.

6. Conclusion

In this article we have shown how to pose estimate the human arm using monocular computer vision. The hypothesis behind our approach is that the fewer possible solutions and the better the search through these, the higher the likelihood of finding the correct pose in a particular frame. To achieve fewer solutions we did two things. Firstly, we introduced a very compact representation of the human arm based on the position of the hand measured in the current image. We denoted this representation the *local screw axis model*. Secondly, we applied the kinematics of the human arm in order to prune the solution space. In average our constraints can prune the solution-space with 97%

Our representation of the arm is based on the position of the hand in the image. In order to account for the inherent uncertainties in such a representation we imbed our approach in the SMC framework. This framework allows us to model the uncertainties of the position of the hand and the disappearing of the hand in the image (tracking failure).

Besides the above-mentioned issues we have also made a contribution in this work by showing how to model the complex movements in the shoulder without introducing additional parameters. That is, the displacements of the shoulder during arm movement can now be modelled without increasing the dimensionality of the solution-space, i.e., a more precise solution can be achieved. Lastly it should be mentioned that the idea of correcting the predictions in the SMC framework using auxiliary information can in general improved all pose estimating systems where the object to be tracked is an open-kinematic chain.

7. References

An, K.N. & Morrey, B.F. (1985). Biomechanics of the Elbow. *The Elbow and its Disorders*. W.B. Saunders Company, 1985.

Azoz, Y.; Devi, L. & Sharme, R. (1998). Reliable Tracking of Human Arm Dynamics by Multiple Cue Integration and Constraint Fusion. *International Conference on computer Vision and Pattern*. Santa Barbara, California, June, 1998.

Ben-Arie, J.; Wang, Z., Pandit, P. & Rajaram, S. (2002). Human Activity Recognition Using Multidimensional Indexing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. Vol. 24, Nr. 7, 2002.

Bregler, C. & Malik, J. (1998). Tracking People with Twists and Exponential Maps. *International Conference on Computer Vision and Pattern Recognition*. Santa Barbara, California, June, 1998.

Breteler, K.; Spoor, C.W. & Van der Helm, F.C.T. (1999). Measuring Muscle and Joint Geometry Parameters of a Shoulder for Modelling Purposes. *Journal of Biomechanics*. Vol. 32, nr. 11, 1999.

Codman, E.A. (1934). *The Shoulder*. Boston: Thomas Todd Co. 1934.

Craig, J.J. (1989). *Introduction to Robotics. Mechanics and Control*. Addison Wesley. 1989.

Culham, E. & Peat, M. (1993). Functional Anatomy of the Shoulder Complex. *Journal of Orthopaedic and Sports Physical Therapy*. Vol. 18, nr. 1, 1993.

Davis, L.; Philomin, V. & Duraiswami, R. (2000). Tracking Humans from a Moving Platform. *International Conference on Pattern Recognition*. Barcelona, Spain, Sept., 2000.

Delamarre, Q. & Faugeras, O. (2001). 3D Articulated Models and Multi-view Tracking with Physical Forces. *Computer Vision and Image Understanding*. Vol. 81, Nr. 3, 2001.

Deutscher, J.; Blake, A. & Reid, I. (2000). Articulated Body Motion Capture by Annealed Particle Filtering. *Computer Vision and Pattern Recognition*. South Carolina, June, 2000.

Dvir, Z. & Berme, N. (1978). The Shoulder Complex in Elevation of the Arm: A Mechanism Approach. *Journal of Biomechanics*. Vol. 11, 1978.

Engin, A.E. & Tumer, S.T. (1989). Three-Dimensional Kinematic Modelling of the Human Shoulder Complex - Part 1: Physical Model and Determination of Joint Sinus Cones. *Journal of Biomechanical Engineering*. Vol. 111, 1989.

Fua, P.; Gruen, A., Plankers, R., D'Apuzzo, N. & Thalmann, D. (1998). Human Body Modeling and Motion Analysis From Video Sequences. *International Symposium on Real Time Imaging and Dynamic Analysis.* Hakodate, Japan, June, 1998.

Gavrila, D.M. & Davis, L.S. (1996). 3-D Model-Based Tracking of Humans in Action: A Multi-View Approach. *Conference on Computer Vision and Pattern Recognition*. San Francisco, USA, 1996.

Goncalves, L.; Bernardo, E.D., Ursella, E. & Perona, P. (1995). Monocular Tracking of the Human Arm in 3D. *International Conference on Computer Vision*. Cambridge, Massachusetts, 1995.

Hogfors, C.; Peterson, B., Sigholm, G. & Herberts, P. (1991). Biomechanical Model of the Human Shoulder Joint - 2. The Shoulder Rhythm. *Journal on Biomechanics.* Vol. 24, nr.

Howe, N.R; Leventon, M.E. & Freeman, W.T. (2000). Bayesian Reconstruction of 3D Human Motion from Single-Camera Video. Advances in Neural Information Processing Systems 12. MIT Press, 2000.

Inman, V.T.; Saunders, J.B. & Abbott, L.C. (1944). Observations on the Function of the Shoulder Joint. *Journal on Bone and Joint Surgery*. Vol. 26, 1944.

Isard, M. & Blake, A. (1998). CONDENSATION - conditional density propagation for visual tracking. *International Journal on Computer Vision*. Vol. 29, nr. 1, 1998.

Kondo, K. (1991). Inverse Kinematics of a Human Arm. *Journal on Robotic Systems*. Vol. 8,

Lee, M.W. & Cohen, I. (2004). Human Upper Body Pose Estimation in Static Images. *European Conference on Computer Vision*. Prague, May 2004.

Leva, P. (1996). Joint Center Longitudinal Positions Computed from a Selected Subset of Chandler's Data. *Journal on Biomechanics*. Vol. 29, 1996.

Maurel, W. (1998). *3D Modeling of the Human Upper Limb including the Biomechanics of Joints, Muscles and Soft Tissues*. Ph.D. Thesis, Laboratoire d'Infographie - Ecole Polytechnique Federale de Lausanne, 1998.

Mitchelson, J. & Hilton, A. (2003). From Visual Tracking to Animation using Hierarchical Sampling. *Conference on Model-based Imaging, Rendering, image Analysis and Graphical special Effects.*, France, March, 2003.

Morrey, B.F. (1985). Anatomy of the Ellow Joint. *The Elbow and its Disorders*. W.B. Saunders Company, 1985.

Moeslund, T.B. & Granum, E. (2001). A Survey of Computer Vision-Based Human Motion Capture. *Journal on Computer Vision and Image Understanding*, Vol. 81, nr. 3, 2001.

Moeslund, T.B. (2003). *Computer Vision-Based Motion Capture of Body Language*. Ph.D. Thesis, Laboratory of Computer Vision and Media Technology, Aalborg University, Denmark, 2003.

Ogaki, K.; Iwai, Y. & Yachida, M. (2001). Posture Estimation Based on Motion and Structure Models. *Systems and Computers in Japan*, Vol 32, Nr. 4, 2001.

Ong, E.J. & Gong, S. (1999). Tracking Hybrid 2D-3D Human Models from Multiple Views. *International Workshop on Modeling People at ICCV'99*. Corfu, Greece, September, 1999.

Pavlovic, V.; Rehg, J.M., Cham, T.J. & Murphy, K.P. (1999). A Dynamic Bayesian Network Approach to Figure Tracking Using Learned Dynamic Models. *International Conference on Computer Vision*. Corfu, Greece, September, 1999.

Plankers, R.; Fua, P. & D'Apuzzo, N. (1999). Automated Body Modeling from Video Sequences. *International Workshop on Modeling People at ICCV'99*. Corfu, Greece, September, 1999.

Rohr, K. (1997). *Human Movement Analysis Based on Explicit Motion Models*. Kluwer Academic Publishers, Dordrecht Boston, 1997.

Ross, S.M. (1987). *Introduction to Probability and Statistics for Engineers and Scientists*. Wiley Series in Probability and Mathematical Statistics. 1987.

Segawa, H.; Shioya, H., Hiraki, N. & Totsuka, T. (2000). Constraint-Conscious Smoothing Framework for the Recovery of 3D Articulated Motion from Image Sequences. *The fourth International Conference on Automatic Face and Gesture Recognition*. Grenoble, France, March, 2000.

Sidenbladh, H.; De la Torre, F. & Black, M.J. (2000). A Framework for Modeling the Appearance of 3D Articulated Figures. *The fourth International Conference on Automatic Face and Gesture Recognition*. Grenoble, France, March, 2000.

Sminchisescu, C. (2002). Consistency and Coupling in Human Model Likelihoods. *International Conference on Automatic Face and Gesture Recognition.* Washington D.C.

Soslowsky, L.J.; Flatow, E.L., Bigliani, L.U. & Mow, V.C. (1992). Articular Geometry of the Glenohumeral Joint. *Journal on Clinical Orthopaedics and Related Research*. Vol. 285, 1992.

Wachter, S. & Nagel, H.-H. (1999). Tracking Persons in Monocular Image Sequences. *Journal on Computer Vision and Image Understanding*. Vol. 74, nr. 3, 1999.

Wren, C.R.; Clarkson, B.P. & Pentland, A.P. (2000). Understanding Purposeful Human Motion. *The fourth International Conference on Automatic Face and Gesture Recognition*. Grenoble, France, March, 2000.

Wu, Y.; Hua, G. & Yu, T. (2003). Tracking Articulated Body by Dynamic Markov Network. *International Conference on Computer Vision*, Nice, France. 2003.

Zatsiorsky, V.M. (1998). *Kinematics of Human Motion*. Champaign, IL: Human Kinetics, 1998.

Cartesian Impedance Control of Flexible Joint Robots: A Decoupling Approach

Christian Ott, Alin Albu-Schaffer, Andreas Kugi & Gerd Hirzinger

1. Introduction

Whenever a robotic manipulator is supposed to get in contact with its environment, the achievement of a compliant behavior is relevant. This is a classical control problem for rigid body robots, which led to control approaches like impedance control, admittance control, or stiffness control (Hogan, 1985; Sciavicco & Siciliano,1996). In contrast to the approach introduced in this contribution most works on the Cartesian impedance control problem consider a robot model which does not include joint flexibility.

In this work a decoupling based control approach for flexible joint robots is described. The considered control objective is the achievement of a desired compliant behavior between external generalized forces and the Cartesian end-effector motion of the robot. The design method will be based on some results from control theory for cascaded systems. The proposed controller will be designed in two steps. First, an inner feedback loop is used to bring the flexible joint robot model into cascaded form. Then, an additional outer control loop implements the desired compliant behaviour (Ott et al., 2003). The stability theory for cascaded control systems (Seibert & Suarez, 1990; Loria, 2001) can be applied to analyze the closed-loop system.

When dealing with a robot model with flexible joints, the maybe most obvious approach for the design of an impedance controller is the singular perturbation approach (Spong, 1987; De Luca, 1996; Ott et al., 2002). Therein the fast subsystem, which is in our case the torque dynamics, is considered as a perturbation of the rigid body model. One can then use any controller designed for the rigid body robot dynamics and apply it in combination with an inner torque control loop to the flexible joint robot model. The main disadvantage of this approach is that it does not allow for a formal stability proof without referring to the approximate singular perturbation consideration.

The controller structure proposed herein is somewhat related to the singular perturbation based controller. Also herein an inner torque control loop is combined with an outer impedance control loop. But these control loops will be designed in such a way that one can give a proof of asymptotic stability, based on the stability theory for cascaded systems. The proposed controller structure is also related to the controllers presented in (Lin & Goldenberg, 1995; 1996). But the following analysis focuses on the design of an impedance controller, while Lin and Goldenberg considered a position controller respectively a hybrid position/force-controller. The procedure for the stability analysis from these works cannot be applied to the impedance control problem considered herein in a straightforward way. The chapter is organized as follows. In Section 2 some relevant results of the stability theory for cascaded systems are reviewed. The considered dynamical model of a flexible joint robot is described in Section 3. In Section 4 the design idea based on an inner torque control loop is presented. Section 5 presents the Cartesian impedance controller. Some simulation results are given in Section 6. Finally, Section 7 gives a short summary of the presented work.

2. Stability Theory for Cascaded Control Systems

Consider an autonomous cascaded system in the form

$$\dot{x}_1 = f_1(x_1, x_2), \tag{1}$$

$$\dot{x}_2 = f_2(x_2), \tag{2}$$

where $x_1 \in \Re^{n_1}$ and $x_2 \in \Re^{n_2}$ are the state variables. It is assumed that the functions $f_1(x_1, x_2)$ and $f_2(x_2)$ are locally Lipschitz and that all solutions exist for all times $t > 0$. Furthermore, it is assumed that the origin is an equilibrium point of (1)-(2), i.e. $f_1(0,0) = 0$ and $f_2(0) = 0$. In the following the situation is analyzed when the uncoupled system (2) and

$$\dot{x}_1 = f_1(x_1, 0), \tag{3}$$

are globally asymptotically stable. Then the question arises, under which conditions also the coupled system (1)-(2) will be asymptotically stable. Locally this is always true, see the references in (Seibert & Suarez, 1990). In order to ensure that this holds also globally it was proven in (Seibert & Suarez, 1990) that it is sufficient to show that all solutions of the coupled system remain bounded. This is formulated in a more general form in the following theorem, taken from (Seibert & Suarez, 1990).

Theorem 1. *If the system system (3) is globally asymptotically stable, and if (2) is asymptotically stable with region of attraction* $A \subseteq \Re^{n_2}$ *, and if every orbit of (1)-(2) with initial point in* $\Re^{n_1} \times A$ *is bounded for* $t > 0$*, then the system (1)-(2) is asymptotically stable with region of attraction* $\Re^{n_1} \times A$.

Notice that Theorem 1 also handles the case of global asymptotic stability, if the region of attraction of (2) is the whole state space $\Re^{n_2}$. Theorem 1 considers only the autonomous case. In case of the tracking control problem, on the other hand, a time-varying system must be considered. An extension of Theorem 1 to a special class of time-varying systems was presented in (Loria, 2001).

Theorem 2. *Consider the system*

$$\dot{x}_1 = f_1(x_1, t) + h(x_1, x_2, t)x_2, \tag{4}$$

$$\dot{x}_2 = f_2(x_2, t), \tag{5}$$

with state $(x_1, x_2) \in \Re^{n_1} \times \Re^{n_2}$. *The functions* $f_1(x_1, t)$, $f_2(x_2, t)$, *and* $h(x_1, x_2, t)$ *are continuous in their arguments, locally Lipschitz in* (x_1, x_2), *uniformly in* t, *and* $f_1(x_1, t)$ *is continuously differentiable in both arguments. This system is uniformly globally asymptotically stable if and only if the following holds:*

- *There exists a non-decreasing function* $H(\cdot)$ *such that*

$$\| h(x_1, x_2, t) \| \leq H(\| (x_1, x_2) \|) \ . \tag{6}$$

- *The systems (5) and* $\dot{x}_1 = f_1(x_1, t)$ *are uniformly globally asymptotically stable.*
- *The solutions of (4)-(5) are uniformly globally bounded.*

Notice that, if the uncoupled systems $\dot{x}_1 = f_1(x_1, t)$ and $\dot{x}_2 = f_2(x_2, t)$ can be shown to be exponentially stable, then there exist also other results in the literature which can be applied to such a triangular system, see, e.g., (Vidyasagar, 1993). The impedance control problem from Section 5 though leads to a closed-loop system for which only (uniform global) asymptotic stability, instead of exponential stability, can be shown for the respective subsystem $\dot{x}_1 = f_1(x_1, t)$. In the following it will be shown how Theorem 2 can be used for the design of a Cartesian impedance controller for the flexible joint robot model.

3. Considered Robot Model

In this work the so-called *reduced model* of a robot with n flexible joints is considered as proposed by (Spong, 1987):

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = \tau + \tau_{ext}, \tag{7}$$

$$B\ddot{\theta} + \tau = \tau_m, \tag{8}$$

$$\tau = K(\theta - q). \tag{9}$$

Herein, $q \in \Re^n$ is the vector of link positions and $\theta \in \Re^n$ the vector of motor positions. The vector of transmission torques is denoted by τ. The link side dynamics (7) contains the symmetric and positive definite inertia matrix $M(q)$, the vector of Coriolis and centripetal torques $C(q,\dot{q})\dot{q}$ and the vector of gravitational torques $g(q)$. Furthermore, B and K are diagonal matrices containing the motor inertias and the stiffness values for the joints. The vector of motor torques τ_m will serve as the control input and τ_{ext} is the vector of external torques being exerted by the manipulator's environment.

Notice that herein friction effects have been neglected which may be justified by a sufficiently accurate friction compensation. It is further assumed that the external torques τ_{ext} can be measured. If these torques are generated solely by external forces and torques at the end-effector, this can be realized by the use of a 6DOF force/torque-sensor mounted on the tip of the robot.

For the further analysis, the model (7)-(8) may be rewritten by choosing the state variables $(q, \dot{q}, \tau, \dot{\tau})$ in the form

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = \tau + \tau_{ext}, \tag{10}$$

$$BK^{-1}\ddot{\tau} + \tau = \tau_m - BM(q)^{-1}\left(\tau + \tau_{ext} - C(q,\dot{q})\dot{q} - g(q)\right). \tag{11}$$

Based on this model, the following controller design procedure is motivated by considering Theorem 2. An inner torque control loop is used to decouple the torque dynamics (11) exactly from the link dynamics (10). The (time-varying) set-point τ_d for this inner control loop is generated by an impedance controller in the outer loop, cf. Section 5. In the next section the design of the inner loop torque controller is treated.

4. Controller Design Idea

In the following a torque controller is designed in such a way that the inner closed-loop system becomes uniformly globally asymptotically stable and is decoupled from the link dynamics. Obviously, some undesired terms in (11) can be eliminated by a feedback compensation of the form

$$\tau_m = u + BM(q)^{-1}\left(\tau + \tau_{ext} - C(q,\dot{q})\dot{q} - g(q)\right), \tag{12}$$

where u is an intermediate control input. This transforms (11) into

$$BK^{-1}\ddot{\tau} + \tau = u \ . \tag{13}$$

By introducing the torque set-point τ_d and a torque error variable $z = \tau - \tau_d$, one obtains

$$BK^{-1}(\ddot{z} + \ddot{\tau}_d) + z + \tau_d = u \ . \tag{14}$$

A controller which makes this system globally asymptotically stable can easily be found. Consider for instance the control law

$$u = \tau_d + BK^{-1}(\ddot{\tau}_d - K_s\dot{z} - K_t z) \ , \tag{15}$$

with symmetric and positive definite gain matrices K_s and K_t. This leads to a closed-loop system of the form

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = z + \tau_d + \tau_{ext}, \tag{16}$$

$$\ddot{z} + K_s\dot{z} + (K_t + KB^{-1})z = 0. \tag{17}$$

Clearly, the decoupled torque error dynamics (17) is an exponentially stable linear system. Next, the torque set-point τ_d is to be chosen such that a particular control goal is achieved. In order to get a uniformly globally asymptotically stable closed-loop system one must ensure that the conditions of Theorem 2 are fulfilled. Notice at this point also that the conditions of Theorem 2 are necessary and sufficient.

5. A Cartesian Impedance Controller

5.1 Task Description

The considered task for the controller design is an impedance controller in Cartesian coordinates $x \in \Re^m$ describing the end-effector pose. These Cartesian coordinates are given by a mapping $x = f(q)$ from the configuration space to the task space. The Jacobian of this mapping is denoted by $J(q) = \partial f(q)/\partial q$. The Cartesian velocity and acceleration are then given by

$$\dot{x} = J(q)\dot{q}, \tag{18}$$

$$\ddot{x} = J(q)\ddot{q} + \dot{J}(q)\dot{q}. \tag{19}$$

In this work the non-redundant and non-singular case is treated. It is thus considered that the number n of configuration coordinates and the number m of task coordinates are equal and that the Jacobian matrix $J(q)$ is non-singular in the considered region of the workspace. Furthermore, the mapping $f(q)$ is assumed to be invertible such that the Cartesian coordinates can be used as a set of generalized coordinates for the further analysis. Formally, the Jacobian matrix $J(q)$ is then expressed in terms of Cartesian coordinates as $\bar{J}(x) = J(f^{-1}(x)) = J(q)$.

These assumptions are quite common in designing controllers with respect to Cartesian coordinates. In practice they clearly can only be fulfilled in a limited region of the workspace which depends on the particular choice of Cartesian coordinates $x = f(q)$. However, in the following the control objective is to achieve a uniformly globally asymptotically stable closed-loop system. Thereby, the validity of the control law is analyzed for a globally valid set of coordinates while disregarding the problem of finding an appropriate set of Cartesian coordinates suitable for a given task).

According to the coordinates x, the external torques τ_{ext} can be written in terms of a generalized Cartesian force vector F_{ext} defined by the relationship $\tau_{ext} = \bar{J}(x)^T F_{ext}$.

Equation (16) can then be rewritten in terms of Cartesian coordinates as

$$\Lambda(x)\ddot{x} + \mu(x,\dot{x})\dot{x} + p(x) = \bar{J}(x)^{-T}(z + \tau_d) + F_{ext} \ , \tag{20}$$

where the Cartesian inertia matrix $\Lambda(x)$ and the matrix $\mu(x,\dot{x})$ are given by

$$\Lambda(x) = \left(J(q)^{-T} M(q) J(q)^{-1}\right)_{q=f^{-1}(x)}, \tag{21}$$

$$\mu(x,\dot{x}) = \left(J(q)^{-T} C(q,\dot{q}) J(q)^{-1} - \Lambda(q)\dot{J}(q)J(q)^{-1}\right)_{q=f^{-1}(x)}, \tag{22}$$

and $p(x) = \bar{J}(x)^{-T} g(f^{-1}(x)) = J(q)^{-T} g(q)$ is the Cartesian gravity vector.

Based on the Cartesian model (20), the desired closed-loop behavior is formulated next. The considered impedance control problem is specified by means of symmetric and positive definite stiffness and damping matrices K_d and D_d, and a (possibly time-varying) desired trajectory $x_d(t)$. The desired trajectory is assumed to be four times continuously differentiable. For many applications the shaping of the desired stiffness and damping behavior is sufficient, and no shaping of the inertia matrix is required. Therefore, the Cartesian robot inertia matrix $\Lambda(x)$ as well as the matrix $\mu(x,\dot{x})$ are preserved in the desired impedance behavior. With the Cartesian error vector $e_x = x - x_d$ the considered desired impedance can be written as

$$\Lambda(x)\ddot{e}_x + (\mu(x,\dot{x}) + D_d)\dot{e}_x + K_d e_x = F_{ext} \ . \tag{23}$$

For this system the following two important properties hold.

Property 1. *For* $F_{ext} = 0$, *the system (23) with the symmetric and positive definite stiffness and damping matrices* K_d *and* D_d *is uniformly globally asymptotically stable.*

Property 2. *For* $\dot{x}_d(t) = 0$, *the system (23) with the symmetric and positive definite stiffness and damping matrices* K_d *and* D_d *gets time-invariant and represents a passive mapping from the external force* F_{ext} *to the velocity error* $\dot{e}_x$.

Property 1 was proven in (Paden & Panja, 1988; Santibanez & Kelly, 1997) for a system like (23) but in configuration coordinates. This result obviously also holds for the considered Cartesian coordinates. Property 2 corresponds to the well known passivity property of mechanical systems and is based on the fact that the matrix $\dot{\Lambda}(x) - 2\mu(x,\dot{x})$ is skew symmetric (Sciavicco & Siciliano, 1996).

These two properties verify that the desired impedance behavior is chosen properly. Property 2 is of particular importance for an impedance controller, since it implies that the feedback interconnection with any passive environment, considered as a mapping $\dot{e}_x \rightarrow -F_{ext}$, results in a passive closed-loop system.

5.2 Controller Design

The remaining part of the controller design aims at choosing a control input τ_d such that the system

$$\Lambda(x)\ddot{x} + \mu(x,\dot{x})\dot{x} + p(x) = \bar{J}(x)^{-T}(z + \tau_d) + F_{ext}, \tag{24}$$

$$\ddot{z} + K_s\dot{z} + (K_t + KB^{-1})z = 0, \tag{25}$$

which already contains the torque feedback action, resembles the desired dynamics (23) as closely as possible. If the systems (24) and (25) were completely uncoupled (i.e. for $z = 0$ in (24)), the control objective could be exactly fulfilled by a feedback of the form

$$\begin{aligned} \tau_d &= \bar{J}(x)^T p(x) + \bar{J}(x)^T \left(\Lambda(x)\ddot{x}_d + \mu(x,\dot{x})\dot{x}_d - D_d\dot{e}_x - K_d e_x\right) \\ &= g(q) + J(q)^T \left(\Lambda(x)\ddot{x}_d + \mu(x,\dot{x})\dot{x}_d - D_d\dot{e}_x - K_d e_x\right). \end{aligned} \tag{26}$$

In the following it will be shown that the feedback law (26) leads to a uniformly globally asymptotically stable closed-loop system also for the flexible joint robot model (24)-(25). Thereby, the cascaded structure of (24)-(25) will be utilized by applying Theorem 2. Moreover, it will be shown that a passivity property analogous to Property 2 holds.

The closed-loop system containing the torque feedback action, cf. (12) and (15), and the impedance control law (26) is given by

$$\Lambda(x)\ddot{e}_x + \mu(x,\dot{x})\dot{e}_x + D_d\dot{e}_x + K_d e_x = F_{ext} + \bar{J}(x)^{-T} z, \tag{27}$$

$$\ddot{z} + K_s\dot{z} + (K_t + KB^{-1})z = 0, \tag{28}$$

Notice that this system is time-varying due to the dependence on both x and e_x.

The main results of the following analysis are formulated in form of two propositions:

Proposition 1. *For* $F_{ext} = 0$*, the system (27)-(28) with the symmetric and positive definite matrices* K_s*,* K_t*,* K_d*, and* D_d *is uniformly globally asymptotically stable.*

Proposition 2. *For* $\dot{x}_d(t) = 0$*, the system (27)-(28) with the symmetric and positive definite matrices* K_s*,* K_t*,* K_d*, and* D_d *gets time-invariant and represents a passive mapping from the external force* F_{ext} *to the velocity error* $\dot{e}_x$*.*

5.3 Proof of Proposition 1

Before the actual proof is started, two well known technical lemmata are quoted for further reference (Horn & Johnson, 1990; Vidyasagar, 1993).

Lemma 1. *Suppose that a symmetric matrix* A *is partitioned as*

$$A = \begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{1,2}^T & A_{2,2} \end{bmatrix}$$

where $A_{1,1}$ *and* $A_{2,2}$ *are square. Then the matrix* A *is positive definite if and only if* $A_{1,1}$ *is positive definite and* $A_{2,2} - A_{1,2}^T A_{1,1}^{-1} A_{1,2} > 0$ *(positive definite).*

Lemma 2. *Given an arbitrary positive definite matrix* Q*, one can find a unique positive definite solution* P *of the Lyapunov equation* $A^T P + PA = -Q$ *if and only if the matrix* A *is Hurwitz.*

For the stability analysis of the system (27)-(28) it is convenient to rewrite it in the state variables $(e_x, \dot{e}_x, z, \dot{z})$. Therefore, the following substitutions are made: $J(e_x,t) = J(f^{-1}(x)) = J(q)$, $\Lambda(e_x,t) = \Lambda(x)$, and $\mu(e_x,\dot{e}_x,t) = \mu(x,\dot{x})$. With $w = (z,\dot{z})$ and

$$A = \begin{bmatrix} 0 & I \\ -K_s & -(K_t + KB^{-1}) \end{bmatrix}$$

(28) can be written as $\dot{w} = Aw$. Thus, for $F_{ext} = 0$ we have

$$\Lambda(e_x,t)\ddot{e}_x + \mu(e_x,\dot{e}_x,t)\dot{e}_x + D_d\dot{e}_x + K_d e_x = J(e_x,t)^{-T} z, \tag{29}$$

$$\dot{w} = Aw. \tag{30}$$

In this form Theorem 2 can be applied. The first condition in Theorem 2 is the existence of a function $H(\cdot)$, for which (6) holds. This is ensured here by the assumption that the Jacobian matrix is non-singular for all times t. Hence there exists a $\delta \in \Re$, $0 < \delta < \infty$ such that

$$\| J(e_x,t)^{-T} \| \le \sup_{t\in[0,\infty)} \sqrt{\lambda_{\max}(J(e_x,t)^{-1} J(e_x,t)^{-T})} \le \delta \ ,$$

with $\lambda_{\max}(A(t))$ the maximum eigenvalue of the matrix $A(t)$ at time t.

Uniform global asymptotic stability of each of the two uncoupled subsystems is ensured by Property 1 and the fact that the linear system $\dot{w} = Aw$ is even globally exponentially stable for positive definite matrices K_s and K_t.

What remains is to show that all solutions of (29)-(30) remain uniformly globally bounded. Consider therefore the following positive definite function

$$V(e_x, \dot{e}_x, w, t) = \frac{1}{2}\dot{e}_x^T \Lambda(e_x, t)\dot{e}_x + \frac{1}{2}e_x^T K_d e_x + w^T P w \ , \tag{31}$$

with a positive definite matrix P. Considering the well known skew symmetry property of $\dot{\Lambda}(x) - 2\mu(x, \dot{x})$, one can derive the time derivative of (31) along the solutions of (29)-(30) as

$$\dot{V}(e_x, \dot{e}_x, w, t) = -\dot{e}_x^T D_d \dot{e}_x - w^T Q w + \dot{e}_x^T J(e_x, t)^{-T} z \ ,$$

where $Q = -(A^T P + PA)$ can be an arbitrary positive definite matrix, since A is Hurwitz (see Lemma 2). Then, $\dot{V}(e_x, \dot{e}_x, w, t)$ can be written as

$$\dot{V}(e_x, \dot{e}_x, w, t) = -\begin{bmatrix} \dot{e}_x \\ z \\ \dot{z} \end{bmatrix}^T N(e_x, t) \begin{bmatrix} \dot{e}_x \\ z \\ \dot{z} \end{bmatrix} ,$$

with

$$N(e_x, t) = \begin{bmatrix} D_d & \begin{bmatrix} -\frac{1}{2}J(e_x, t)^{-T} & 0 \end{bmatrix} \\ \begin{bmatrix} -\frac{1}{2}J(e_x, t)^{-1} \\ 0 \end{bmatrix} & Q \end{bmatrix} .$$

From Lemma 1 it follows that $N(e_x, t)$ is positive definite if and only if

$$Q - \frac{1}{4}J(e_x, t)^{-1} D_d^{-1} J(e_x, t)^{-T} > 0 \ . \tag{32}$$

Condition (32) can be fulfilled for every positive definite matrix D_d, because the Jacobian does not get singular and the matrix Q is a positive definite matrix which may be chosen arbitrarily. Hence, one can conclude

$$\dot{V}(e_x, \dot{e}_x, w, t) \le 0 \ . \tag{33}$$

At this point it is worth mentioning that $V(e_x, \dot{e}_x, w, t)$ is bounded from above and below by some time-invariant functions $W_1(e_x, \dot{e}_x, w)$ and $W_2(e_x, \dot{e}_x, w)$, i.e.

$$W_1(e_x, \dot{e}_x, w) \le V(e_x, \dot{e}_x, w, t) \le W_2(e_x, \dot{e}_x, w),$$

$$W_1(e_x, \dot{e}_x, w) = \frac{1}{2}\lambda_1 \| \dot{e}_x \|_2^2 + \frac{1}{2}e_x^T K_d e_x + w^T P w,$$

$$W_2(e_x, \dot{e}_x, w) = \frac{1}{2}\lambda_2 \| \dot{e}_x \|_2^2 + \frac{1}{2}e_x^T K_d e_x + w^T P w,$$

where

$$0 < \lambda_1 < \inf_{t \in [0,\infty)} \lambda_{\min}(\Lambda(e_x, t)) < \sup_{t \in [0,\infty)} \lambda_{\max}(\Lambda(e_x, t)) < \lambda_2 < \infty \ ,$$

with $\lambda_{\min}(A(t))$ and $\lambda_{\max}(A(t))$ as the minimum and maximum eigenvalue of the matrix $A(t)$ at time t.

Based on these properties of $V(e_x, \dot{e}_x, w, t)$, Lemma A.1 from the Appendix, and $\dot{V}(e_x, \dot{e}_x, w, t) \leq 0$ one can show that all the solutions of (29)-(30) are uniformly globally bounded. Proposition 1 follows then from Theorem 2.

It is important to mention that the need for referring to Theorem 2 in this stability analysis results from the fact that, on the one hand, the considered system is time-varying and, on the other hand, the time derivative of the chosen function $V(e_x, \dot{e}_x, w, t)$ is only negative semi-definite. This situation, and the exploitation of the fact that the matrix Q can be chosen arbitrarily, are the most important differences to the stability proofs in (Lin & Goldenberg, 1995; 1996).

5.4 Proof of Proposition 2

Proposition 2 can be shown by considering $V(e_x, \dot{e}_x, w, t)$ of (31) as a candidate storage function. In case of $F_{ext} \neq 0$ the time derivative of $V(e_x, \dot{e}_x, w, t)$ along the solutions of (27)-(28) gets

$$\dot{V}(e_x, \dot{e}_x, w, t) = -\begin{bmatrix} \dot{e}_x \\ z \\ \dot{z} \end{bmatrix}^T N(e_x, t) \begin{bmatrix} \dot{e}_x \\ z \\ \dot{z} \end{bmatrix} + \dot{e}_x^T F_{ext} \ .$$

The matrix $N(e_x, t)$ has already be shown to be positive definite. From this one can immediately conclude the passivity property from Proposition 2.

6. Simulation Results

In this section a simulation study of the proposed impedance control law is presented. The controller is evaluated for a flexible joint robot model of the seven-axis DLR-Lightweight-Robot-III (Hirzinger et al., 2002) (see Fig. 1). This robot is equipped with joint torque sensors in addition to the common motor position sensors and thus is ideally suited for the verification of the presented controller. In the present simulation only the first six joints of the robot are considered.

It is assumed that the complete state of the system (10)-(11) is available and that also the (generalized) external forces can be measured by a force-torque-sensor mounted on the tip of the robot.

For the desired impedance behavior (23) diagonal stiffness and damping matrices K_d and D_d are chosen. The stiffness values for the translational components of K_d were set to a value of 2000 N/m, while the rotational components were set to 100 Nm/rad. The damping values were chosen as 150 Ns/m and 50 Nms/rad for the translational and the rotational components, respectively. The Cartesian coordinates are composed of three translational and three rotational coordinates, in which the modified Euler-angles, see, e.g., (Natale, 2003) according to the common roll-pitch-yaw representation were used.

In the following two different gains for the torque control loop are evaluated and compared to the response of the desired impedance behavior (23). In both cases the torque control gain matrices K_s and K_t were chosen as diagonal. In the first case a rather small proportional gain of $K_t = I$ was chosen, while in the second case a higher gain of

$K_t = 10 \cdot I$ was used. In both cases the torque damping matrix K_s was chosen according to an overall damping factor of 0.7 for the linear system (28).

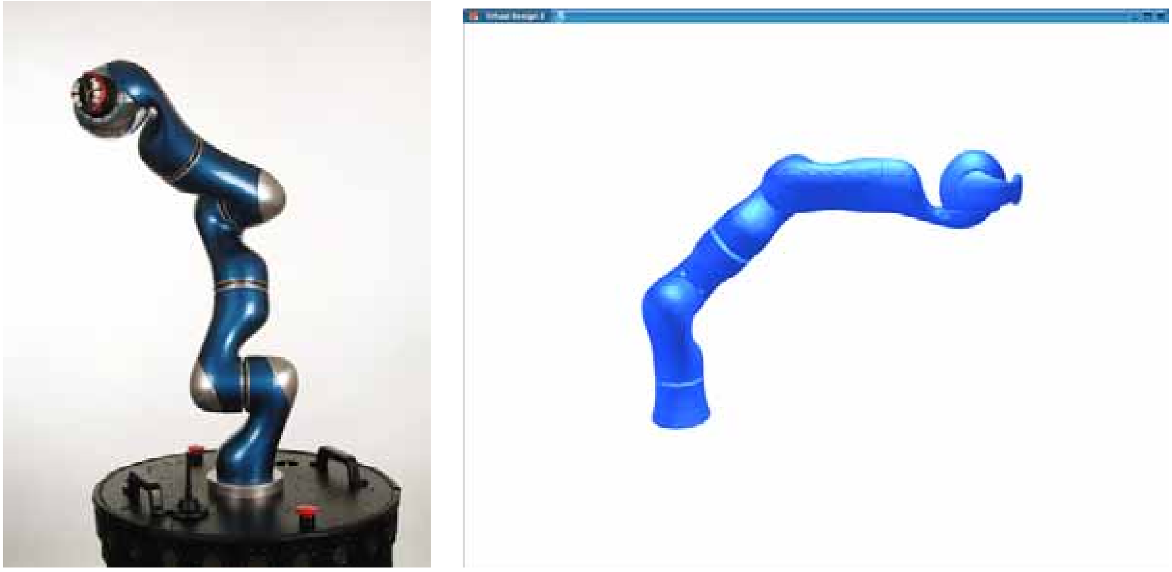
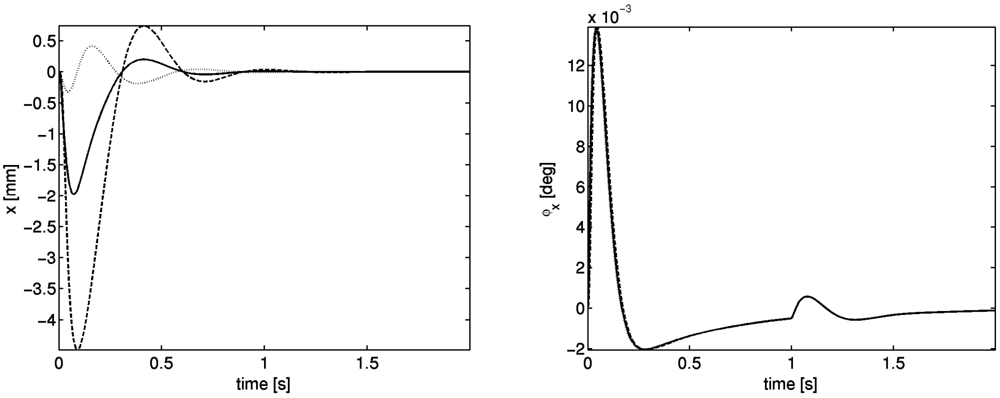


Figure 1. DLR-Lightweight-Robot-III. The picture on the right hand side shows the initial configuration of the robot in the simulation study

In the simulation two step responses are presented, starting from a joint configuration as shown at the right hand side of Fig. 1. First, a step for the virtual equilibrium position of 5 cm in z-direction is commanded at time instant $t = 0$. Apart from this step the virtual equilibrium position is constant. Then, after a delay of one second, a step-wise excitation by an external force of 1 N in y-direction is simulated. Notice that this force step results in an excitation of the link side dynamics (27) only, while the torque error dynamics (28) keeps unaffected.

Fig. 2 shows the simulation results for the two decoupling based controllers (with low and high gains) in comparison with the desired impedance behavior. The translational end-effector coordinates are denoted by x, y, and z, while the orientation coordinates are denoted by ϕ_x, ϕ_y, and ϕ_z. Due to the coupling via the (fully occupied) inertia matrix, all the coordinates deviate from their initial values in the step responses. As expected, the closed-loop behavior resembles the desired behavior better for higher torque control gains. The step of the external force, instead, does not affect the torque control loop, and



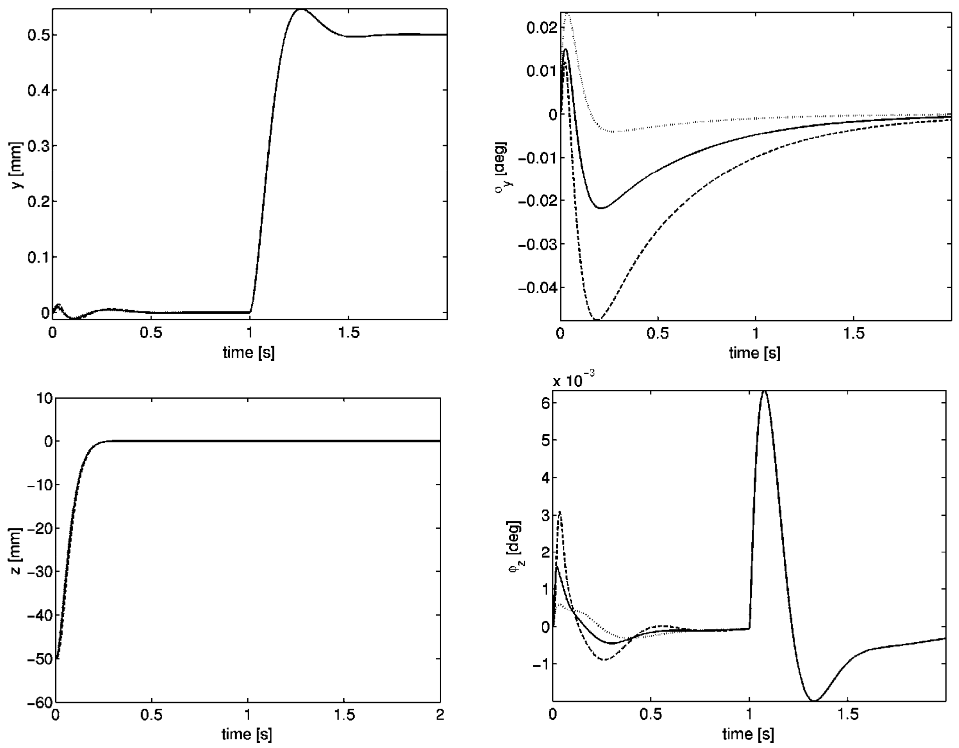
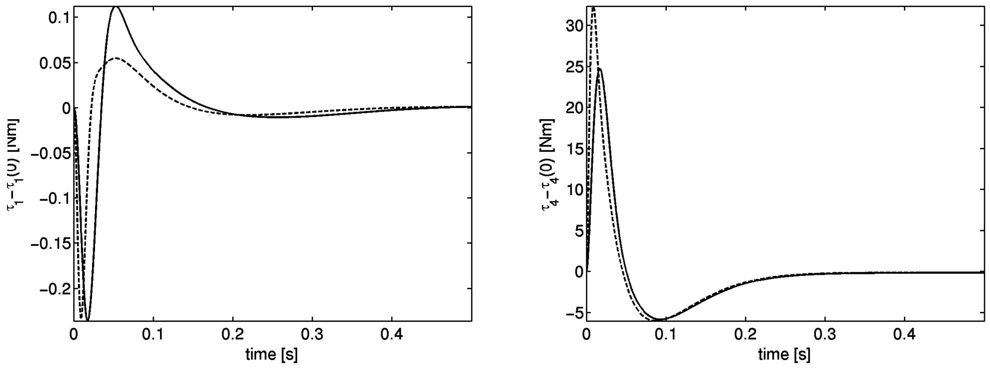


Figure 2. Simulation results for the two step responses (in the virtual equilibrium position and in the external force). The dotted lines show the simulation results of the desired impedance (23). The dashed and solid lines show the results for the decoupling based impedance controller with low and high gains, respectively

thus the closed-loop behaviors of the two controllers correspond exactly to the desired behavior for this excitation, see Fig. 2. The difference between the two controllers is also shown in Fig. 3, where a comparison of the resulting joint torques is given.

In the presented simulation study it was assumed that all the state variables, as well as the external forces, are available for the controller. In case of a typical industrial robot usually only the motor position can be measured, and the motor velocity can be computed from this via numerical differentiation. All the other state variables, including the link acceleration and the jerk, must be estimated. Some advanced modern robot arms, like for instance the DLR lightweight robots, are instead also equipped with joint torque sensors. Apart from the implementation of the inner torque feedback loop, these sensors can also be used for a more reliable estimation of the link acceleration and the jerk. Even more promising would be the use of acceleration sensors, either of a six-dof Cartesian sensor or of individual joint acceleration sensors.



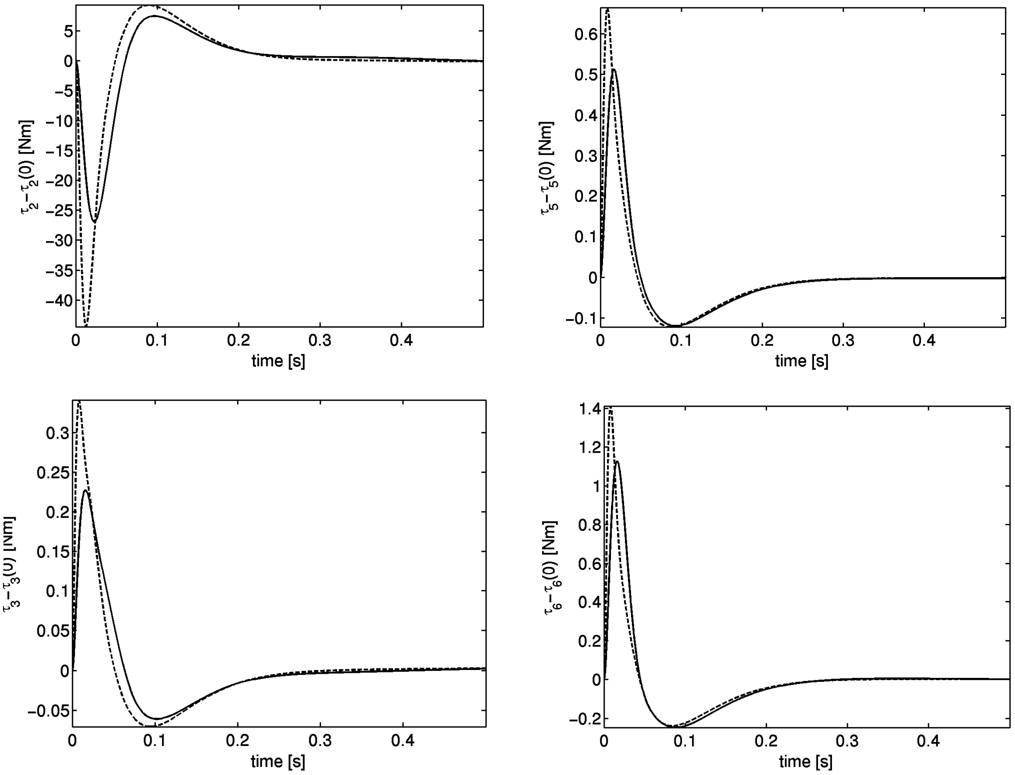


Figure 3. Deviation of the joint torques from their initial values for the decoupling based controllers. The dashed and solid lines show the results for the controller with high and low gains, respectively

7. Summary

In this work a Cartesian impedance controller for flexible joint robots was proposed. The control approach was based on the stability theory for cascaded systems. In the proposed controller an inner torque feedback loop decouples the torque dynamics from the rigid body dynamics. For the implementation of the impedance behaviour a control law well known from rigid body robotics is used in combination with the torque controller.

It should also be mentioned that, apart from the considered impedance control problem, different rigid body controllers can be applied to the flexible joint model analogously to the procedure described herein. For the proof of the asymptotic stability of the closed-loop system one can take advantage of Theorem 2.

Appendix

In this appendix, a short lemma about uniform global boundedness of the solutions of time-varying differential equations is presented, which is used in Section 5.3 for the proof of uniform global asymptotic stability of the Cartesian impedance controller.

Consider a time-varying system of the form

$$\dot{x} = f(x,t) \ , \tag{A.1}$$

with state $x \in \Re^n$. In order to show that the solutions of (A.1) are uniformly globally bounded (according to Definition A.1, which is taken from (Loria, 2001)), the following Lemma A.1 is usefull. This lemma can be proven easily based on the theorems presented in (Khalil, 2002).

Definition A.1. *The solution* $x(t,t_0,x_0)$ *of (A.1), with initial state* x_0 *and initial time* t_0*, is said to be uniformly globally bounded if there exists a class* K_∞ *function* α *and a number* $c > 0$ *such that* $\| x(t,t_0,x_0) \| \le \alpha(\| x_0 \|) + c$ *holds* $\forall t \ge t_0$.

Lemma A.1. *If there exists a continuously differentiable, positive definite, radially unbounded, and decrescent function* $V(x,t)$*, for which the time derivative along the solutions of (A.1) satisfies* $\dot{V}(x,t) = \partial V(x,t)/\partial x \cdot f(x) + \partial V(x,t)/\partial t \le 0$*, then the solutions of (A.1) are uniformly globally bounded.*

8. References

Hirzinger, G.; Sporer, N.; Albu-Schäffer, A.; Hähnle, M.; Krenn, R.; Pascucci, A. & Schedl M. (2002). DLR's torque-controlled light weight robot III – are we reaching the technological limits now?, *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 2002, pp. 1710-1716.

Hogan, N. (1985). Impedance Control: An approach to manipulation, Part I-III, *Journal of Dynamic Systems, Measurement and Control*, Vol. 107, 1985, pp. 1-24.

Horn, R. A. & Johnson, C. R. (1990). *Matrix Analysis*, Cambridge Univ. Press, 1990.

Khalil H. K. (2002). *Nonlinear Systems*, Prentice Hall, 3rd Ed., 2002.

De Luca, A. (1996). Elastic Joints, Chapter 5 in *Theory of Robot Control*, De Wit, C.; Siciliano, B. & Bastin, G. (Ed.), Springer-Verlag, London, 1996.

Lin, T. & Goldenberg, A. A. (1995). Robust Adaptive Control of Flexible Joint Robots with Joint Torque Feedback, *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 1995, pp. 1229-1234.

Lin, T. & Goldenberg, A. A. (1996). A Unified Approach to Motion and Force Control of Flexible Joint Robots, *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 1996, pp. 1115-1120.

Loria, A. (2001). Cascaded nonlinear time-varying systems: analysis and design, *Minicourse at the Congreso Internacional de Computacion*, Cd. Mexico, 2001.

Natale, C. (2003). *Interaction Control of Robot Manipulators: Six-Degrees-of-Freedom Tasks*, Springer, Springer Tracts in Advanced Robotics (STAR), 2003.

Ott, Ch.; Albu-Schäffer, A. & Hirzinger, G. (2002). Comparison of Adaptive and Nonadaptive Tracking Control Laws for a Flexible Joint Manipulator, *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2002, pp. 2018-2024.

Ott, Ch.; Albu-Schäffer, A.; Kugi, A. & Hirzinger, G. (2003). Decoupling Based Cartesian Impedance Control of Flexible Joint Robots, *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 2003, pp. 3101-3107.

Paden, B. & Panja, R. (1988). Globally asymptotically stable 'PD+' controller for robot manipulators, *Int. Journal of Control*, Vol. 47, No. 6, 1988, pp. 1697-1712.

Santibanez, V. & Kelly, R. (1997). Strict Lyapunov Functions for Control of Robot Manipulators, *Automatica*, Vol. 33, No. 4, 1997, pp. 675-682.

Sciavicco L. & Siciliano, B. (1996). *Modeling and Control of Robot Manipulators*, McGraw-Hill, 1996.

Seibert, P. & Suarez, R. (1990). Global stabilization of nonlinear cascade systems, *Systems and Control Letters*, Vol. 14, No. 4, 1990, pp. 347-352.

Spong, M.W. (1987). Modeling and Control of Elastic Joint Robots, *Journal of Dynamic Systems, Measurement, and Control*, Vol. 109, 1987, pp. 310-319.

Vidyasagar, M. (1993). *Nonlinear Systems Analysis*, Prentice Hall, 2nd Ed., 1993.

Collision-Free Path Planning in Robot Cells Using Virtual 3D Collision Sensors

Tomislav Reichenbach & Zdenko Kovacic

1. Introduction

In industrial robotic systems, it is crucial to avoid collision among the manipulators or with other objects in the workspace. The problem is more complicated for manipulators than for mobile robots, since not only the effector end must move without collision to the desired location, but the links of the arm must also avoid collisions. The problem has been addressed in various ways in literature (Lozano-Perez, 1986) and (Barraquand, Langlois & Latombe, 1992). These methods can generally be divided into two methodologies: global and local.

A global typical technique in collision avoidance consists of exploring a uniform grid in configuration-space (C-space) or configuration time space (CT space) using a best-first search algorithm guided by goal-oriented potential field (Barraquand & Latombe, 1991) and (Hwang, & Ahuja, 1992). Two main problems exist, the obstacles must be mapped into the configuration space (a very complex operation) and a path through the configuration space must be found for the point representing the robot arm. Usage of this method in dynamic complex environments, with more robots sharing the same space, is too time demanding to be accomplished in real time.

A new idea proposed in this work is to detect possible collisions among robotic system entities in a virtual world. Moreover, the idea is to use a collision detection engine, which is extensively used in computer graphics and simulation, to test for possible collisions in full 3D environment with objects consisting of thousands of polygons and modeled as similar to the real ones as close as possible. A modification of manipulator kinematics is proposed in order to control directly a colliding point, hence enforcing fastest collision avoidance. This approach has some advantages of both global and local planning methodologies, as the complete environment is considered in planning and fast collision avoidance response suitable for on-line application in dynamic environments is achieved.

Motion planning for models with fairly complex geometry has been proposed in some former papers (Cameron & Qin, 1997) with modified Gilbert, Johnson and Keerthi (GJK) algorithm applied for distance computations and collision detection, but it differs in the nature and use of the collision detection algorithm.

A software simulation package for virtual modeling of complex kinematic configurations has been developed (C++ and OpenGL). Models are created in CAD software and then imported into virtual environments described in the XML. Virtual models provide very inexpensive and convenient way for design, analysis, control, dynamic simulation and visualization of complex robotic systems. Given that virtual models are accurate, collisions in a virtual environment should occur in the same way as in the real world. The 3D models used in this work are polygonal structured, i.e. polygons form a closed manifold, hierarchical non-convex models (Lin & Gottschalk, 1998) undergoing series of rigid-body transformations. Polygons are made entirely of triangles, as hardware accelerated rendering of the triangles is commonly available in the graphic hardware. All the parameters (link positions, link lengths) for a direct kinematics solution are extracted from a 3D description.

This chapter is organized as follows. First, a brief introduction to kinematic models is given, then a method for collision detection among objects in virtual environment is explained along with an algorithm for determination and generation of new kinematic parameters for the colliding objects. A modification of manipulator kinematics is proposed in order to control directly a colliding point, hence enforcing fastest collision avoidance. Following is the description of collision avoidance strategy and the results of simulations of collision free trajectory planning in the experimental Flexible Manufacturing Systems (FMS).

2. Kinematic Model

The endpoint position and orientation of a manipulator may be expressed as a nonlinear function of the constants such as physical link parameters, and by joint angles or displacements that are variables. To represent the spatial transformations between the joints, several symbolic notations have been proposed. The most widely used notation is the classic Denavit-Hartenberg (D-H) notation (Denavit & Hartenberg, 1955) or its Paul (Paul, 1981) version. These notations however are only valid for the kinematic chains with one branch (i.e. serial kinematic chains), and can lead to ambiguities when dealing with kinematic chains with more then one branch (parallel kinematic chains). The Sheth-Uicker (S-U) (Sheth & Uicker, 1971) notation extends D-H notation for multiple loop kinematic chains in the general case, but is much more complicated because it introduces additional coordinate systems. Another notation presented by Kleinfinger (Khalil & Kleinfinger, 1986), is usable for all serial, treelike or closed loop kinematic chains, and has fewer parameters then the S-U notation, however, since all manipulators employed in this chapter are serial kinematic chains, only D-H notation, albeit in a somewhat modified form, is used.

3. Kinematic Parameters

Six parameters are needed to describe a complete relation between two independent coordinate systems - frames. Like most of the other kinematic notations, D-H notation requires following some rules when designating axes and orientations to the link frames.

3.1. Modified D-H Notation

Assuming that robot links have only one dimension, the length, the classic D-H notation describes the complete relation between two adjacent link frames with only four kinematic parameters $[a_i \quad d_i \quad \alpha_i \quad \theta_i]$. The meaning of these parameters is the following: **rotation** around a z^{i-1} axis with an angle θ_i, **translation** along a z^{i-1} axis with a displacement d_i, **translation** along a $x^i = x^{i-1}$ axis with a displacement a_i, **rotation** around a x^{i-1} axis with an angle α_i.

For two coordinate systems having an arbitrary position and orientation in the 3D space, the conversion between them can be described with a matrix $\mathbf{M}_i$ given in the [4x4] homogenous matrix form,

$$\mathbf{M}_i = \mathbf{T}_i \cdot \mathbf{R}_i = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_1 \\ r_{12} & r_{22} & r_{13} & p_1 \\ r_{13} & r_{23} & r_{13} & p_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1}$$

where

$$\mathbf{T}_i = \begin{bmatrix} 1 & 0 & 0 & p_1 \\ 0 & 1 & 0 & p_1 \\ 0 & 0 & 1 & p_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \mathbf{R}_i = \begin{bmatrix} r_{11} & r_{12} & r_{13} & 0 \\ r_{12} & r_{22} & r_{13} & 0 \\ r_{13} & r_{23} & r_{13} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

To convert from matrix **M***i*, parameter b_i representing a link length along the y^{i-1} axis of the *i*th coordinate system, has been added. Now, there are 5 independent parameters $[a_i \;\; b_i \;\; d_i \;\; \alpha_i \;\; \theta_i]$ used in modified D-H notation. Only one of these parameters is a variable, joint position (θ_i if the joint is rotational or d_i if the joint is prismatic) and the other 4 (or 3 in the original D-H notation) are fixed parameters determined only by a manipulator construction. In this way, the D-H transformation matrix between two adjacent link frames has the following form:

$${}^{i-1}\mathbf{T}_i = \begin{bmatrix} \cos\theta_i & -\cos\alpha_i \sin\theta_i & \sin\alpha_i \sin\theta_i & a_i \cos\theta_i - b_i \sin\theta_i \\ \sin\theta_i & \cos\alpha_i \cos\theta_i & -\cos\alpha_i \sin\alpha_i & a_i \sin\theta_i + b_i \cos\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2}$$

To get D-H parameters $[a_i \;\; b_i \;\; d_i \;\; \alpha_i \;\; \theta_i]$ for the *i*th link from a joint frame transformation matrix ${}^{i-1}\mathbf{T}_i$ written in the form as in eq. (2), a correct solution has to be chosen from a solution set $S = \{\alpha_i^1, \alpha_i^2, \alpha_i^3, \theta_i^1, \theta_i^2, \theta_i^3\}$. Because of the ambiguity of solutions of trigonometric equations, several possible solutions are available:

$$\begin{aligned} \alpha_i^1 &= atan2({}^{i-1}\mathbf{T}_i(3,2), {}^{i-1}\mathbf{T}_i(3,3)) \\ \alpha_i^2 &= atan2(-{}^{i-1}\mathbf{T}_i(2,3), {}^{i-1}\mathbf{T}_i(2,2)) \\ \alpha_i^3 &= atan2({}^{i-1}\mathbf{T}_i(1,3), -{}^{i-1}\mathbf{T}_i(1,2)) \\ \theta_i^1 &= atan2({}^{i-1}\mathbf{T}_i(2,1), {}^{i-1}\mathbf{T}_i(1,1)) \\ \theta_i^2 &= atan2(-{}^{i-1}\mathbf{T}_i(1,2), {}^{i-1}\mathbf{T}_i(2,2)) \\ \theta_i^3 &= atan2({}^{i-1}\mathbf{T}_i(1,3), -{}^{i-1}\mathbf{T}_i(2,3)) \end{aligned} \tag{3}$$

4. Direct Kinematics

A transformation from the *n*th local frame to the global frame is defined as:

$${}^{0}\mathbf{T}_n = \prod_{k=0}^{n} \left(\mathbf{M}_k \cdot \mathbf{M}_{jk} \right) \tag{4}$$

$$\mathbf{M}_k = \mathbf{T}_k \cdot \mathbf{R}_k$$

Where:

- $\mathbf{T}_k$ is kth joint translation matrix.
- $\mathbf{R}_k$ is kth joint rotation matrix.
- $\mathbf{M}_{jk}$ is kth joint transformation matrix.

Matrix $\mathbf{M}_{jk}$ is defined as:

$$\mathbf{M}_{jk} = \begin{bmatrix} \cos\beta_k \cos\gamma_k & -\sin\gamma_k & \sin\beta_k & l_1 \\ \sin\gamma_k & \cos\alpha_k \cos\gamma_k & -\sin\alpha_k & l_2 \\ -\sin\beta_k & \sin\alpha_k & \cos\alpha_k \cos\beta_k & l_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5}$$

where α_k, β_k and γ_k are the rotation angles around the respective x_k, y_k, z_k axes, and l_1, l_2, l_3 are translations along the respective x_k, y_k, z_k axes of the kth coordinate system. If the system is using D-H notation, then all joint rotations and translations are done around and along the local z_k-axis, so the eq. (5) takes the following form:

$$\mathbf{M}_{jk} = \mathbf{M}_{qk} = \begin{bmatrix} cos\gamma_k & -\sin\gamma_k & 0 & 0 \\ \sin\gamma_k & \cos\gamma_k & 0 & 0 \\ 0 & 0 & 1 & l_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos q_k & -\sin q_k & 0 & 0 \\ \sin q_k & \cos q_k & 0 & 0 \\ 0 & 0 & 1 & q_k \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6}$$

where q_k is a joint variable defined as $q_k = (1-\xi_k)\cdot d_k + \xi_k \cdot \theta_k$. If the joint is rotational then $\xi_k = 1$, else it is prismatic and $\xi_k = 0$. Due to high complexity of 3D models, there are more coordinate systems than joints. All coordinate systems that do not undergo rigid-body motion are considered static with matrix $\mathbf{M}_{qk}$ equal to the unity matrix. A direct kinematic solution, a tool position and an orientation, are easily extracted from eq. (4).

5. Collision Detection

Collision detection is a part of *interference detection*, which can be divided into three portions: collision detection that detects whether objects are in collision, collision determination that finds the exact collision point, and finally, collision avoidance that determines what actions should be taken in response to the collision. There are numerous approaches to a collision detection problem (Jimenez, Thomas & Torras, 2001), which can be mainly grouped into; space-time volume intersection, swept volume interference, multiple interference detection, and trajectory parameterization. A collision detection algorithm used in this work belongs to a multiple interference detection category. This implies that the algorithm reduces a general collision detection problem to multiple calls to the static interference tests focused on detecting intersections among simple geometrical entities; triangles and oriented bounding boxes (OBB) belonging to the objects being tested. As the algorithm is static, i.e. collision detection occurs only at discrete times, it is fast enough and effective from the computational point of view, thus it can provide real-time collision detection in very complex (high polygon count) virtual environments.

5.1. Oriented Bounding Boxes (OBB)

An oriented bounding box (OBB) can be described by the center point of the box $\mathbf{b}^c$, and a center rotation vector $\mathbf{b}^r$ (or three normalized vectors $\mathbf{b}^u$, $\mathbf{b}^v$, $\mathbf{b}^w$ that represent the face normals). To express the OBB dimensions the most compact representation is with the half-lengths h_u^B , h_v^B , h_w^B , which represent dimensions along the respective u, v, w axis.

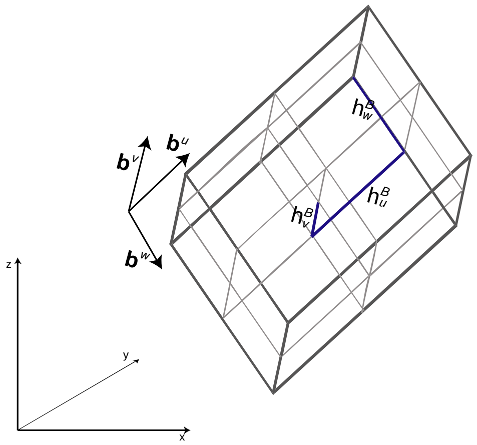


Figure 1. Oriented bounding boxes

An intersection test between two OBBs A and B is accomplished with a fast routine presented in (Gottschalk, Lin & Manocha, 1996). This routine is based on separating axis theorem and is about an order of magnitude faster than methods that use linear programming or closest features. According to the separating axis theorem, it is sufficient to find one axis that separates objects A and B to determine they do not overlap. The total number of tested axes is fifteen, three from the faces of A, three from the faces of B and nine from combination of edges from both OBBs (3x3). The potential separating axes that should be orthogonal to the faces of A and B so the simplest way is to use the normals to the faces $\mathbf{a}^u$, $\mathbf{a}^v$, $\mathbf{a}^w$ and $\mathbf{b}^u$, $\mathbf{b}^v$ and $\mathbf{b}^w$. The remaining nine potential axes are formed by one edge from each OBB, $c^{ij} = a^i \times b^j, \forall i \in \{u, v, w\}$ and $\forall j \in \{u, v, w\}$. The OBBs are then projected onto the axes to see if both projections overlap on all axes, i.e. if the OBB A and B overlap. The maximum number of operations is reached when the two OBBs overlap, and is around 180 (without the transform of B into A's coordinate system) (Gottschalk, 2000). However, in most cases without overlap, a separating axis is found earlier.

5.2. Triangle/triangle Intersection

After the intersection between the OBBs is determined, an exact collision point is found with triangle/triangle intersection test (see Fig. 2). Often not only the information whether two triangles intersect but also their exact intersection point is needed. Several methods for triangle/triangle intersection test are available, with two of the fastest being the *interval overlap method* developed by (Möller, 1997) and the triangle/triangle intersection method found in ERIT package (Held, 1997).

The ERIT's method of detecting whether two triangles T_1 and T_2 intersect, can be briefly summarized as:

1. Compute $\pi_2 : \mathbf{n}_2 \cdot \mathbf{x}$, the plane in which T_2 lies.
2. Trivially reject if all points of T_1 are on the same side of π_2
3. Compute the intersection between π_2 and T_1
4. If this line segment intersects or is totally contained in T_2 , then T_1 and T_2 intersect; otherwise, they do not.

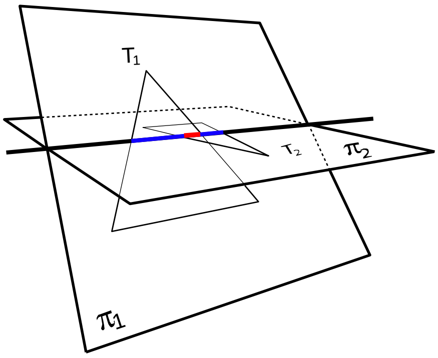


Figure 2. Triangle/triangle intersection

In some occasions, it is not necessary to determine the exact collision point, but only to check if objects are in collision. If the object is distant it can be approximated with just one OBB (see Fig. 3) and consequently collision detection will be faster. The balance between fast collision detection and exact collision point determination is the case shown in Fig. 4. The 3rd hierarchy level OBBs are used for objects that are near each other (see Fig. 5). Normally, a manipulator will not contain higher levels of OBBs hierarchies (4th, 5th, etc...), since they would not provide computational advantage over a triangle/triangle intersection test. Binary trees are often used for the hierarchy representation and different strategies for hierarchy building are available, e.g. K-DOPTtree, OBBtree (Gottschalk, Lin & Manocha, 1996). The OBBtree approach is the most similar to the one used here. A depth of hierarchical tree, and a decision when the triangle/triangle test will be used instead of OBB overlap check, can be conditioned by available computational time and the complexity of the model.

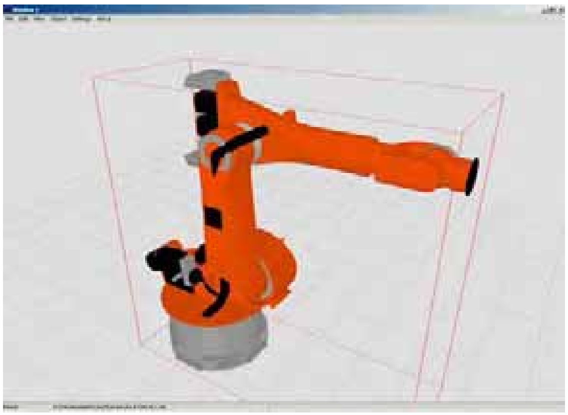


Figure 3. 1^{st} level of OBB hierarchy

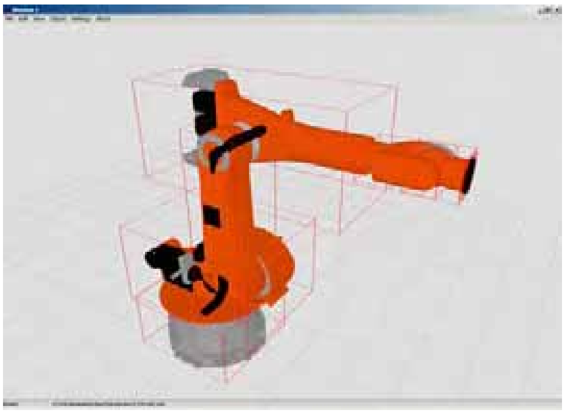


Figure 4. 2^{nd} level of OBB hierarchy

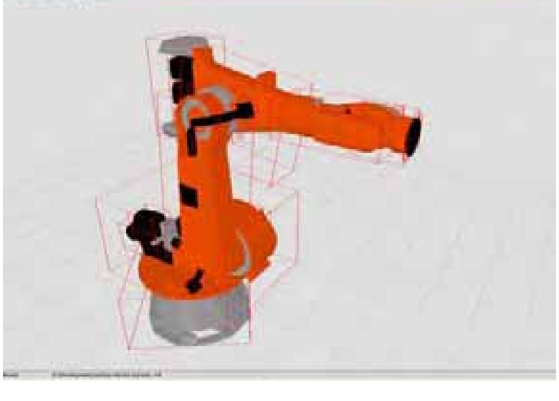


Figure 5. 3^{rd} level of OBB hierarchy

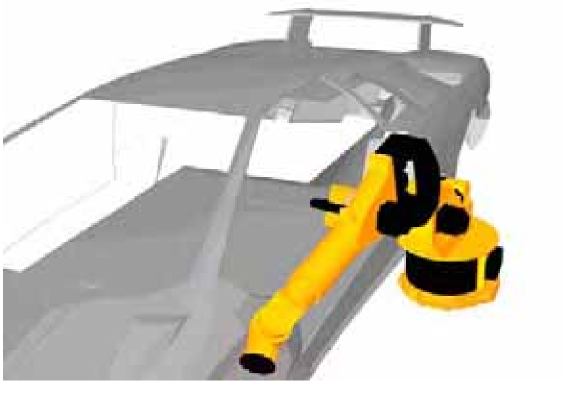
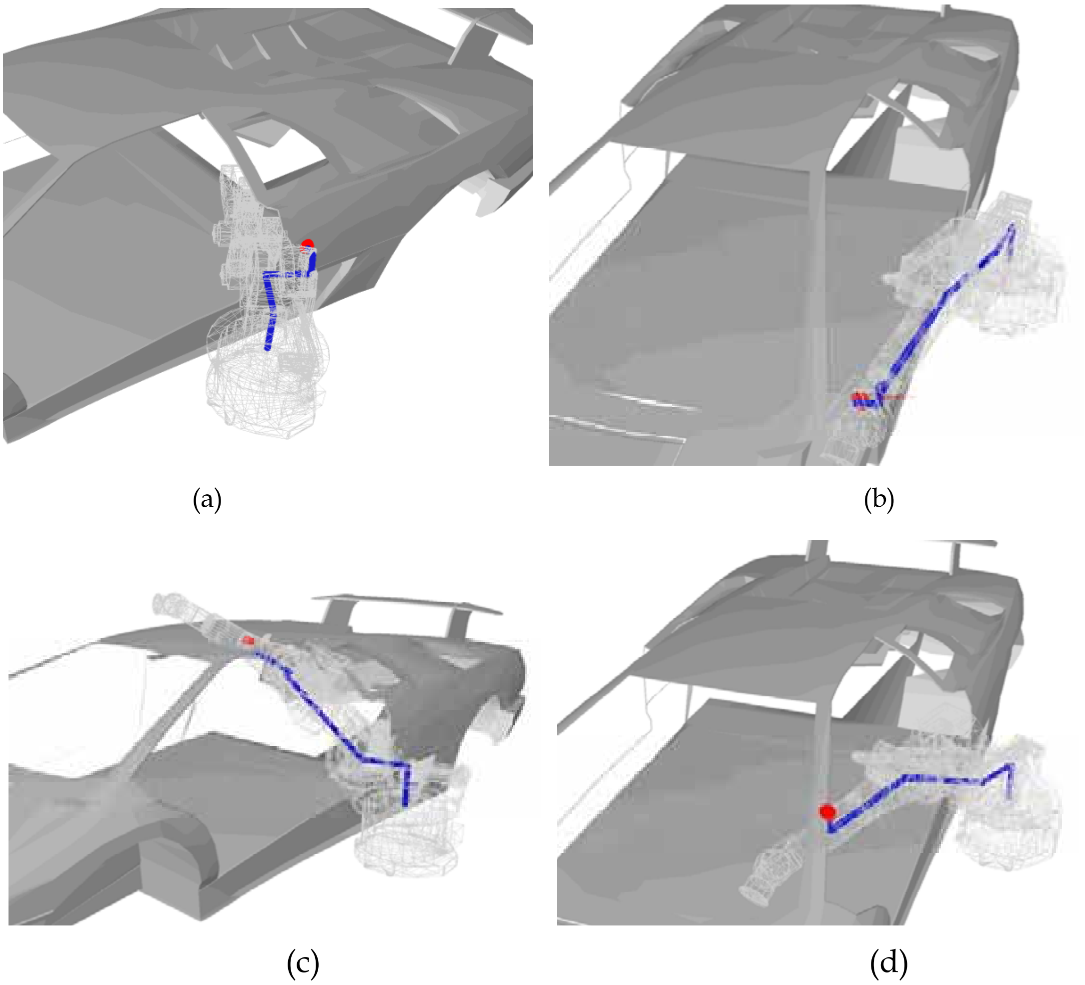


Figure 6. Manipulator prior to the collision

The next logical step is to use information about the intersection point (i.e. collision) and try to prevent the collision by changing the path of one or both colliding objects. Assuming that at least one colliding object is a robot, one must know kinematic parameters of the robot to be able to prevent collisions and plan collision-free robot trajectories. For that purpose regular kinematic parameters associated with positions and orientations of all robot joints and end effectors are not sufficient for proper collision-free trajectory planning. Namely, these parameters do not describe all points on the robot surface that could collide with the environment. While in practice, determination of kinematic parameters for an arbitrary collision point on the real robot surface is a very difficult goal, in the virtual environment this may be resolved in an elegant way by using a kinematic model of the robot derived from its virtual 3D model (Reichenbach & Kovačić, 2003). Once the triangle/triangle intersection test has established the exact collision point, the determination of a link in the hierarchy where collision will take place is straightforward. If more than one link is in the collision, the link that is higher in the object hierarchy is preferred. A new D-H kinematic model is generated from eq. (3) with the collision point serving now as a tool frame origin (see Figs. 6 and 7), so inverse kinematics for the collision point may be calculated.



(a)

(b)

(c)

(d)

Figure 7. Different kinematic chains determined by a collision point

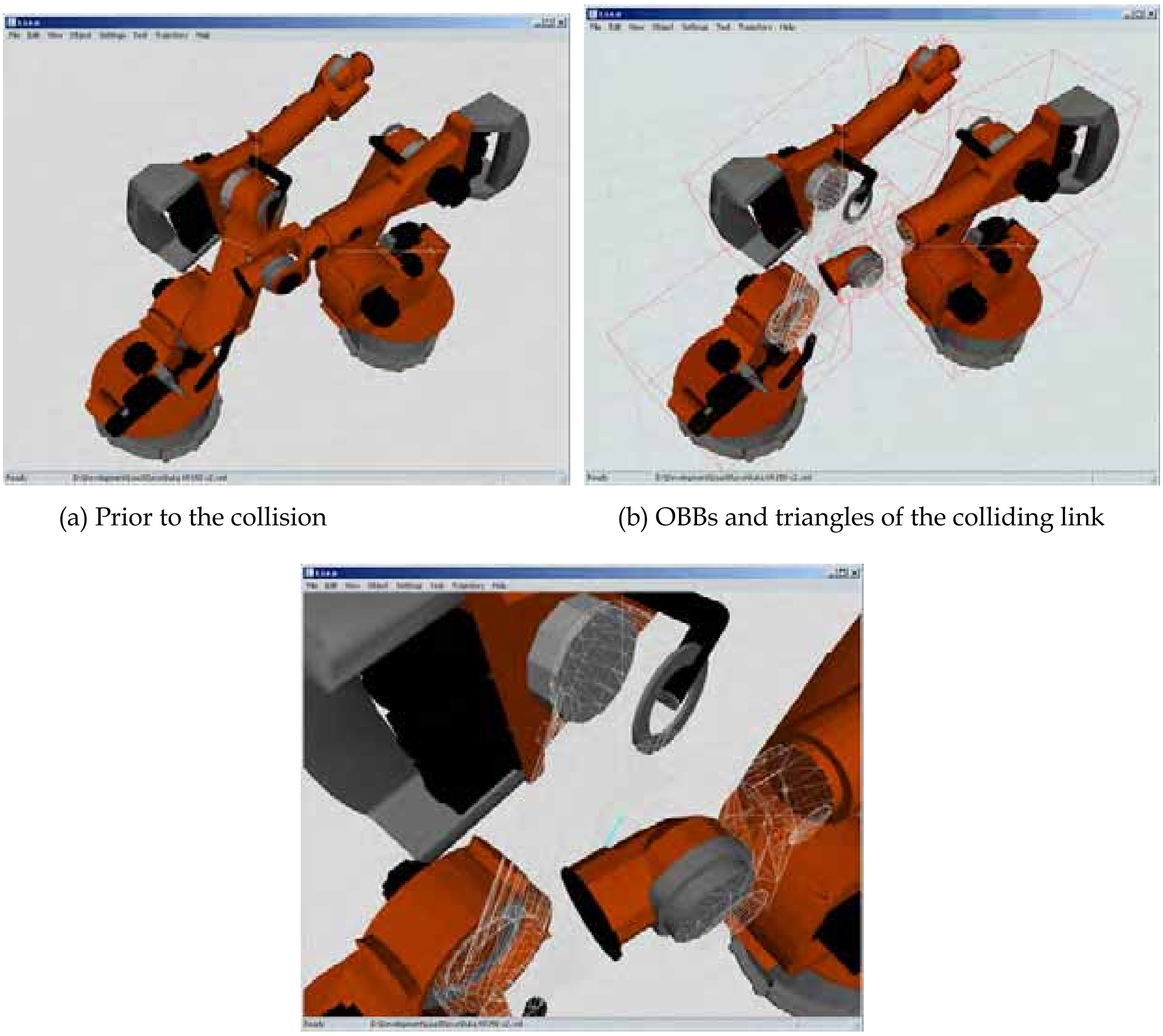
The number of collision checks in one step is:

$$N_{col} = \binom{N_{dyn}}{2} + N_{dyn} \cdot N_{stat} + N_{link} \quad , \tag{7}$$

where N_{dyn} is a number of dynamic objects, N_{stat} is a number of static objects and N_{link} is number of links in a manipulator. Further reduction in the number of collision checks can be achieved by using a *sweep and prune* technique, which exploits the temporal coherence, normally found in virtual environments (Klosowski, 1998) and (Möller & Rundberg, 1999). Temporal coherence (or frame-to-frame coherence) means that objects undergo small changes in their position and orientation between two consecutive frames.

6. Collision Avoidance

Oriented bounding boxes (OBB) are used to determine the distance and the collision between different objects at the first hierarchical stage. As one moves down the generated OBB hierarchy-tree a search for the collision point is narrowed, thus finally allowing the exact collision point determination with triangle/triangle intersection test at the final overlapping OBB nodes (see Fig. 8). How deep is the hierarchical tree, or when the triangle/triangle test will be used instead of OBB overlap check, can be specified depending on the computational time available and the complexity of the model.



(a) Prior to the collision

(b) OBBs and triangles of the colliding link

(c) Close-up

Figure 8. Two KukaKr150 robots in collision

Required trajectory of a manipulator is checked against possible collisions and if a collision is detected in some imminent manipulator position, the manipulator link is moved away from the possible collision point. Newly found collision-free points are then inserted into the previous trajectory and on-line re-planning of the trajectory is made. A complete check for all points in the trajectory and a trajectory re-planning are considered as one iteration of a collision avoidance algorithm. Iterative collision avoidance actions are taken until trajectories become collision-free.

Different collision avoidance strategies are proposed and tested. One strategy iteratively moves the first collision point for a predetermined displacement value while direction of the displacement is calculated based on collision point. A value of the displacement is predetermined according to the sparsity of objects in an environment, with larger movements possible in sparser environments. The algorithm proceeds to the next iteration until there are no collisions in the trajectory of the manipulator. Another possible strategy is to move all collision points simultaneously in one iteration for a predetermined displacement value. A strategy, that moves the middle[1] collision point for a minimum distance required to evade the collision, is presented in section 7. The avoidance movement is made in a direction of the normal to the colliding surface (determined by a collision detection algorithm) and the value of displacement is calculated as the distance from the collision surface to the end of the link along the kinematic chain. In addition, the amount of displacement may be incremented by the projection of the colliding link OBB to the direction of collision avoidance movement and the direction of general movement between the points in the trajectory (see Fig. 9).

Inverse kinematics calculus at this point is done with modified manipulator kinematics, with the collision point serving as a tool position, and only links higher in a hierarchy from the collision link are moved. In the environments with dynamic objects it is possible to estimate the time interval when collision is likely to occur, by observing how far the objects are and how rapidly they move. Collision tests are then focused on this interval. A trajectory planning is done on-line, according to the algorithm proposed by (Ho & Cook, 1982) and (Taylor, 1979) taking into account maximum possible joint velocities and accelerations.

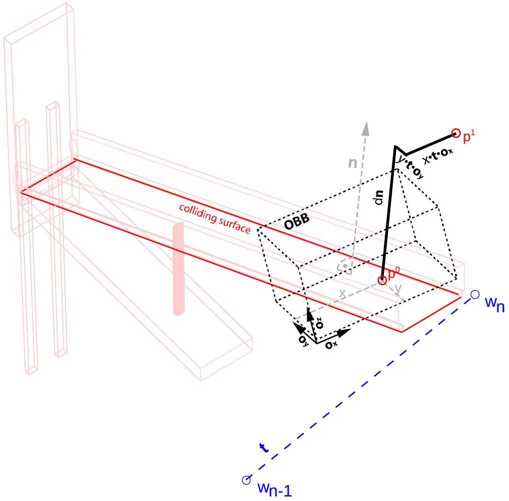


Figure 9. A displacement calculation

When a trajectory planning is made, a volume swept by robot is checked against possible collisions. While the swept volume collision checks are made, an off-line trajectory planner may normally operate with a deeper hierarchical OBB level, due to a different amount of the computational time available, than an on-line trajectory planner. During an on-line

[1] Point in the middle of a continuous collision stretch is referred as the middle point

search for a collision-free path, a progressive hierarchical OBB level approach is used. Objects that are considered far from each other are tested only in the first level OBB tree hierarchy. As objects are approaching to each other, deeper hierarchical OBB trees are used to check against collisions. Further improvement of the search for a collision-free path is made by reducing the number of checks for the moving objects. The projection of object face normals to a relative velocity vector of the object must be positive, similarly to what is proposed in (Kheddar, Coquillart & Redon 2002), otherwise the object is not checked for collisions[2].

7. FMS Control Application

A controlled environment is static in the sense that all positions, dimensions and velocities of objects are known, but objects can be dynamic, i.e. they can undergo rigid-body transformations. In a constantly changing and partially unpredictable environment on-line trajectory planning must be used. In the FMS, trajectories are currently planned off-line, which causes serious limitations in reassignments to new tasks and results in time-consuming coordination rules. The FMS resources are required to move from a job before it is completed (pre-emption property), and the process should not have to hold the resources already allocated to it until it has all resources required to perform a task (hold while waiting property).

A target FMS (see Figs. 10 and 11) contains two educational robots Rhino XR-3 and Rhino XR-4. There could be the case where the Rhino XR-4 robot is requested to move from a conveyer belt pickup/release place to one x-y table position. During this task, the Rhino XR-4 robot can also hold some object in its gripper. In the regarded FMS testbed, one of the obstacles is the gravitational buffer. In case of manipulator holding objects in its gripper, other FMS elements could also become obstacles, as it is shown in section 7.4 where x-y table is the first obstacle RhinoXR4 robot could collide with during its movement. The collision can be prevented by using the collision avoidance strategy in which the distance from the collision point and the extremes of a colliding object is calculated and directly used for the collision avoidance maneuver.

The idea is to use virtual models as a part of supervisory control of the target FMS. All sensor data are processed in the way that the virtual objects can be moved exactly as the real ones, while at the same time testing for possible collisions and employing the collision avoidance strategies when necessary. The virtual supervisory control provides collision-free trajectories generation having only the trajectory start position and the trajectory end position, thus eliminating the need to specify additional way points in the trajectory planning.

The results of several different simulation experiments and their comparison are presented in the following subsections. In addition, the influence of end-effector construction and the influence of the conveyed object is discussed (section 7.5). Different end-effectors, or different objects moved by the manipulator, produce different trajectories. With a growing complexity in the construction of the end-effector of the manipulator, the collision avoidance maneuver computational time and planned trajectory complexity increase.

[2] The approach is based on the following premise, that the objects that are moving away from each other are not checked for collisions

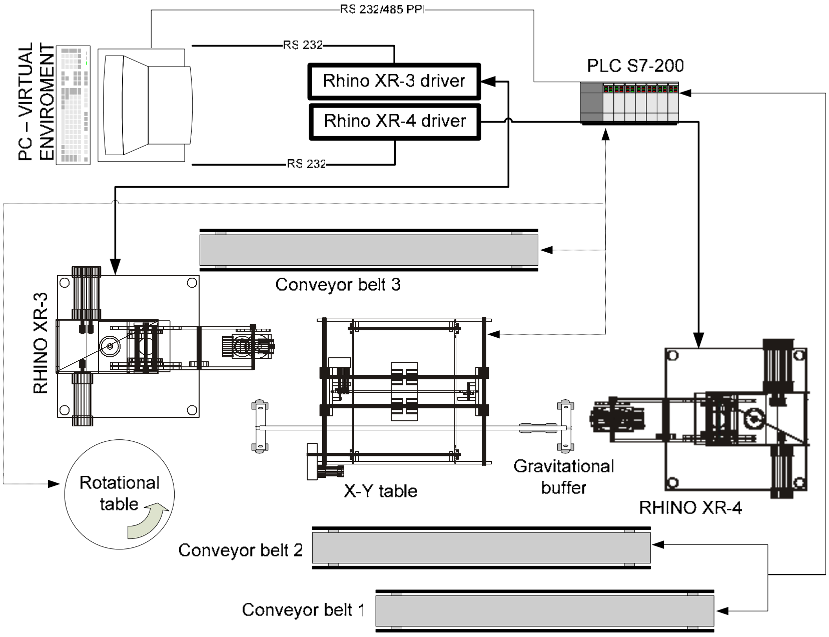


Figure 10. FMS Schematics

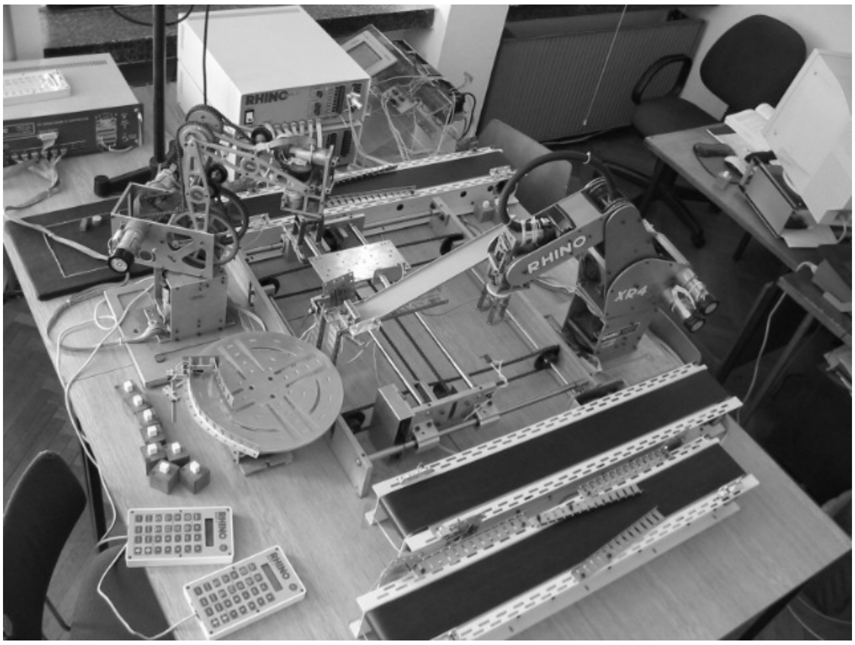
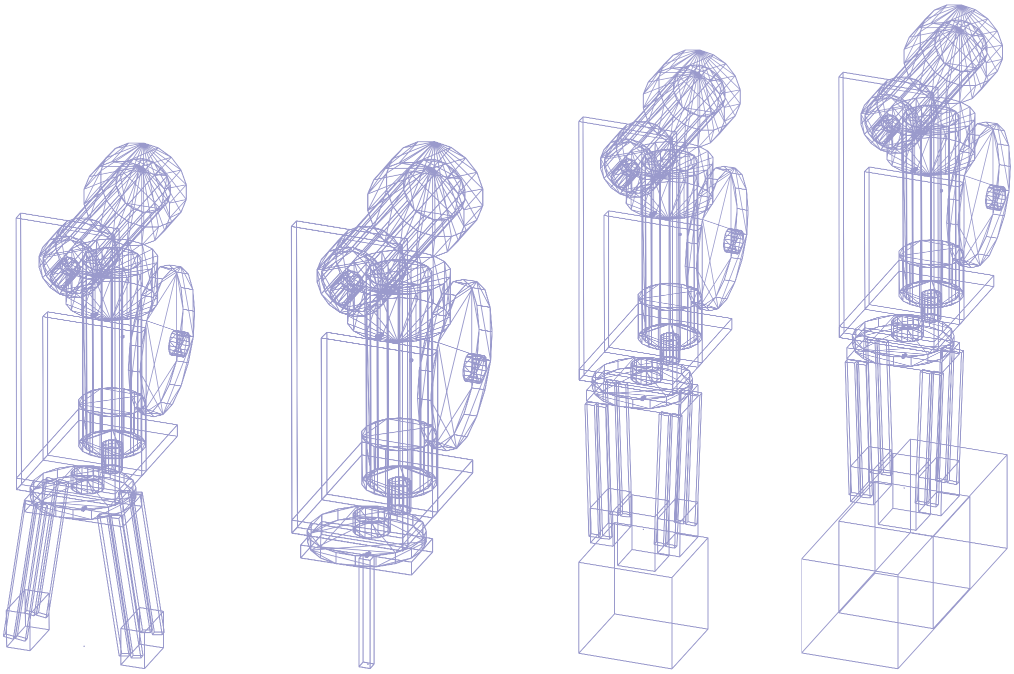


Figure 11. Laboratory Rhino FMS

There are four different end-effectors used in the simulations (see Fig. 12). Actually, the end-effector type-I carrying two different objects in its gripper represents end-effectors type-III and type-IV, respectively. Although they are not new end-effectors, for the purpose of collision evasion they may be considered as completely different end-effector configurations. The problem of one trajectory becoming invalid when tool is changed becomes irrelevant, as automatic adaptation to kinematic changes, i.e. different end-effector configurations, is made during the collision avoidance.



(a) Type-I (b) Type-II (c) Type-III (d) Type-IV

Figure 12. Tool configurations

7.1 Planning with End-Effector Type-I

In the first experiment a collision avoidance trajectory planning is done with the end-effector type-I attached to a manipulator (see Fig. 12a). Joint values of the Rhino XR-4 robot are shown in Fig. 16. In the first iteration of collision avoidance trajectory planning, collisions are continuous along the trajectory stretch where RhinoXR-4 robot is moving near the gravitational buffer (see Figs. 16a and 17a). Actual manipulator positions can be observed in Fig. 13 (complete system) and in Fig. 18a. In order to depart from the collision point the manipulator is moved in a direction pointed by the normal to the colliding surface of the gravitational buffer (detected in the collision stage).

A movement amount is calculated as a distance between a collision point and a tool tip, in order that this point becomes collision-free in the next iteration of trajectory planning. A newly inserted way point, determined by the direction of the normal and the calculated movement amount, has significantly decreased a number of collision points. Joint values of the manipulator in the 2nd trajectory planning iteration are shown in Fig. 16b.

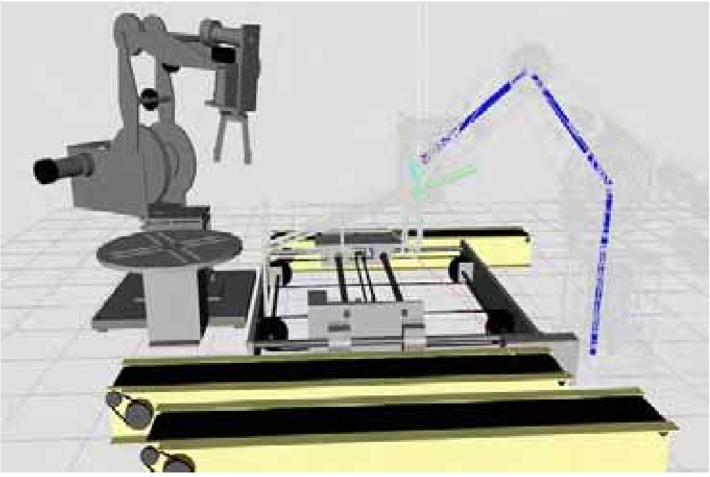


Figure 13. RhinoXR4 robot in collision with gravitational buffer 1st iteration

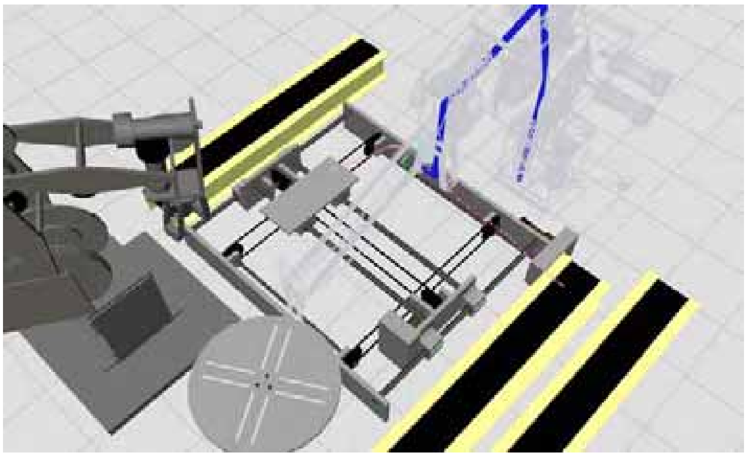


Figure 14. RhinoXR4 robot in collision with gravitational buffer 2nd iteration

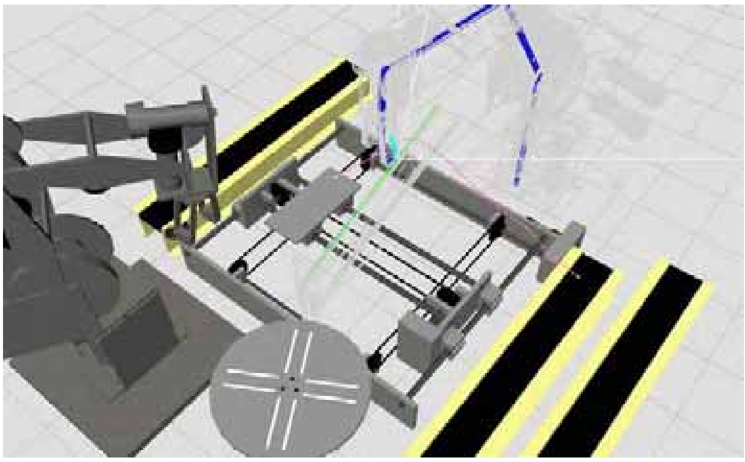
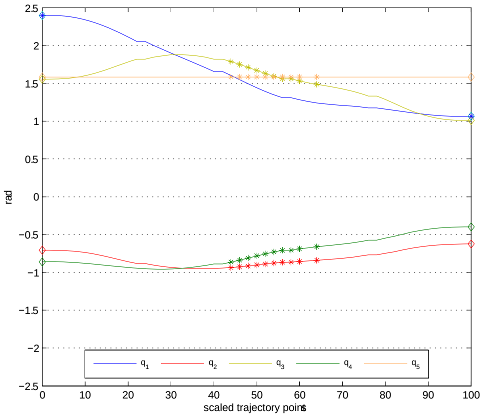
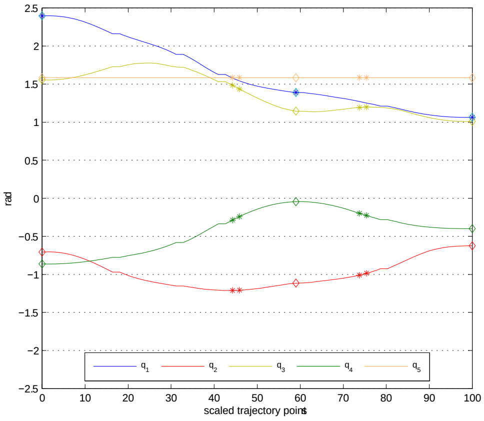


Figure 15. RhinoXR4 robot in collision with gravitational buffer 3rd iteration

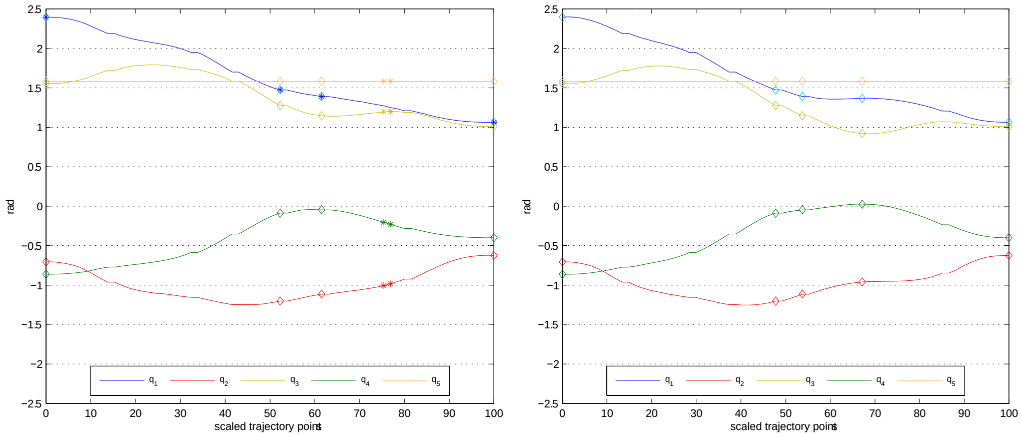
Now, there are only two small trajectory stretches where the collision is still present. Subsequent two iterations (see Figs. 16c and 16d) insert two additional way points and eliminate all collision points from the trajectory. It may be valuable to observe that the 3rd and the 4th iteration are necessary because of the position of the end-effector and its construction. In Figs. 14 and 15, one can notice that only the pincers of the end-effector (gripper tool) are colliding, but otherwise manipulator is outside the collision area. The manipulator positions throughout the iterations of collision-free trajectory planning are shown in Fig. 18. A complete trajectory evolution in the Euclidian space R^3 can be seen in Fig. 25a.





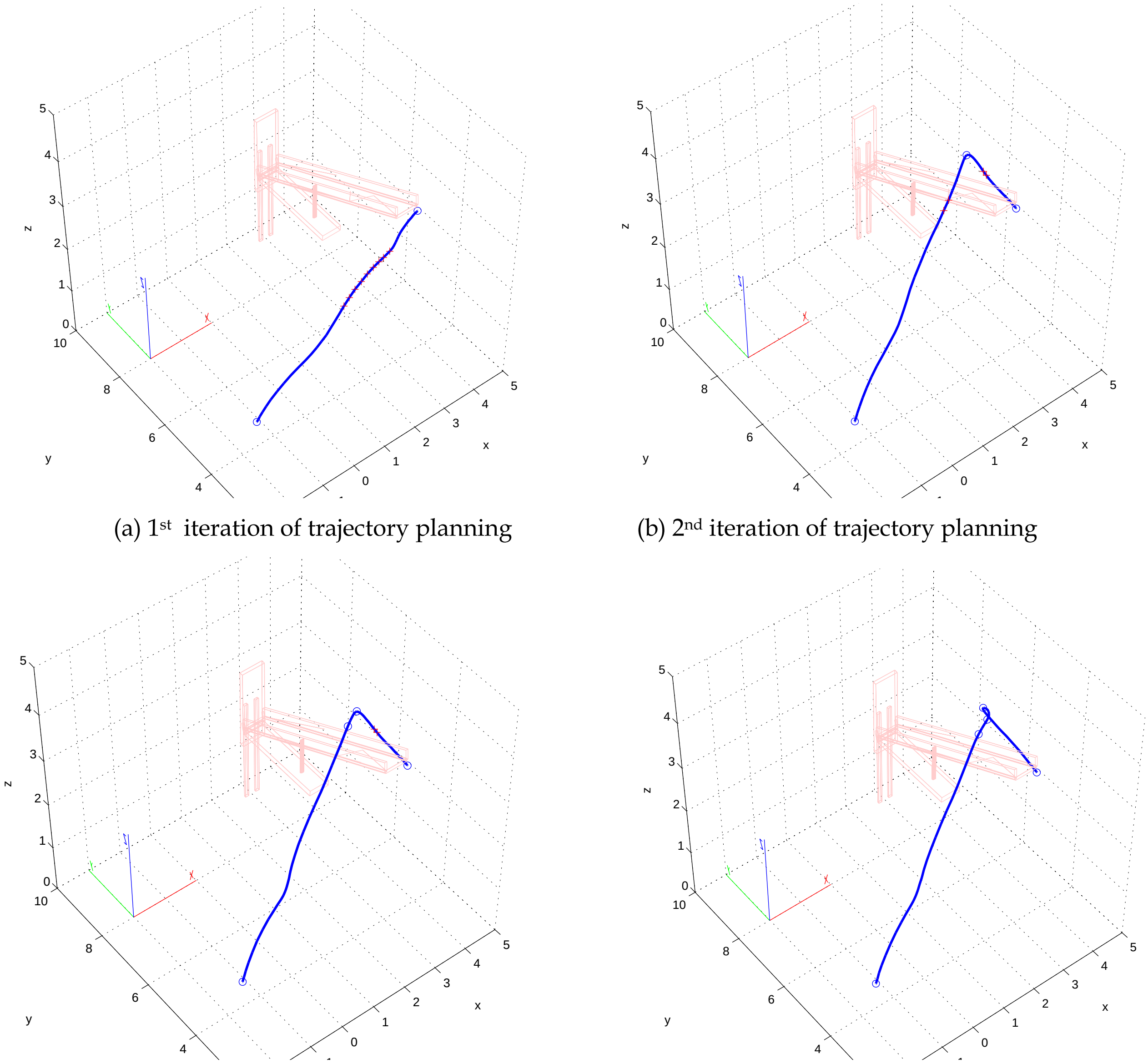
(a) 1st iteration of trajectory planning

(b) 2nd iteration of trajectory planning



(c) 3rd iteration of trajectory planning (d) 4th iteration of trajectory planning

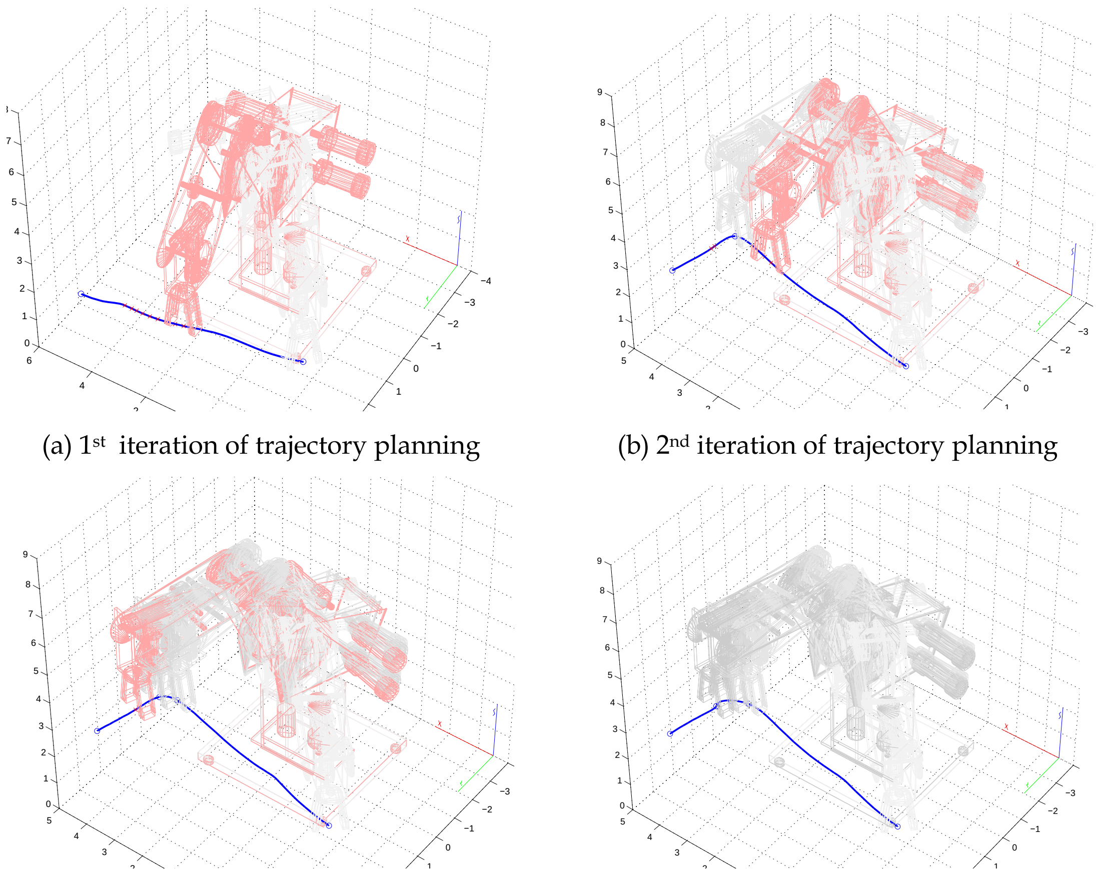
Figure 16. Joint values throughout the iteration in collision-free trajectory planning



(a) 1st iteration of trajectory planning (b) 2nd iteration of trajectory planning

(c) 3rd iteration of trajectory planning (d) 4th iteration of trajectory planning

Figure 17. End-effector positions throughout the iteration in collision-free trajectory planning



(a) 1st iteration of trajectory planning

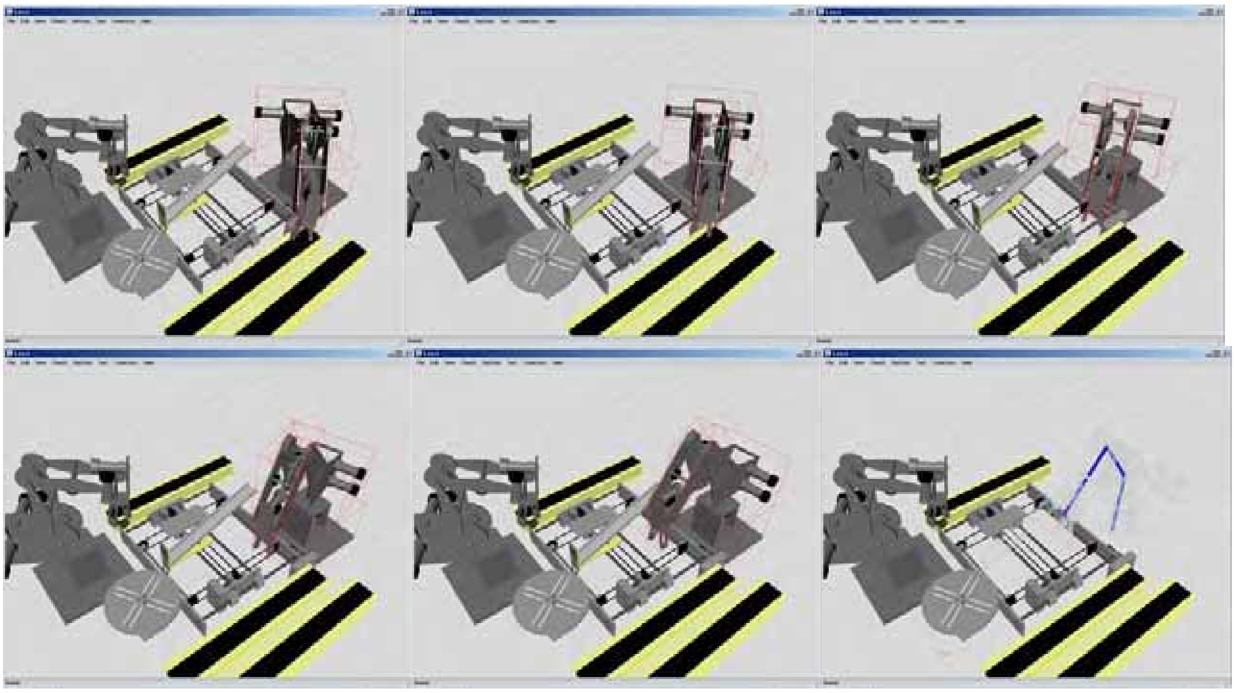
(b) 2nd iteration of trajectory planning

(c) 3rd iteration of trajectory planning

(d) 4th iteration of trajectory planning

Figure 18. Manipulator positions throughout the iteration in collision-free trajectory planning

The complete trajectory before the collision avoidance is shown in Fig. 19 and we can see the trajectory stretch where RhinoXR-4 robot is colliding with the gravitational buffer. The final trajectory generated by the collision avoidance algorithm (after four iterations) is shown in Fig. 20. The new positions of the manipulator, where previously the collision with the gravitational buffer was, can be noticed.



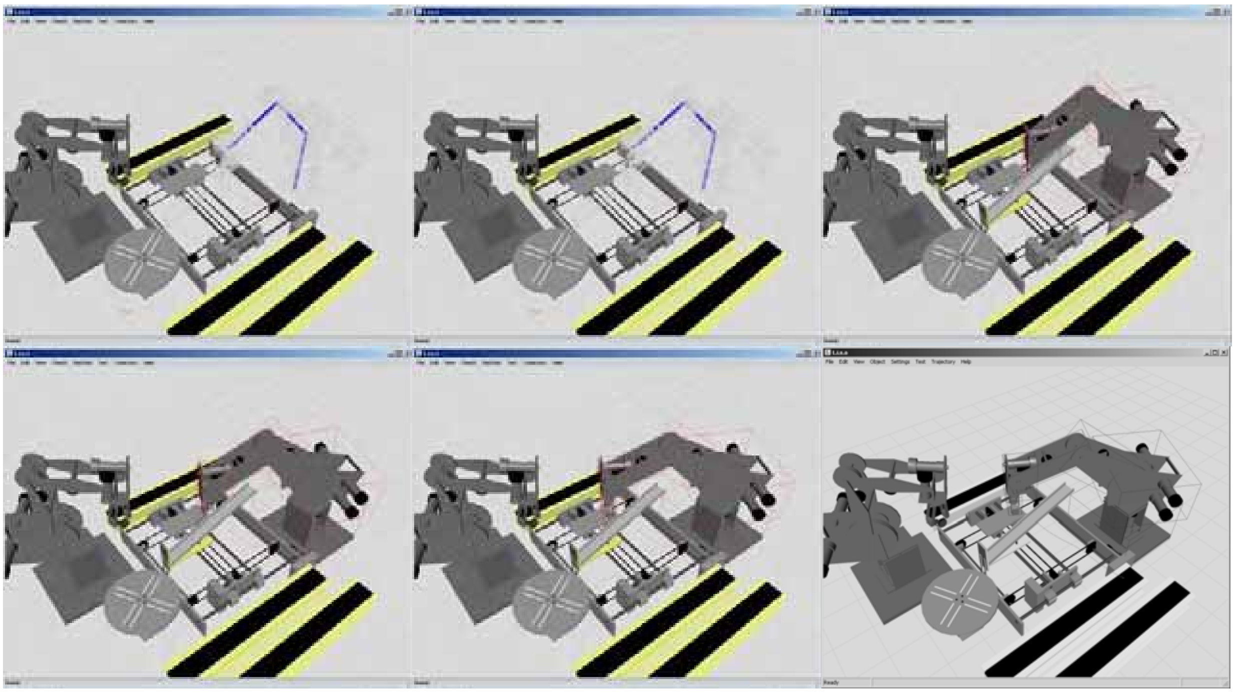
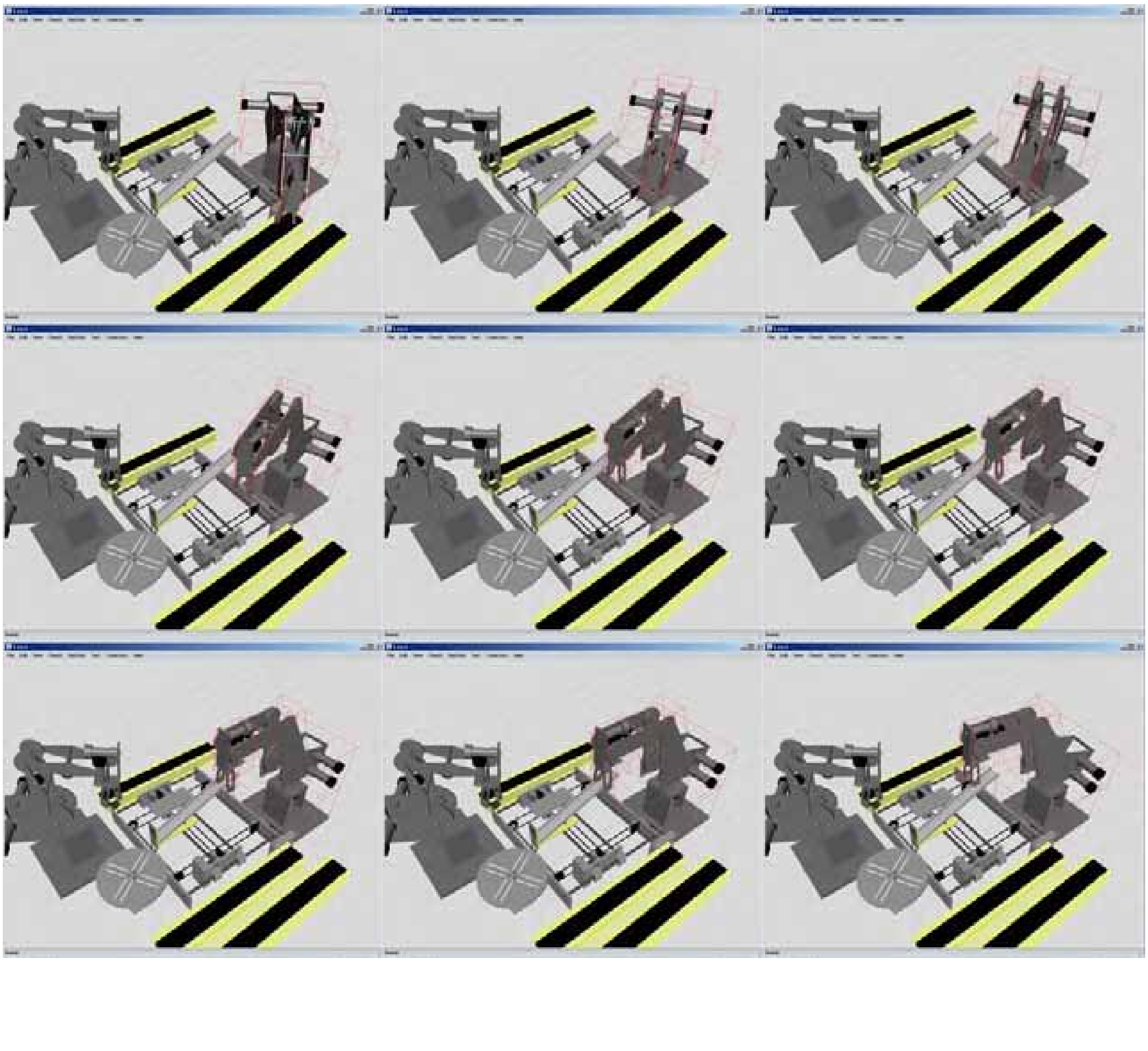


Figure 19. The starting trajectory



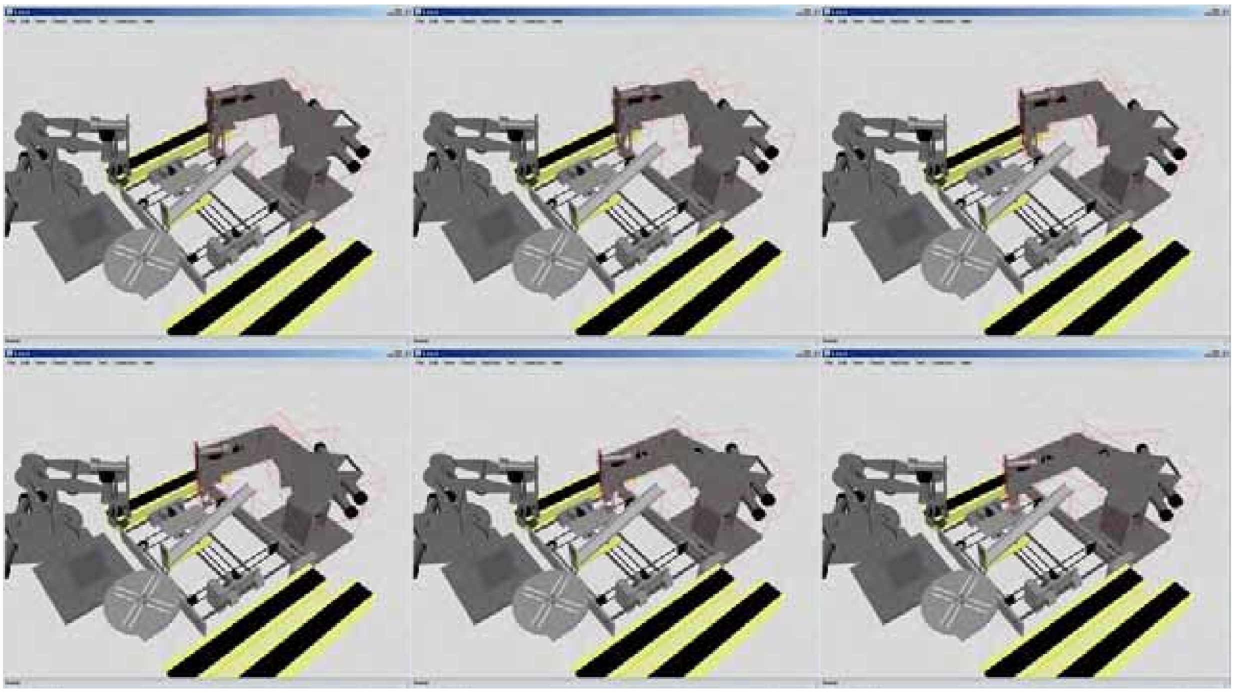
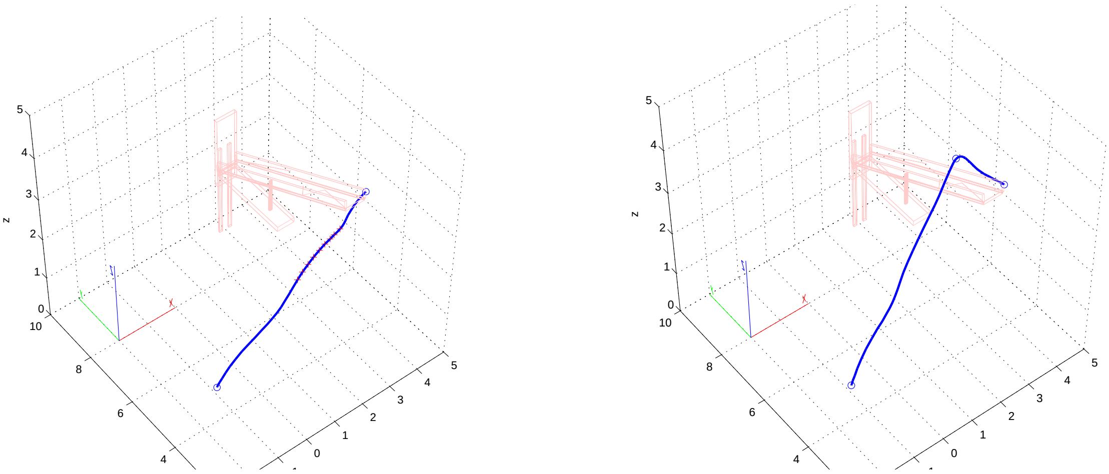


Figure 20. The final (collision-free) trajectory

7.2 Planning with an End-Effector Type-II

A collision avoidance in this simulation is done with a different end-effector (type-II) attached to the manipulator (see Fig. 12b). Because of a much simpler end-effector construction, only two iterations were needed for the collision avoidance trajectory planning, and only one additional way point was inserted into the evasion trajectory, producing a simpler collision avoidance trajectory then for other types of end-effectors (see Fig. 21). The complete evolution of trajectories is shown in Fig. 25b.



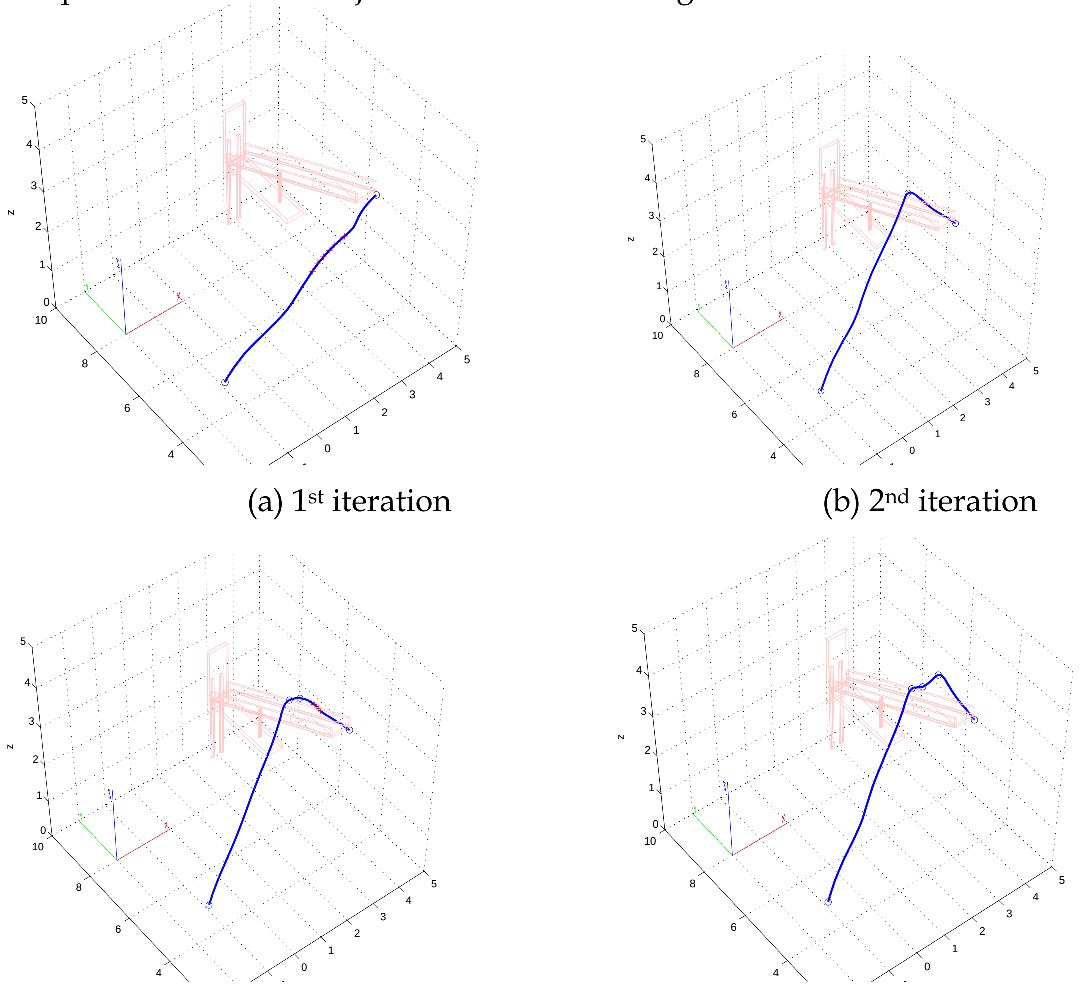
(a) 1st iteration

(b) 2nd iteration

Figure 21. End-effector positions throughout the iteration in collision-free trajectory planning

7.3 Planning with End-Effector Type-III

Simulation results of the collision avoidance of the RhinoXR-4 robot using the end-effector type-III (see Fig. 12c) are presented in this section. Four iterations were needed to evade successfully the collision object. i.e. the gravitational buffer. Three way points were inserted accordingly into the trajectory (see Fig. 22). On the account of geometrical structures similarity of the type-I and type-III end-effectors, their collision avoidance trajectories are very similar (see section 7.1), although with much "wider" trajectory around the gravitational buffer calculated in the case of the type-III end-effector. The complete evolution of trajectories is shown in Fig. 25c.



(a) 1st iteration

(b) 2nd iteration

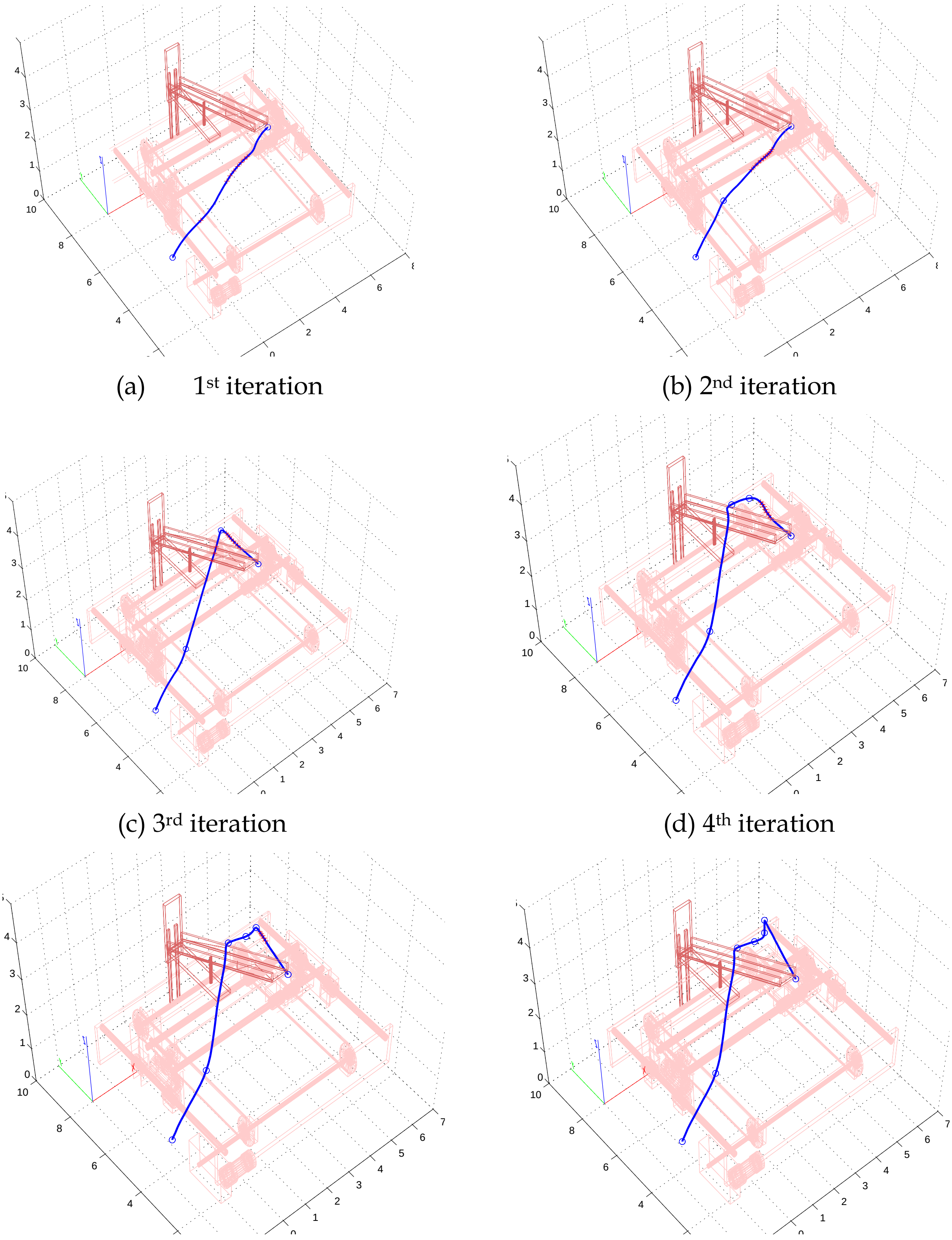
(c) 3rd iteration

(d) 4th iteration

Figure 22. End-effector positions throughout the iteration in collision-free trajectory planning

7.4 Planning with End-Effector Type-IV

The simulation results of collision avoidance of the RhinoXR4 robot using an end-effector type-IV are presented in this section. Since the end-effector type-IV is larger than previously considered end-effectors (see Fig. 12d), six iterations were needed to get a collision-free trajectory. In this simulation, as it can be observed in Fig. 23, the x-y table has become an additional obstacle, and first to be evaded by the manipulator. Even wider trajectories around the gravitational buffer were required for a successful evasion, because of the dimensions of the object held in the end-effector. The complete evolution of trajectories is shown in Fig. 25d.



(a) 1st iteration

(b) 2nd iteration

(c) 3rd iteration

(d) 4th iteration

(e) 5th iteration

(f) 6th iteration

Figure 23. End-effector positions throughout the iteration in collision-free trajectory planning

7.5 Comparison of Planning with Different End-Effectors

In the comparison of planning with different end-effectors, one can notice that the number of inserted trajectory points is increasing with the end-effectors complexity, from merely two inserted points in the case of the end-effector type-II to six inserted points in the case of the end-effector type-IV. The comparison of the evolution of trajectories in the R^3 space is presented in Fig. 24.

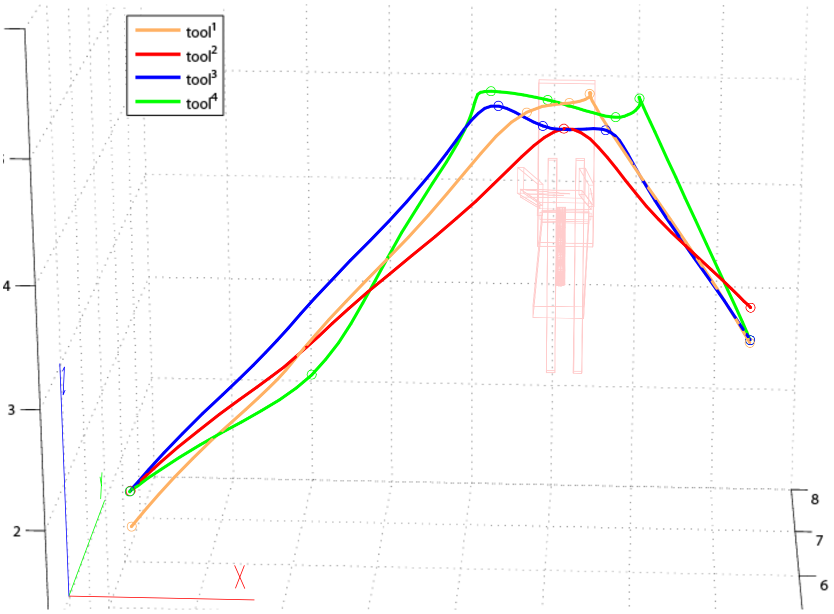
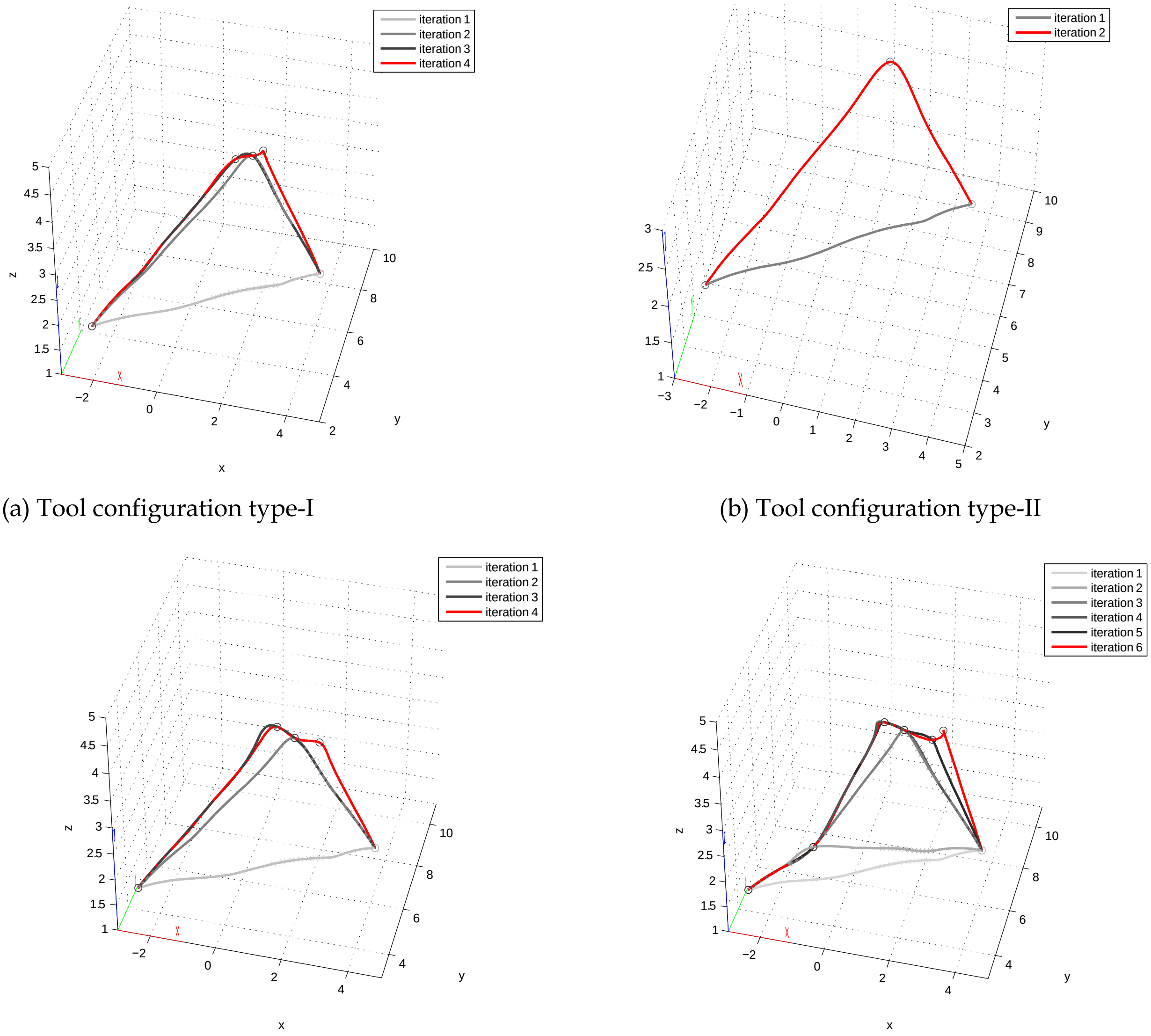


Figure 24. Comparison of trajectories in R^3 throughout the iterations



(a) Tool configuration type-I

(b) Tool configuration type-II

(c) Tool configuration type-III

(d) Tool configuration type-IV

Figure 25. Evolution of trajectories for different end-effectors

9. Conclusions

The algorithm for a collision-free trajectory generation is proposed and its implementation in on-line supervisory control of FMS is described. The algorithm makes the use of a collision detection engine applied in computer graphics and the change in a manipulator kinematics related to the virtual model of the FMS and exact position of the collision point. Virtual model surfaces are used as contact sensors to prevent objects from colliding. Any point of the robot's surface can be controlled and its trajectory planned. The exact 3D model is not a prerequisite for successful collision avoidance; however, with a better model more precise collision avoidance is possible. The algorithm is also very adaptable when planning trajectories for manipulators with different end-effectors or manipulators carrying various objects.

10. Future Work

Collision free trajectory generation presented in this paper is heuristic and suffers from a local minima problem, i.e. a manipulator can get stuck at certain positions when a direct planning from a start position to a goal position is made. An addition of sub-goals, as it was proposed in some different planning strategies, will help solve the problem. In addition, a general convergence of the iterative avoidance strategy should be proved. Environments with unknown objects will be monitored using a stereovision for mapping of objects, or their simplified OBB approximations, into a virtual space.

11. References

Barraquand, R. & Latombe, J.C. (1991). Robot Motion planning: A distributed Representation Approach, In: *Int. J. of Robotic Research*, Vol. 10 No. 6, (Dec. 1991) 628-649.

Barraquand, J.; Langlois, B. & Latombe, J. C. (1992). Numerical potential field techniques for robot path planning. In: *IEEE Transactions on Systems, Man, and Cybernetics*, (2):224-241, March-April 1992.

Cameron, S. & Qin, C. (1998). Motion Planning and Collision Avoidance with Complex Geometry, *Proceedings of Industrial Electronics Society, IECON '98.* pp. 2222-2226 vol.4., 0780345037, Aachen Germany, IEEE, USA.

Denavit, J. & Hartenberg (1955), R. S. A kinematic notation for lower-pair mechanisms based on matrices. In: *J. Appl. Mechanics*, June 1955, 22:215-221.

Jimenez, P.; Thomas, F. & Torras, C. (2001). 3D collision detection: a survey. *Computers and Graphics*, 25(2):269-285, April 2001.

Held, M. (1997) ERIT: A collection of efficient and reliable intersection tests. In: *Journal of Graphics Tools*: JGT, 2(4):25-44.

Ho, C.Y. & Cook, C.C. (1982). The application of spline functions to trajectory generation for computer controlled manipulators, In: *Digital Systems for Industrial Automation*, 1982. 1(4): 325-333. Crane, Rusak & Co., New York.

Hwang, Y.K. & Ahuja, N. (1992). A Potential Field Approach To Path Planning. IEEE Trans. on Robotics and Automation, Vol. 8, No. 1, 23-32, 1042-296X.

Gottschalk, S.; Lin, M. & Manocha, D. (1996). OBB-Tree: A Hierarchical Structure For Rapid Interference Detection; Proceedings of ACM SIGGRAPH, pp. 171-180, 0201948001, New Orleans, August 1996, Addison-Wesley.

Gottschalk, S. (2000) Collision queries using oriented bounding boxes. PhD thesis, University of North Carolina at Chapel Hill.

Kheddar, A.; Coquillart, S. & Redon S. (2002). Hierarchical backface culling for collision detection, *Available from:* http://www-rocq.inria.fr/~redon/papers/ /iros2002.ps.

Khalil, W. & Kleinfinger, J. F. (1986). A new geometric notation for open and closed-loop robots. In IEEE International Conference on Robotics and Automation, San Francisco, April 7-10, 1986, pages 1174-1179.

Klosowski, J.T (1998). Efficient collision detection for interactive 3D graphics and virtual environments. PhD thesis, State University of New York at Stony Brook.

Lin, M. & Gottschalk, S. (1998). Collision Detection between Geometric Models: A Survey, *Proceedings of IMA Conference on Mathematics of Surfaces*, Cripps R.(Ed.), 1874728151, Birmingham. Winchester. Information Geometers.

Lozano-Perez, T. (1986). A simple motion planning algorithm for general manipulators. In: *IEEE J. of Robotics and Automation*. Also MIT AI Memo 896, June 1986, RA-3(3):224-238.

Möller, T. (1997). A fast triangle-triangle intersection test. *Journal of Graphics Tools: JGT*, 2(2):25-30, 1997.

Möller, T. & Rundberg, P. (1999). Caching for collision detection, June 08, 1999. *Available from:* http://www.ce.chalmers.se/staff/biff/exjobb/collision.ps.gz.

Paul, R.P. (1981). *Robot Manipulators: Mathematics, Programming and Control.* MIT Series in Artificial Intelligence. The MIT Press.

Reichenbach, T. & Kovačić, Z. (2003). Collision Avoidance In Virtual Environment Based On Usage Of Kinematics Derived From A 3d Robot Model, *Proceedings of MED2003 CD*, DNBI 85/-766/5-0-/, Rhodes, June 2003.

Sheth, P.N. & Uicker, J.J. (1971). A generalized symbolic notation for mechanisms. *Journal of Engineering for Industry*, 93, pages 102-112.

Taylor, R.H. (1979). Planning and execution of straight line manipulator trajectories. In: *IBM Journal of Research and Development*, 23(4):424-436, July 1979.

-X-

Mechatronics

Exploring Open-Ended Design Space of Mechatronic Systems

Zhun Fan, Jiachuan Wang & Erik Goodman

1. Introduction

In general, design of mechatronic systems includes two steps: conceptual design and detailed design. In the conceptual design phase, the following questions should be answered (Tay et al., 1998): 1) what is the exact design problem to be solved? (This requires a complete and consistent listing of the requirements), and 2) what are the key problem areas in the solution? (This requires the identification of critical parts of the solution that will determine the performance).

Then the process of detailed design can be undertaken, identifying those candidate solutions that meet the requirements and provide the level of performance needed. The research in this paper focuses on the detailed design of mechatronic systems. The strategy is to develop an automated procedure capable of exploring the search space of candidate mechatronic systems and providing design variants that meet desired design specifications or dynamical characteristics.

The method must be able to explore the design space in a topologically open-ended manner, yet still find appropriate configurations efficiently enough to be useful.

Much research has been done on design automation of single domain systems using evolutionary computation approach.

For example, automated design of analog circuits has attracted much attention in recent years (Grimbleby, 1995) (Lohn, 1999)(Koza, 1999)(Fan, 2000). They could be classified into two categories: GA-based and GP-based. Most GA-based approaches realize topology optimization via a GA and parameter optimization with numerical optimization methods (Grimbleby, 1995).

Some GA approaches also evolve both topology and component parameters; however, they typically allow only a limited amount of components to be evolved (Lohn, 1999). Although their work basically achieve good results in analog circuit design, they are not easily extendable to interdisciplinary systems like mechatronic systems.

Design of interdisciplinary (multi-domain) dynamic engineering systems, such as mechatronic systems, differs from design of single-domain systems, such as electronic circuits, mechanisms, and fluid power systems, in part because of the need to integrate the several distinct domain characteristics in predicting system behavior (Coelingh. et al., 1998).

However, most current modeling and simulation tools that provide for representation at a schematic, or topological, level have been optimized for a single domain. The bond graph provides a unified model representation across inter-disciplinary system domains. Tay uses bond graphs and GA to generate and analyze dynamic system designs automatically (Tay et al., 1998). He uses nested GA to evolve both topology and parameters for dynamic systems. However, the efficiency of his approach is hampered by the weak ability of GA to search in both topology and parameter spaces simultaneously.

Genetic programming is an effective way to generate design candidates in an open-ended, but statistically structured, manner. There have been a number of research efforts aimed at exploring the combination of genetic programming with physical modeling to find good engineering designs.

Perhaps most notable is the work of Koza et al.. He presents a single uniform approach using genetic programming for the automatic synthesis of both the topology and sizing of a suite of various prototypical analog circuits, including low-pass filters, operational amplifiers, and controllers.
This approach appears to be very promising, having produced a number of patentable designs for useful artefacts. It is closely related to our approach, except that it searches in a single energy domain.

To develop an integrated mechatronic design environment, we investigate an approach combining genetic programming and bond graphs to automate the process of design of mechatronic systems to a significant degree.
To improve the topology search capability of GP and enable it to provide a diversity of choices to the designer, a special form of parallel GP, the Hierarchical Fair Competition GP (HFC-GP), is used in this paper (Hu, et al., 2002).

The efficiency and effectiveness of the approach are illustrated in an interesting redesign example involving the drive mechanism for an electric typewriter. Several design alternatives for the typewriter drive are derived through exploring open-topologies in bond graph space.

2. Design Domain and Methodology

2.1 Mechatronic Systems and Bond Graph Representations

The reason we used bond graphs in research on mechatronic system synthesis is because mechatronic systems are intrinsically multi-domain systems. We need a uniform representation of mechatronic systems so that designers can not only shift among different hierarchies of design abstractions but also can move around design partitions in different physical domains without difficulty.

The bond graph is a modeling tool that provides a unified approach to the modeling and analysis of dynamic systems, especially hybrid multi-domain systems including mechanical, electrical, pneumatic, hydraulic components, etc. It is the explicit representation of model topology that makes the bond graph a good candidate for use in open-ended design search.

Fig. 1. shows an example of a single bond graph model that represents a resonator unit in both mechanical domain and electrical domain.

In addition to appropriate "drivers" (sources), the left part of Fig. 1. shows a lumped-parameter dynamical mechanical system model typically including a mass, spring and damper while the right part of Fig.1. shows an "RLC" electric circuit including a resistor, inductor and capacitor. However, they could both be expressed in the same bond graph shown in the middle of Fig.1 because both the mechanical and electrical subsystem share the same dynamic behavior and governed by the same dynamic equations.

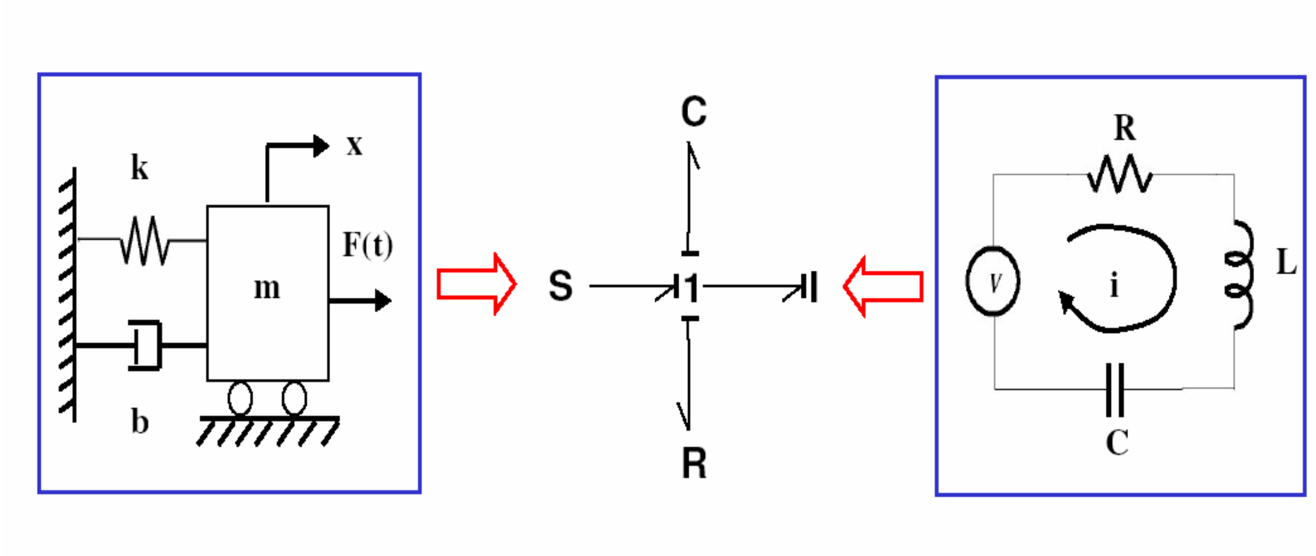


Figure 1. Bond graph representation of dynamic systems

It is also very natural to use bond graphs to represent a dynamic system, such as a mechatronic system, with cross-disciplinary physical domains and even controller subsystems (Fig. 2.)

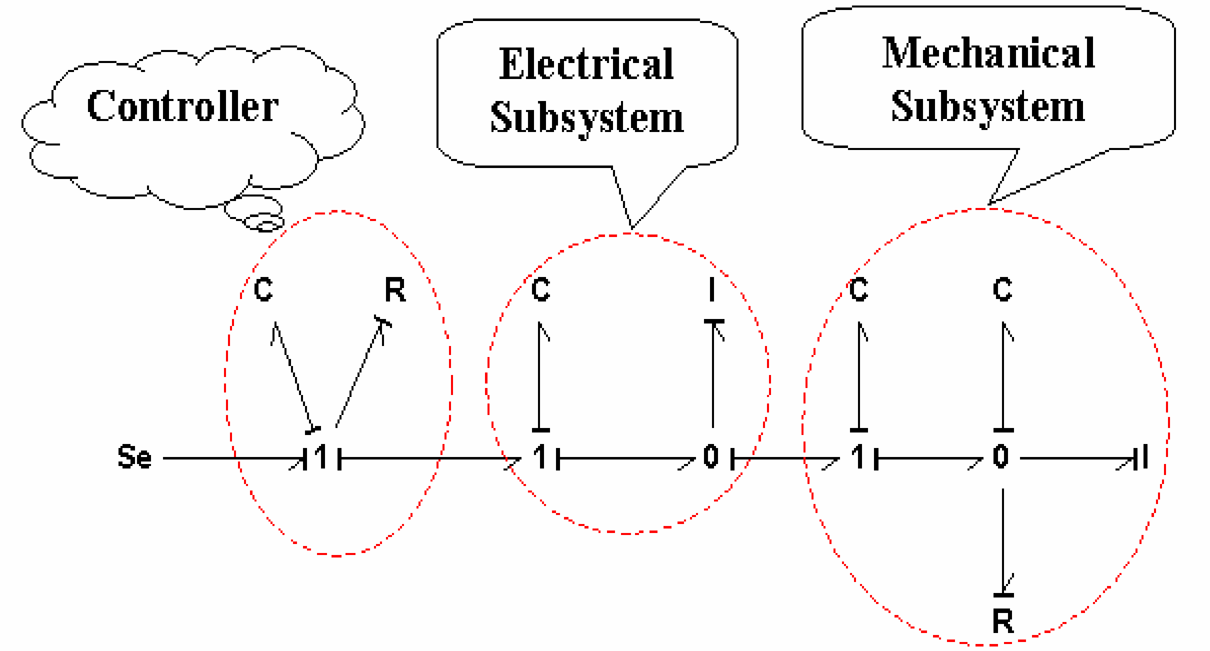


Figure 2. Bond graph representing a mechatronic system with mixed energy domains and a controller subsystem

2.2 Bond Graph

The bond graph is a modeling tool that provides a unified approach to the modeling and analysis of dynamic systems, especially hybrid multi-domain systems including mechanical, electrical, pneumatic, hydraulic, etc. (Karnopp et al., 2000). It is the explicit representation of model topology that makes the bond graph a good candidate for use in open-ended design searching.

For notation details and methods of system analysis related to the bond graph representation see Karnopp et al. and Rosenberg (Rosenberg et al., 1992). Much recent research has explored the bond graph as a tool for design (Sharpe & Bracewell, 1995, Tay et al., 1998, Youcef-Toumi, 1999, Redfield R., 1999).

In our research, the bond graph has additional desirable characteristics for selection as the tool for system representation and simulation. The evaluation efficiency of the bond graph model can be improved because analysis of causal relationships and power flow between elements and subsystems can be done quickly and easily, and reveals certain important system properties and inherent characteristics.

This makes it possible to discard infeasible design candidates even before numerically evaluating them, thus reducing time of evaluation to a large degree. Because virtually all of the circuit topologies passing causal analysis are simulatable, our system does not need to check validity conditions of individual circuits to avoid singular situations that could interrupt the running of a program evaluating them.

Another characteristic of bond graphs is their ease of mapping to the engineering design process (Xia et al., 1991). Because each component of the system can be represented correspondingly in a bond graphs, junctions and elements can be added to or deleted from a model without causing dramatic changes.

This emulates the engineering process of modifying systems, refining simple designs discovered initially, adding size and complexity as needed to meet more complicated design demands step by step. As genetic programming usually shows a weak causality of structure evolution (Rosca, 1995), this potential strong causality of the bond graph modification process also makes bond graph representation an attractive technique to use in genetic programming to explore the open-ended mechatronic system design space in an evolutionary process.

2.3 Combining Genetic Programming and Bond Graph

The most common form of genetic programming (Koza, 1994) uses trees to represent the entities to be evolved. The tree representation on GP chromosomes, as compared with the string representation typically used in GA, gives GP more flexibility to encode solution representations for many real-world design applications.

The bond graph, which can contain cycles, is not represented directly on the GP tree—instead, the function set (nodes of the tree) encode a constructor for a bond graph. We define the GP functions and terminals for bond graph construction as follows.

There are four types of functions:

- first, add functions that can be applied only to a junction and which add a C, I, or R element; -
- second, insert functions that can be applied to a bond and which insert a 0-junction or 1-junction into the bond;
- third, replace functions that can be applied to a node and which can change the type of element and corresponding parameter values for C, I, or R elements; and
- fourth, arithmetic functions that perform arithmetic operations and can be used to determine the numerical values associated with components (Table 1). Details of function definitions are illustrated in Seo et al. (2001).

Name	Description
add_C	Add a C element to junctions
add_I	Add an I element to junctions
add_R	Add an R element to junctions
insert_J0	Insert a 0-junction in bond
insert_J1	Insert a 1-junction in bond
replace_C	Replace current element with C element
replace_ I	Replace current element with I element
replace_ R	Replace current element with R element
+	Add two ERCs
-	Subtract two ERCs
endn	End terminal for add element operation
endb	End terminal for insert junction operation
endr	End terminal for replace element operation
erc	Ephemeral random constant (ERC)

Table 1. Function and terminal set for bond graph evolution

Defining a proper function set is one of the most significant steps in using genetic programming. It may affect both the search efficiency and validity of evolved results and is closely related to the selection of building blocks for the system being designed. In this work, the genotypes assembled from the function sets are constructors which, upon execution, specify a bond graph.

In other words, when the genotype is executed, it generates the phenotype in a developmental manner. In this research, we have an additional dimension of flexibility in generating phenotypes, because bond graphs are used as modeling representations for multi-domain systems, serving as an intermediate representation between the mapping of genotype and phenotype, and those bond graphs can be interpreted as systems in different physical domains, chosen as appropriate to the circumstances.

Fig. 3 gives a particular example in the domain of electrical circuits and Fig. 4 illustrates the role of bond graphs in the mappings from genotypes to phenotypes (Fan et al., 2001)

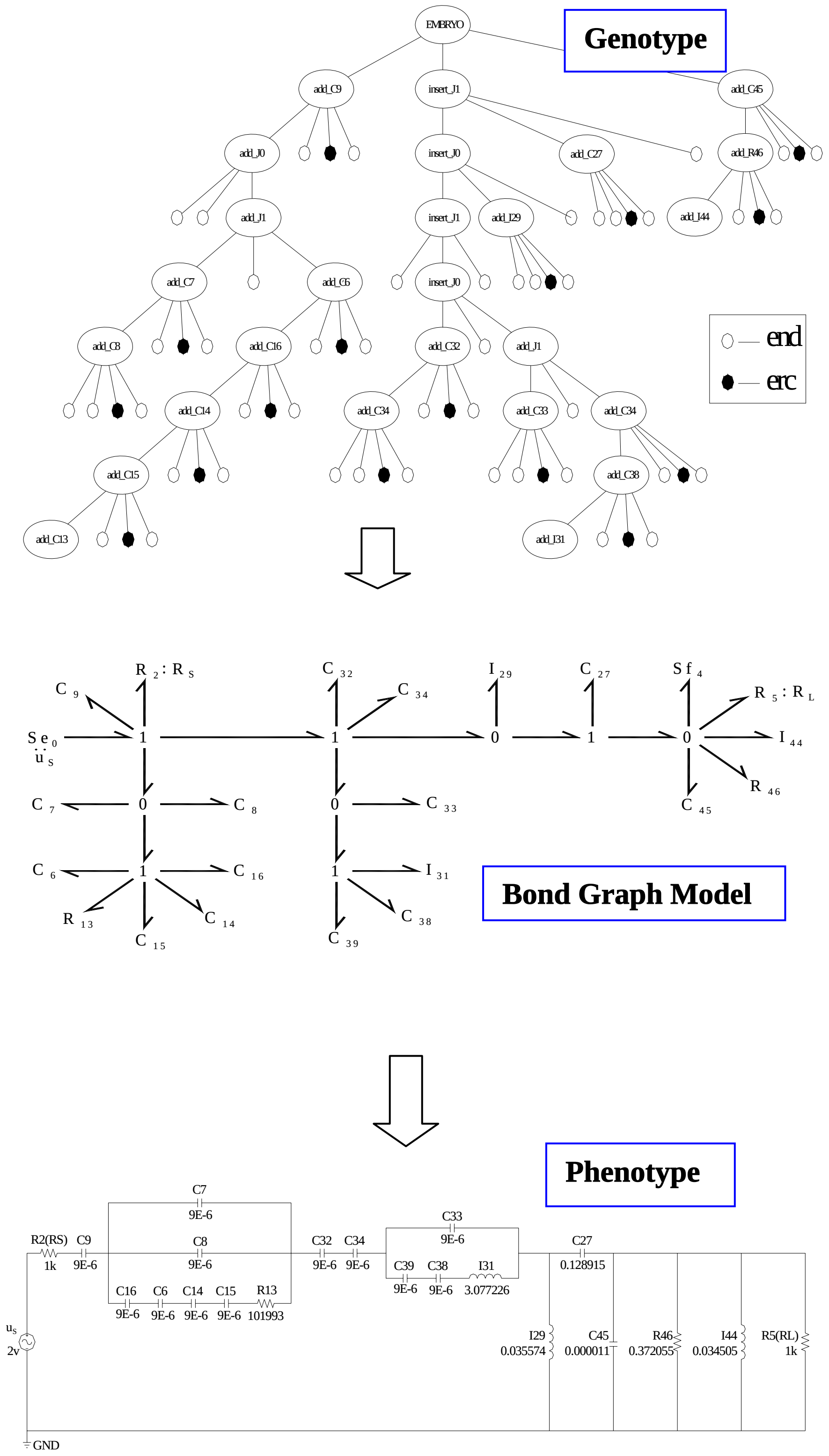


Figure 3. Example of Genotype-Phenotype Mapping in the Electrical Circuit Domain

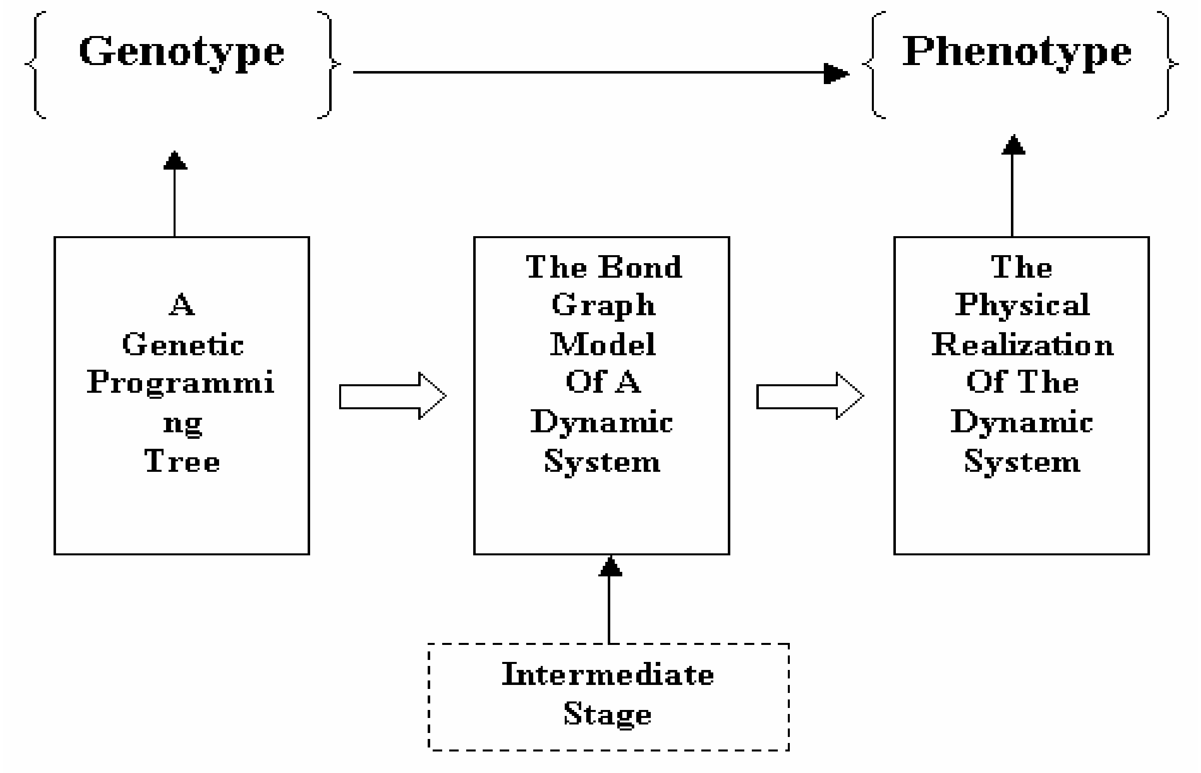

Genotype
Phenotype
A Genetic Programmi ng Tree
The Bond Graph Model Of A Dynamic System
The Physical Realization Of The Dynamic System
Intermediate Stage

Figure 4. Genotype-Phenotype mapping

2.4 Design Procedure

The flow of the entire algorithm is shown in Fig. 5. Based on a preliminary analysis, the user specifies the embryonic physical model for the target system (i.e., its interface to the external world, in terms of which the desired performance is specified) After that, an initial population of GP trees is randomly generated. Each GP tree maps to a bond graph tree.

Analysis is then performed on each bond graph tree. This analysis consists of two steps – causal analysis and state equation analysis. After the (vector) state equation is obtained, the important dynamic characteristics of the system are sent to the fitness evaluation module and the fitness of each tree is evaluated.

For each evaluated and sorted population, genetic operations – selection, crossover, mutation and reproduction – are carried out to seek design candidates with improved quality. The loop of bond graph analysis and GP operation is iterated until a termination condition is satisfied or specified number of iterations is performed. The final step is to instantiate a physical design, replacing the bond graphs with physical components it represents.

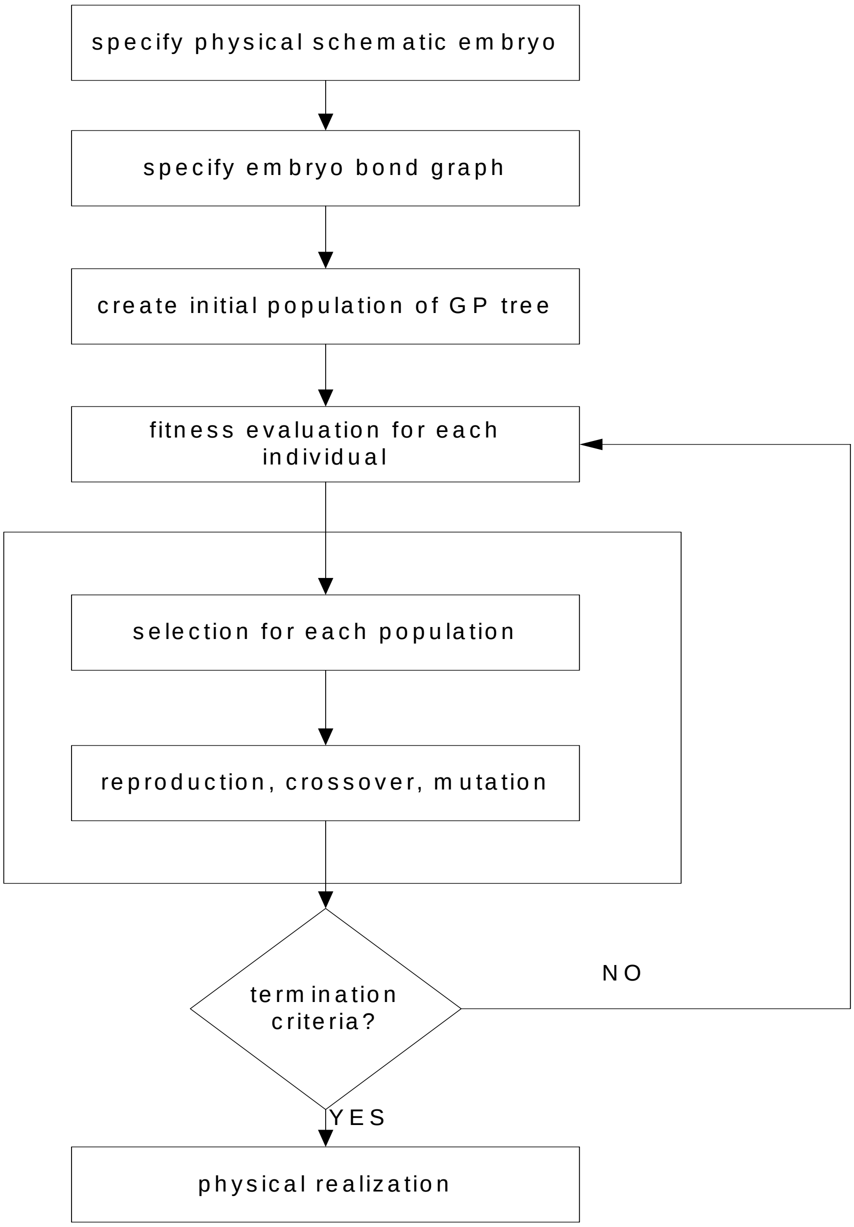

specify physical schematic embryo
specify embryo bond graph
create initial population of GP tree
fitness evaluation for each individual
selection for each population
reproduction, crossover, mutation
termination criteria?
NO
YES
physical realization

Figure 5. Flow chart of the design procedure

2.5 Integrated Evolutionary Mechatronics Synthesis

Because the final target of this research is to improve the quality of strategic and early design decisions, enhance human-computer cooperation, and ultimately reduce product and overall system life cycle cost, an integrated design and synthesis framework for mechatronic systems is presented and to be investigated, to cover a full spectrum of customer needs and product life considerations.

Fig. 6. provides a graphical overview of the integrated design environment (Wang, 2004). Evolutionary computation techniques, including genetic programming, are utilized to explore the open-ended design space for mechatronic systems as the core high-performance computing algorithm. Bond Graphs are used to unify representations of mechatronic subsystems from different domains. Design performances can be evaluated both through time domain simulation and via frequency domain analysis.

The design primitives at both conceptual and physical realization levels are stored in the design repository so that designers can retrieve them either manually or through classifying schemes. The Graphical User Interface (GUI) is designed to better understand customer needs, through the specification of design representation, requirements and constraints interactively. It can also incorporate user preferences under different trade-off strategies. The process of synthesis and analysis are iterative until design process converges to satisfactory design solutions.

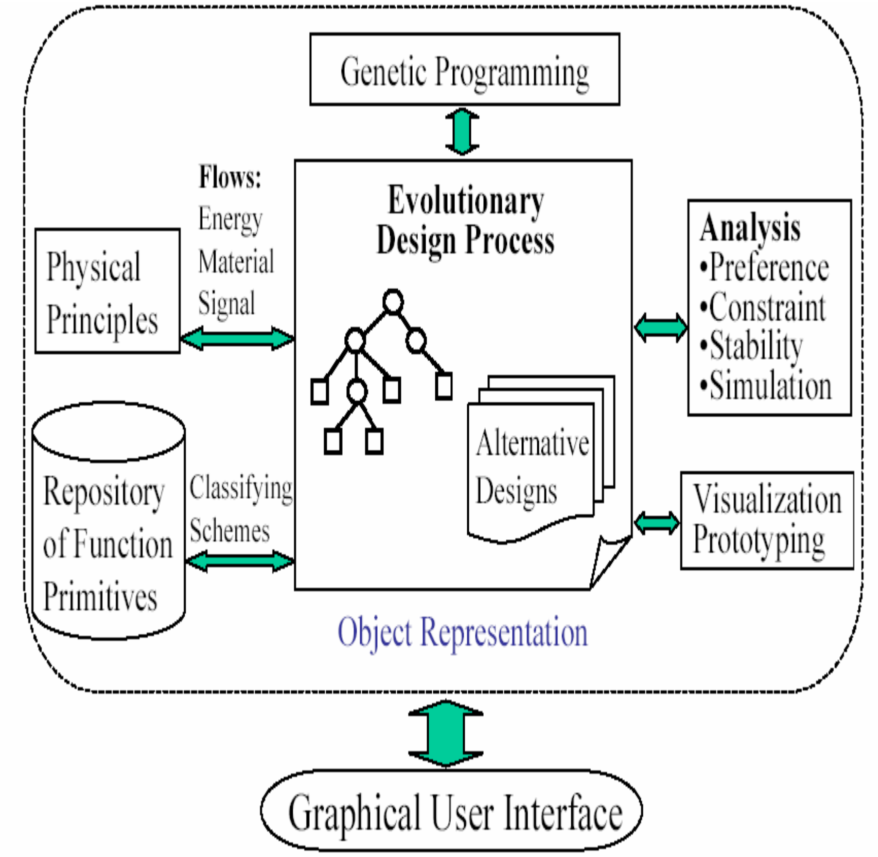


Figure 6. Integrated mechatronics design environment

3. Case Study

3.1 Problem Formulation

The original problem was presented by C. Denny and W. Oates of IBM, Lexington, KY, in 1972. Fig. 7. shows a closed-loop control system to position a rotational load (inertia) denoted as JL.

The system includes electric voltage source, motor and mechanical parts. As it is a multi-domain mechatronic system, a bond graph is convenient to use for modelling (see Fig. 8a). The problem with the design is the position output of the load JL has intense vibrations (see Fig. 8b).

The design specification is to reduce the vibration of the load to an acceptable level, given certain command conditions for rotational position. We want the settling time to be less than 70ms when the input voltage is stepped from zero to one. Note that the settling time of the original system is about 2000ms. The time scale in Fig. 8b is 4000 ms.

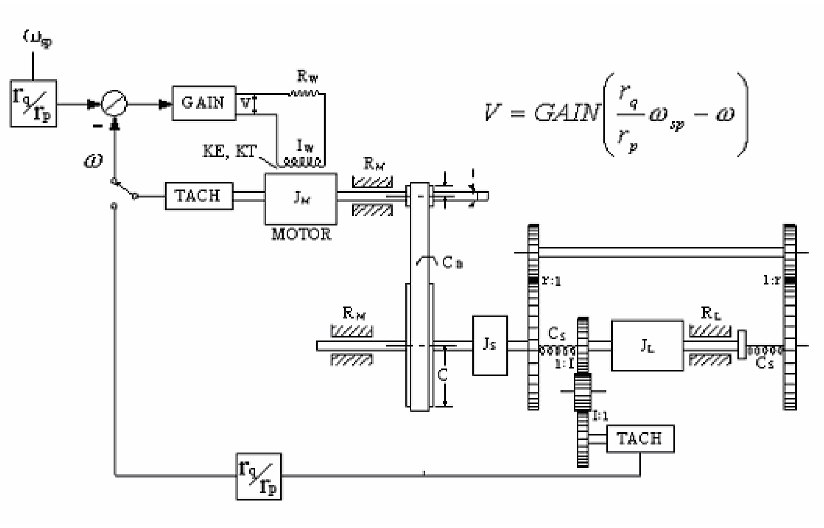
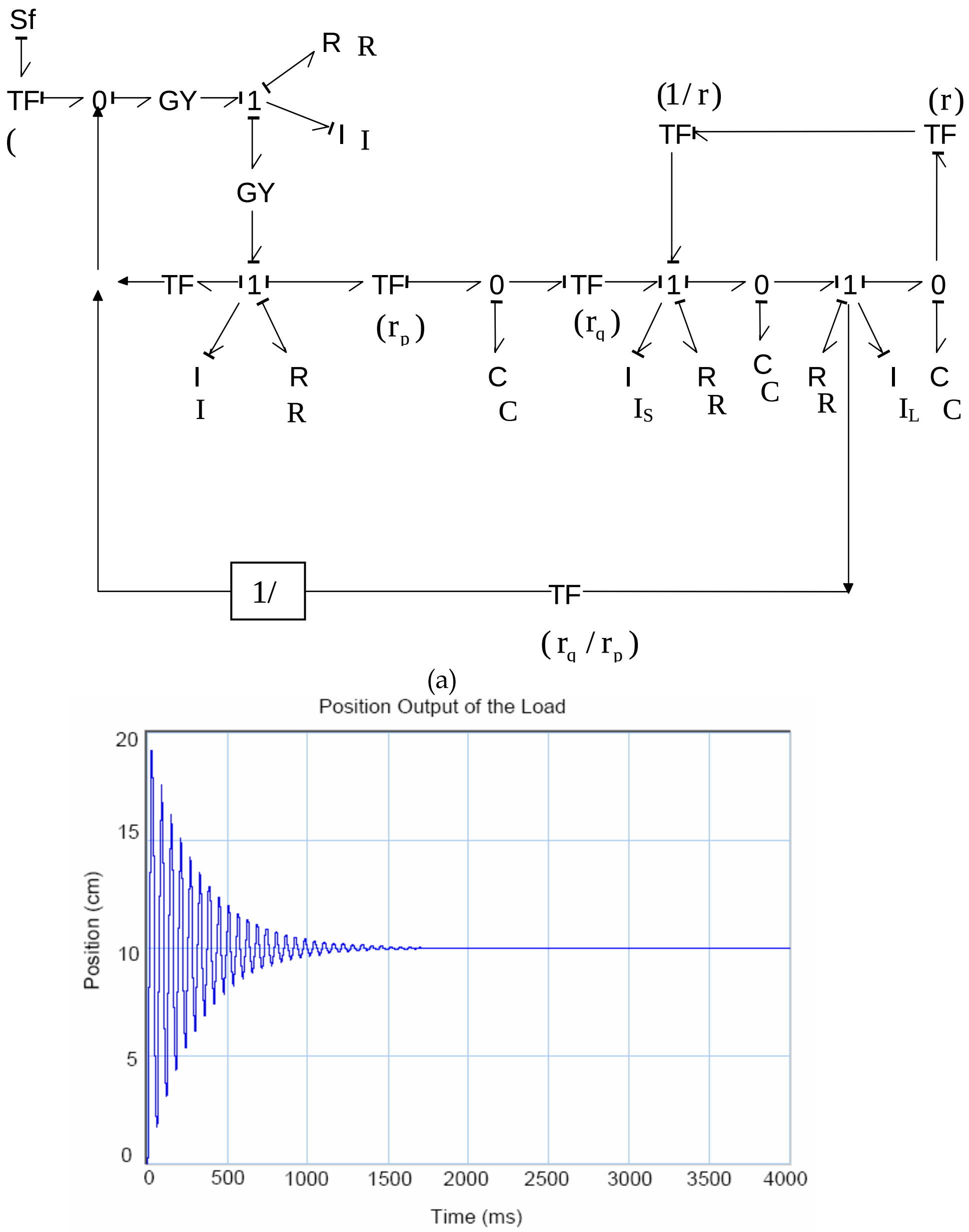


Figure 7. Schematic of the typewriter drive system



(a)

(b)

Figure 8. a) Bond graph model b) Positional vibration of the load

3.2 Embryo of Design

By analysing the model, we conclude that the critical part for the design is a subsystem that involves the drive shaft and the load (see Fig. 9). The input is the driving torque, Td, generated through the belt coupling back to the motor (not shown).

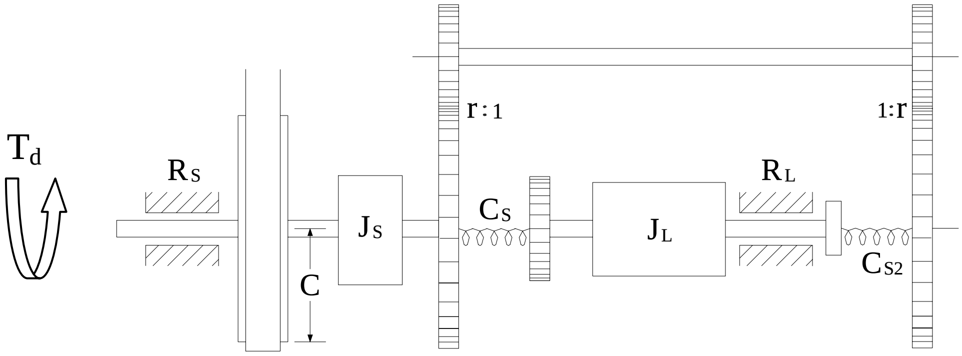


Figure 9. The embryo subsystem

This subsystem was deemed a logical place to begin the design problem. The questions left to the designer now are: 1) at which exact spots of the subsystem new components should be inserted, 2) which types of components and how many of them should be inserted, in which manner, and 3) what should be the values of the parameters for the components to be added? The approach reported in this paper is able to answer these three questions in one stroke in an automated manner, once the embryo system has been defined.

To search for a new design using the BG/GP design tool, an embryo model is required. The embryo model is the fixed part of the system and the starting point for GP to generate candidates of system designs by adding new components in a developmental manner. The embryo used for this example, expressed in bond graph language, is shown in Fig. 10, with the modifiable sites highlighted. The modifiable sites are places that new components can be added. The choice of modifiable sites is typically easy for the designer to decide. Note that modifiable sites are only possible spots for insertion of new components; the search may not use all of them. In this particular example, designers need have no idea whether assemblies of new components will be inserted at modifiable site (1), or at modifiable site (2), at site (3), or at any combinations of them. Instead, the algorithm will answer these questions in an automatic way, without intervention by the human designer.

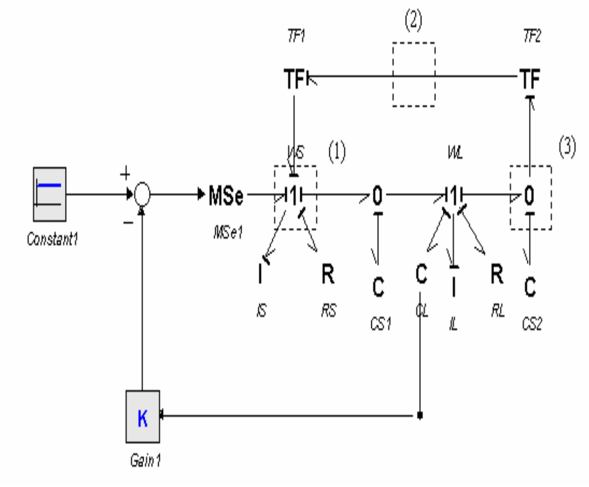


Figure 10. The embryo subsystem

The parameters for the embryo model are:

$$I_s : 6.7 \times 10^{-6}\ \mathrm{kg} \cdot \mathrm{m}^2$$

$$R_s : 0.013 \times 10^{-3}\ \mathrm{N} \cdot \mathrm{m} \cdot \mathrm{sec} \,/\, \mathrm{rad}$$

$$C_{s1} : 0.208\ \mathrm{N} \cdot \mathrm{m} \cdot /\, \mathrm{rad}$$

$$C_{s2} : 0.208\ \mathrm{N} \cdot \mathrm{m} \cdot /\, \mathrm{rad}$$

$$R_L : 0.58 \times 10^{-3}\ \mathrm{N} \cdot \mathrm{m} \cdot \mathrm{sec} \,/\, \mathrm{rad}$$

$$I_L : 84.3 \times 10^{-6}\ \mathrm{kg} \cdot \mathrm{m}^2$$

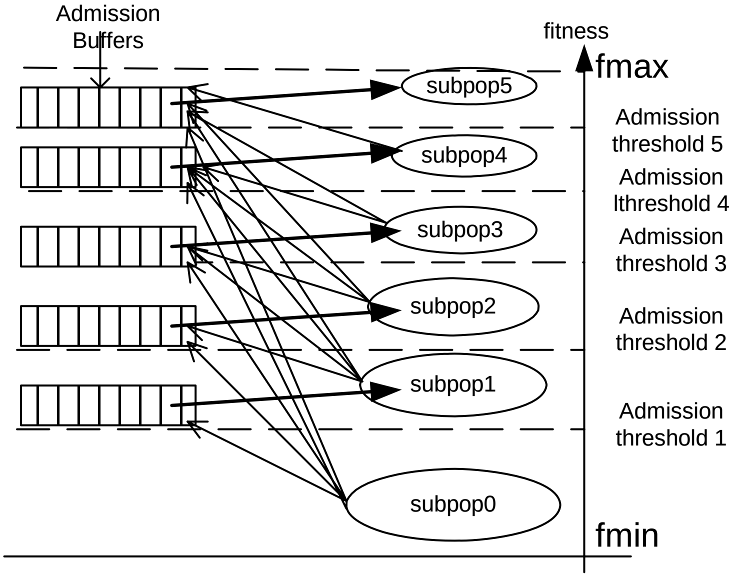
$$C_L : 1.0 \times 10^{6}\ \mathrm{N} \cdot \mathrm{m} \cdot /\, \mathrm{rad}$$

$$TF1 : 0.1, \qquad TF2 : 10$$

For simplicity and without loss of generality, both gains of K and MSe are set to be unit.

3.3 The Hierarchical Fair Competition (HFC) Model

A special form of genetic programming is applied in this research. In HFC (Hierarchical Fair Competing) model (Fig. 11), multiple subpopulations are organized in a hierarchy, in which each subpopulation can only accommodate individuals within a specified range of fitnesses (Hu et al., 2002). New individuals are created continuously in the bottom layer. Use of HFC model balances exploration and exploitation of GP effectively. Our experience shows that using the HFC model can also substantially increase the topology diversity of the whole population and help to provide the designer a diverse set of competing design candidates for further trade-offs.



In HFC model, subpopulations are organized in a hierarchy with ascending fitness level. Each subpopulation accomodates individuals within a certaiin fitness range determined by the admission thresholds

Figure 11. Hierarchical Fair Compete model of GP

3.4 Definition of Fitness Function

The fitness function of individual design is defined according to the position output response of the load JL as follows.
Within the time range of interest (0~500ms in this example), uniformly sample 1000 points of the output response (yielding a time interval between two adjacent sampling points of 0.5ms). Compare the magnitudes of the position output of the load JL at the sample points with target magnitudes (unity in this example), compute their difference and get a squared sum of difference as raw fitness, defined as $\mathrm{Fitness}_{\mathrm{raw}}$. Then normalized fitness is calculated according to:

$$\mathrm{Fitness}_{\mathrm{norm}} = 0.5 + 1000 \ / \left(2000 \ + \mathrm{Fitness}_{\mathrm{raw}} \right)$$

It can be assumed approximately that the higher the normal fitness, the better the design. Two reasons make the fitness definition an approximate one. 1) it does not reflect directly the strict definition of settling time, and 2) it does not include other considerations in design of the system except output response.
A modified fitness function could be defined later if required. However, in this research, the definition is enough to manifest the feasibility and efficiency of the approach reported. The achieved design results (Fig. 12-18) show performances satisfying the design specification presented in this research.

3.5 Experimental Setup

We used a strongly-typed version (Luke, 1997) of lilgp (Zongker & Punch, 1996) to generate bond graph models.

The major GP parameters were as shown below:

Number of generations: 100
Population sizes: 200 in each of 15 subpopulations
Initial population: half_and_half
Initial depth: 3-6 Max depth: 17
Selection: Tournament (size=7)
Crossover: 0.9
Mutation: 0.1

Three major code modules were created in our work. The algorithm kernel of HFC-GP was a modified version of an open software package developed in our research group -- lilgp. A bond graph class was implemented in C++. The fitness evaluation package is C++ code converted from Matlab code, with hand-coded functions used to interface with the other modules of the project. The commercial software package 20Sim was used to verify the dynamic characteristics of the evolved design.

3.6 Experimental Observations

The GP program obtains satisfactory results on a Pentium-IV 1GHz in 5~15 minutes, which shows the efficiency of our approach in finding good design candidates.

Ten runs of this problem have been done and most of the runs produced satisfactory solutions.

The fitness history of a typical run is shown in Fig. 12. Two competing design candidates with different topologies, as well as their performances, are provided in Fig. 13 to Fig. 18 (evolved components are circled). We can see from the output rotational position responses that they all satisfy the design specification of settling time less than 70ms. Note that the time scale of the plots is 100 ms.

One of the designs is shown in Fig. 13. It is generated in only 20 generations with 200 designs in each of 15 subpopulations, and has a very simple structure. Three elements, one each of 0-junction, C, and R, are added to modifiable site 1 of the embryo model (Fig. 13). Dashed circles highlight the newly evolved components in the bond graph figures. The performance of this model is shown in Fig. 14.

The position response for step function input quickly converges in about 50msec, which was an acceptable timeframe. One possible physical realization of the bond graphs model is shown in Fig. 13. A spring and a damper are added and coupled to the original printer subsystem as shown in Fig. 14. Another design is shown in Fig. 16.

Four elements, 0-junction with C, 1-junction with R are added to modifiable site 2 and one R is added to modifiable site 3. One possible physical realization of the design is shown in Fig. 17. Fig. 18 displays the performance of this model.Table 2 represents the statistical results of 10 runs for the printer drive.

It is clear that the approach reported in this research is both efficient and effective, capable of providing designers with a variety of design alternatives. This gives designers considerable flexibility to generate and to compare a large variety of design schemes.

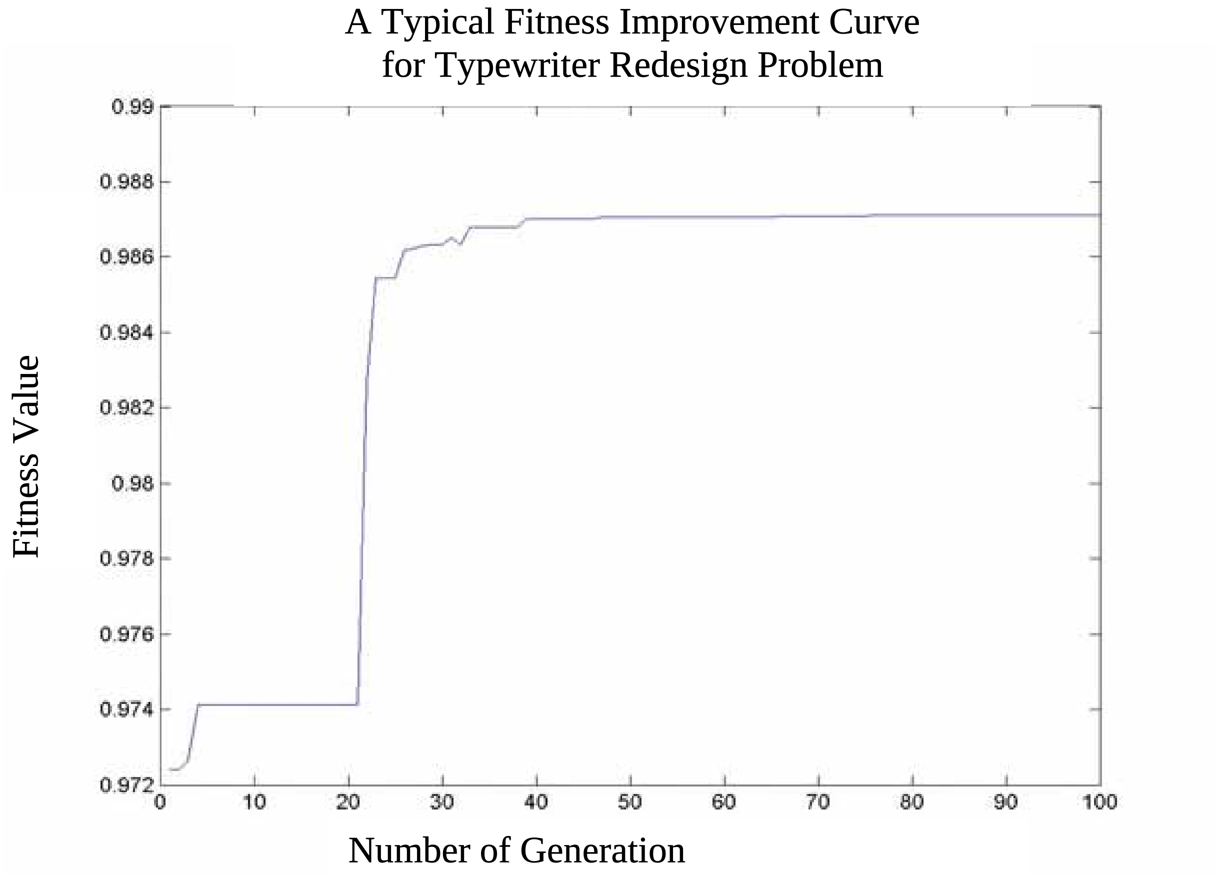
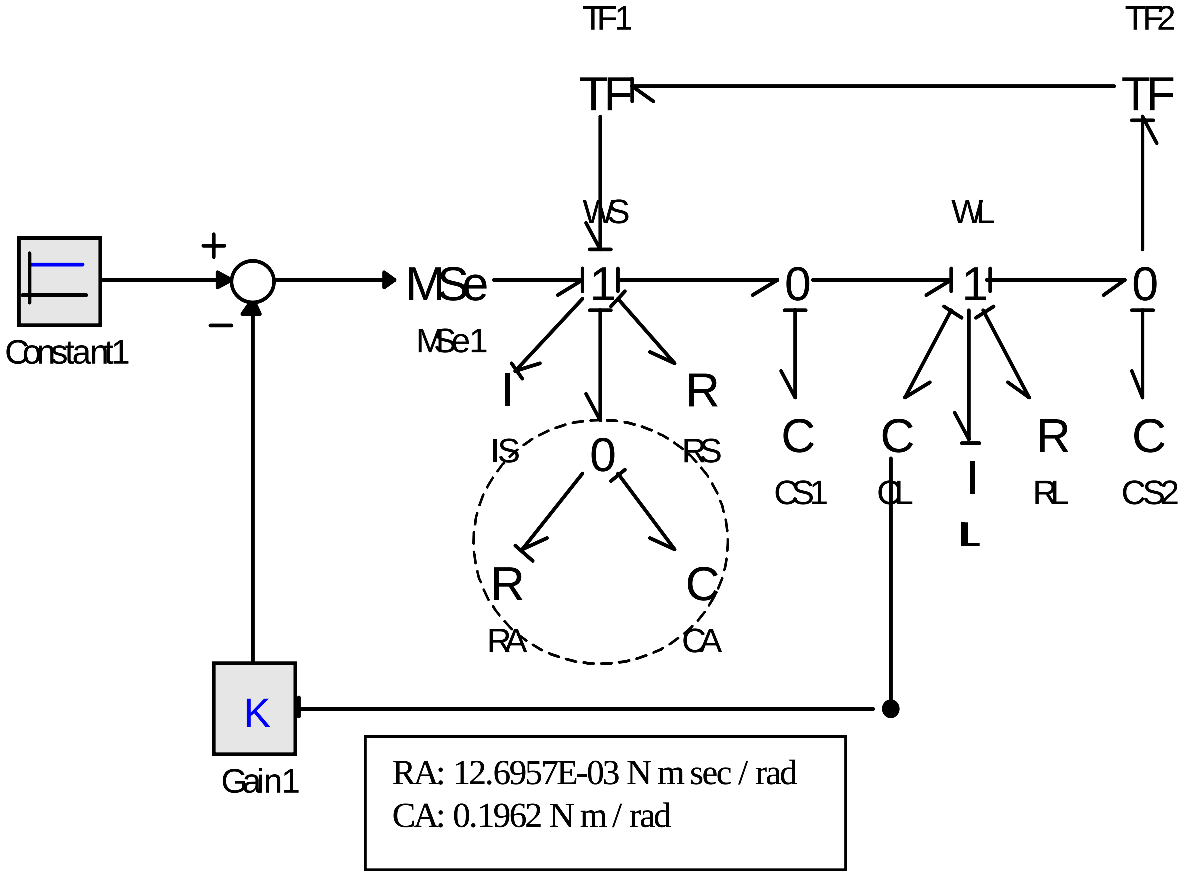


Figure 12. Fitness history for a typical typewriter drive redesign run



RA: 12.6957E-03 N m sec / rad
CA: 0.1962 N m / rad

Figure 13. The evolved bond graph model I

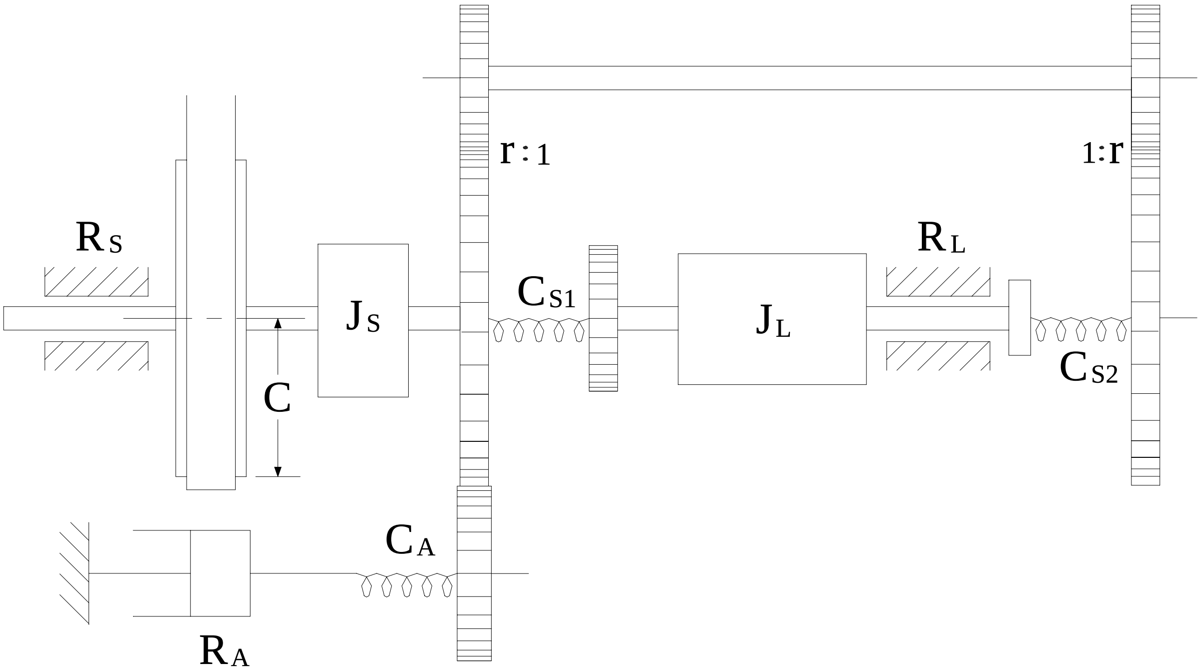


Figure 14. The physical realization of evolved bond graph model I

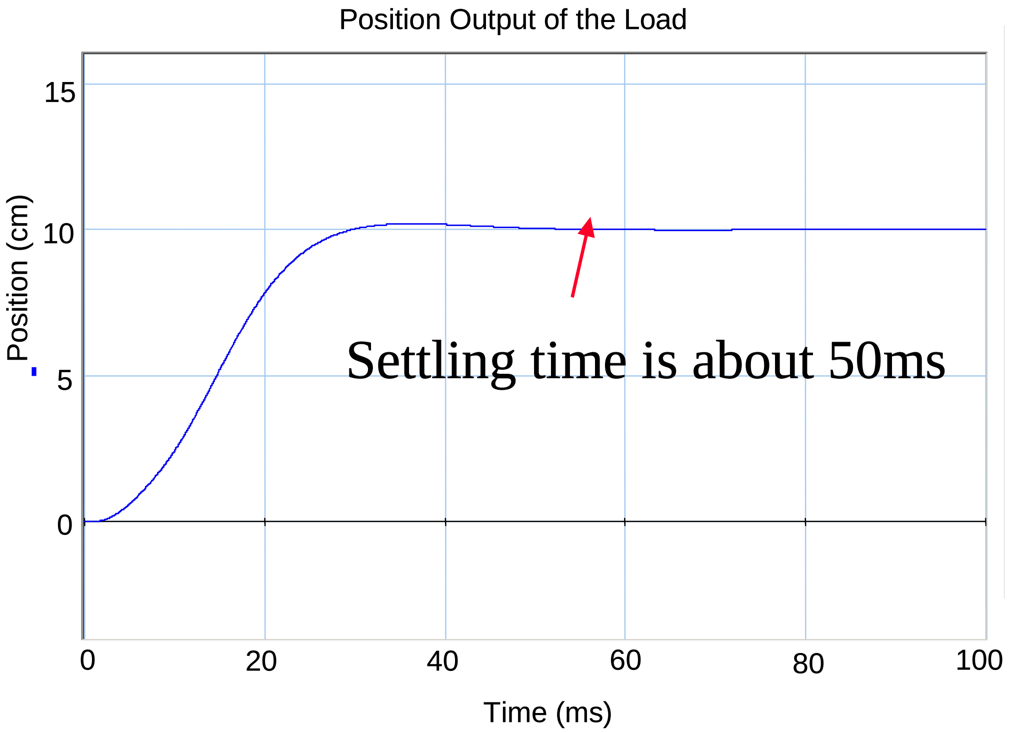
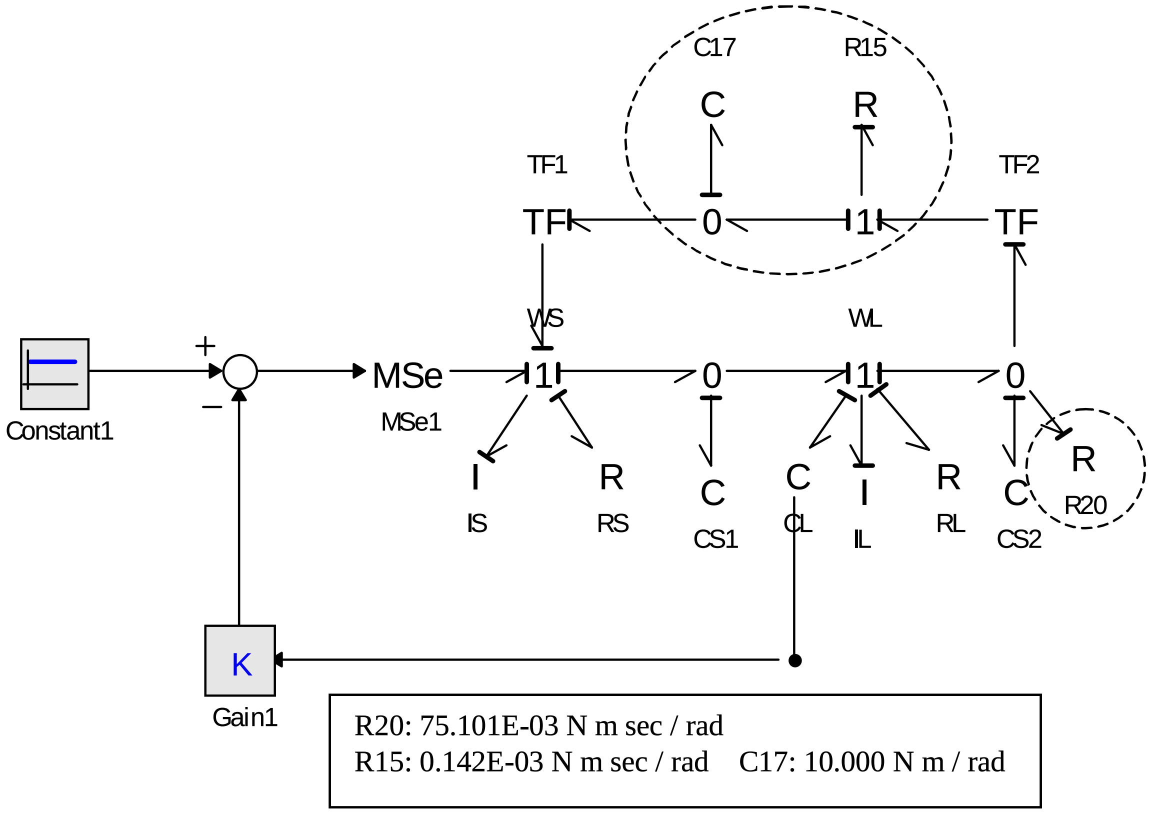


Figure 15. Simulation result of evolved bond graph model I



R20: 75.101E-03 N m sec / rad
R15: 0.142E-03 N m sec / rad   C17: 10.000 N m / rad

Figure 16. The evolved bond graph model II

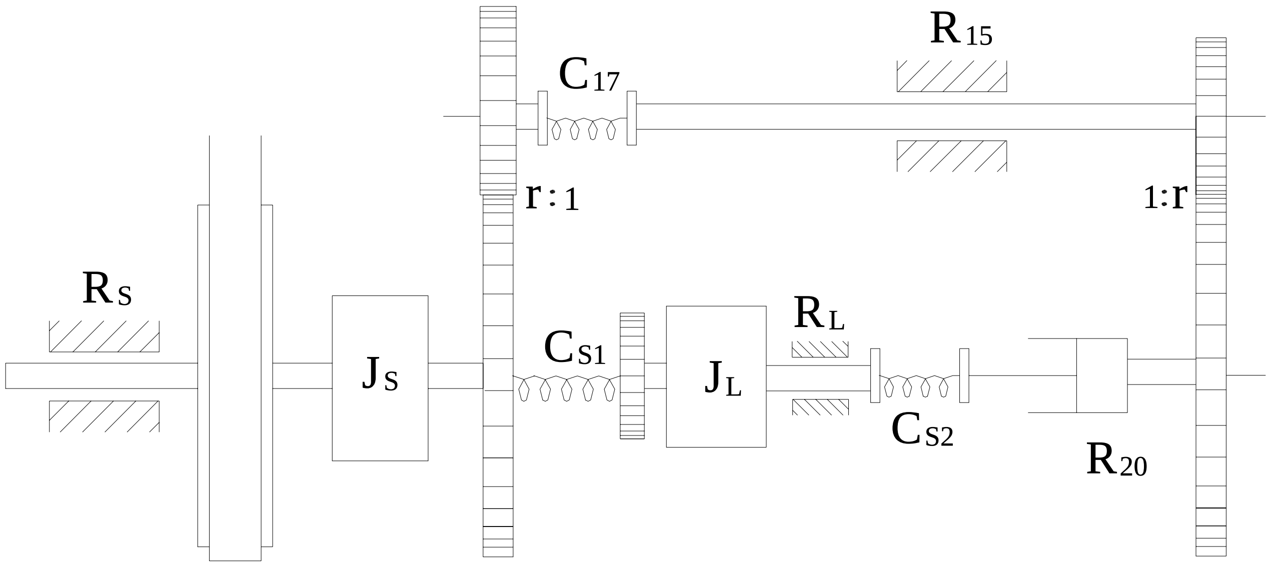


Figure 17. The physical realization of evolved bond graph model II

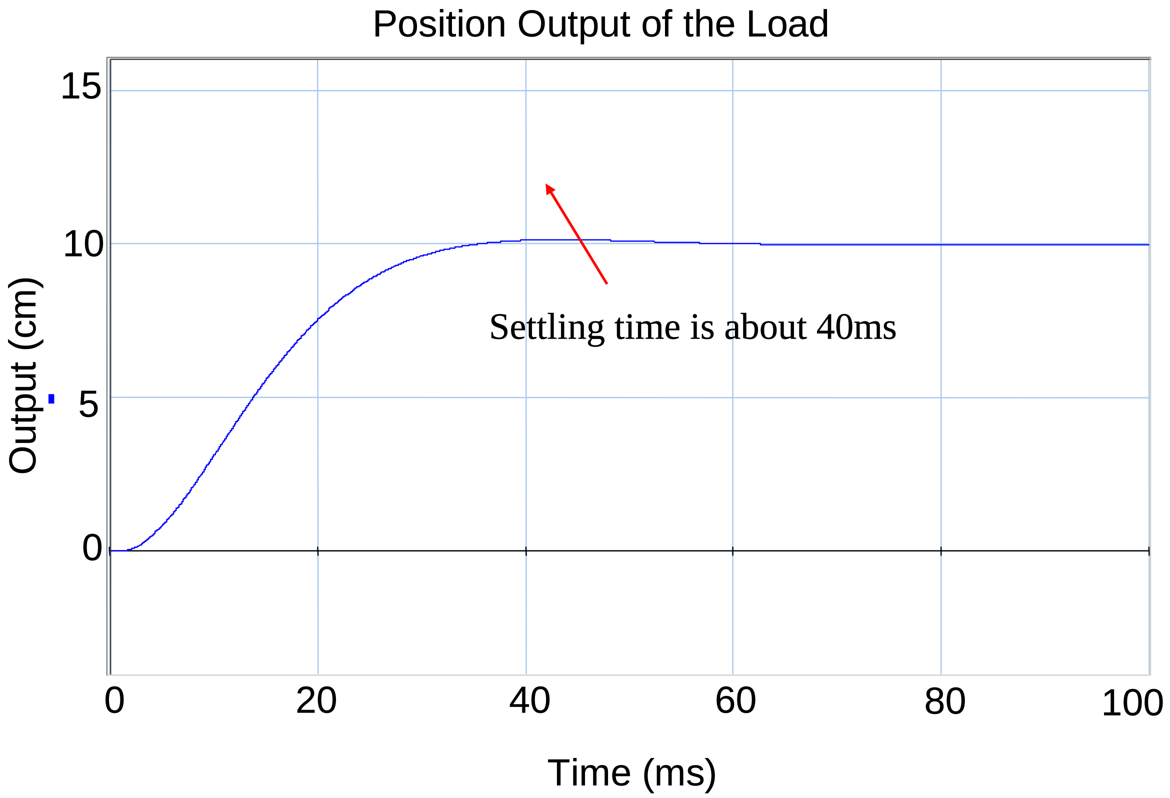


Figure 18. Simulation result of evolved bond graph model II

Run No.	Fitness of Printer	
	Distance	Fitness
1	15.076	0.985
2	15.818	0.984
3	15.188	0.985
4	16.720	0.983
5	15.053	0.985
6	14.085	0.986
7	15.122	0.985
8	15.502	0.985
9	15.132	0.985
10	15.881	0.984
Best	14.085	0.986
Worst	16.720	0.984
Average	15.358	0.985
Standard Deviation	0.6903	0.000669

Table 2. Summary results of fitness for printer

4. Conclusions

This research has explored a new automated approach for synthesizing designs for mechatronic systems. By taking advantage of genetic programming as a search method for competent designs and the bond graph as a representation for mechatronic systems, we have created a design environment in which open-ended topological search can be accomplished in a semi-automated and efficient manner and the design process thereby facilitated. By incorporating specific design considerations the method can be used to explore design space of special types of mechatronic systems such as robotic systems.

The paper illustrates the process of using this approach in detail through a typewriter redesign problem. Bond graphs have proven to be an effective tool for both modeling and design in this problem. Also a special form of GP, Hierarchical Fair Competition-GP, has been shown to be capable of providing a diversity of competing designs with great efficiency.

Our long-term target in this research is to design an integrated and interactive synthesis framework for mechatronic systems that covers the full spectrum of design processes, including customer needs analysis, product development, design requirements and constraints, automated synthesis, design verification, and life-cycle considerations.

Acknowledgments

The authors gratefully acknowledge the support of the National Science Foundation through grant DMI 0084934.

5. References

Coelingh H. ; T.J.A. de Vries & van Amerongen J. (1998). Automated Performance Assessment of Mechatronic Motion Systems during the Conceptual Design Stage. Proc. 3rd Int'l Conf. on Adv. Mechatronics, pp. 472-477, Okayama, Japan.

Fan Z., Hu J., Seo K., Goodman E., Rosenberg R., and Zhang B. (2001). Bond Graph Representation and GP for Automated Analog Filter Design, Proceedings of the Genetic and Evolutionary Computation Conference, pp. 81-86.

Grimbleby J. (2000). Automatic analogue circuit synthesis using genetic algoriths. IEE Proc. – Circuits Devices Syst, pp. 319-323.

Hu J., Goodman E. D., Seo K., Pei M., (2002). Adaptive Hierarchical Fair Competition Model for Parallel Evolutionary Algorithms, Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-2002, New York, pp. 772-779.

Karnopp D., Margolis D. & Rosenberg R. (2000). System Mechatronics: Modelling and Simulation of Mechatronic Systems. Third Edition. New York: John Wiley & Sons, Inc.

Koza J. (1992). Genetic Programming: On the Programming of Computers by Means of Natural Selection, The MIT Press.

Koza J., Bennett F. III, Andre D. & Keane M. (1999b). The design of analogue circuits by means of genetic programming. In P. J. Bentley (ed.), Evolutionary Design by Computers, 365-385. London: John Wiley & Sons Ltd.

Lohn J., Colombano S. (1999). A circuit representation techniques for automated circuit design. IEEE Transactions on Evolutionary Computation: 205-219.

Luke S., 1997, Strongly-Typed, Multithreaded C Genetic Programming Kernel, available from http://cs.gmu.edu/~sean/research/lil-gp-patch/ . Accessed: 2005-01-15

Paynter H. (1991). An epistemic prehistory of bond graphs. In P. C. Breedveld and G. Dauphin-Tanguy (ed.), Bond Graphs for Engineers, 3-17. Amsterdam, The Netherlands: Elsevier Science Publishers.

Redfield R. (1999). Bond Graphs in Mechatronic Systems Designs: Concepts for a Continuously Variable Transmission. International Conference on Bond Graph Modeling and Simulation, pp. 225-230.

Rosca J. , Ballard D. (1995) Causality in genetic programming. In L. Eshelman (ed.), Genetic Algorithms: Proceedings of the Sixth International Conference (ICGA95), pp. 256-263. San Francisco, CA: Morgan Kaufmann.

Rosenberg R., Whitesell J., & Reid J. (1992). Extendable Simulation Software for Mechatronic Systems. Simulation Vol. 58, pp. 175-183.

Seo K., Goodman E. & Rosenberg R. (2001). First steps toward automated design of mechatronic systems using bond graphs and genetic programming. Proceedings of the Genetic and Evolutionary Computation Conference, pp. 189.

Sharpe J., and Bracewell R. (1995). The Use of Bond Graph Reasoning for the Design of Interdisciplinary Schemes. International Conference on Bond Graph Modeling and Simulation, pp. 116-121.

Tay E., Flowers W. & Barrus J. (1998). Automated Genration and Analysis of Mechatronic System Designs. Research in Engineering Design, Vol. 10, pp. 15-29.

Wang J. (2004). Integrated Coevolutionary Synthesis of Mechatronic Systems Using Bond Graphs. PhD dissertation, Department of Mechanical and Industrial Engineering, University of Massachusetts, Amherst

Xia S., Linkens D. and Bennett S. (1991). Integration of qualitative reasoning and bond graphs: an engineering approach. In P. C. Breedveld and G. Dauphin-Tanguy(ed.), Bond Graphs for Engineers, pp. 323-332. Amsterdam, The Netherlands: Elsevier Science Publishers.

Youcef-Toumi K., Ye Y., Glaviano A., & Anderson P. (1999). Automated Zero Mechatronics: Derivation from Bond Graph Models. International Conference on Bond Graph Modeling and Simulation, pp. 39-44.

Zongker D. and Punch W., III, (1998), lil-gp 1.1 User's Manual, GARAGe, College of Engineering, Michigan State University, available from http://garage.cps.msu.edu/. Accessed: 2005-01-15

Online Identification for the Automated Threaded Fastening Using GUI Format

Nicola Ivan Giannoccaro & Mongkorn Klingajay

1. Introduction

Screw insertions are a common joining process and are especially popular in assemblies that need to be disassembled for repair, maintenance or relocation.

A human performing screw fastening will typically use four stages (L.D. Seneviratne et al., 1992):

1) Alignment: the holes in the mated parts are aligned and firmly held together.

2) Positioning: the screw is globally positioned with respect to the aligned holes.

3) Orientation: the screw is oriented until its axis coincides with the axis of the aligned holes.

4) Turning: the screw is turned with the appropriate torque until fastening is achieved.

In manual screw insertions, the torque exerted by the screwdriver depends mainly on the applied operator force. Human operators are particularly good at on-line monitoring of the operation. However, with power tools, the increased insertion speed reduces the human ability to monitor the insertions on-line. Thus on-line automated monitoring strategies for the screw fastening process are highly desirable. One such approach is based on the "Torque Vs. Insertion Depth" signal measured in real time; if this signal is within a pre-defined bound of the correct insertion signal, then the insertion is considered to be satisfactory. The torque signature signal for a correct insertion is either taught as predicted using an analytical model (Klingajay & Seneviratne, 2002). Industrial applications of automated screw insertions have been implemented in several forms. These achieved forms are applied in different objectives. With the development of electrically powered screwdrivers the attempts at automating the screw insertion process with emphasis on the torque signature signal vs. angle signal and become to the primary mathematical model. In 1997, this analytical model was implemented by (Ngemoh, 1997). The Neural Network techniques have been applied by using the ability of Weightless to monitor the screw insertion processes in difference insertion cases (Visuwan, 1999). Bruno has distinguished between successful and unsuccessful insertion based on Radial Basic function (Bruno, 2000). Both monitoring performs are to apply Artificial Neural Network in view points of classifications. "A distinction without a difference has been introduced by certain writers who distinguish 'Point estimation', meaning some process of arriving at an estimate without regard to its precision, from 'Interval estimation' in which the precision of the estimate is to some extent taken into account" (Fisher, 1959). Fisher founded the Probability theory as logic agree, which gives us automatically both point and interval estimates from a single calculation. The distinction commonly made between hypothesis testing and parameter estimations are considerably greater than that which concerned Fisher. In this point of view, it may be not a real difference. The screw fastening processes have carried out insertion on eight different materials of plate and the general self-tapping screws with using mainly screw AB sizes 4, 6, and 8. These self-tapping screw sizes are the most common sizes in manufacturing. The corresponding theoretical profiles of a curve of the torque signature signal and rotation angle for each set of insertions have been also generated using the mathematical model by (Seneviratne et al., 2001). The mechanical properties of the plate materials have used the theoretical model that was obtained from (Ngemoh,1997). The average properties have used the provided data sheets that were the published texts and materials standards specifications (ASM V.1, 1990); (ASM V.2 ,1990); (Verlag, 1993); (Walsh, 1993); and (Bashford, 1997).

2. The Experimental Test Rig Setup

The Implementation of Screw Insertion System model (ISIS) has been integrated with three main factors. The first one is the screwdriver with the pilot materials for attempt to insertion screw. The second one is the instruments and sensor controller. This controller consists of the Rotary transducer or Torque sensor for capture a torque signature signal and the Optical Encoder for measurement the rotation angle during the on-line process. This controller is including the torque meter and manipulator equipments. The third one is the monitoring system based-on the parameter estimation has employed for prediction of the required parameters of the threaded fastening process. These factors have interfaced and implemented by the Graphical User Interface technique (GUI) using Matlab programming. More details on these equipments have described in next.

2.1 Tools and Manipulations

The tools and manipulations are necessary as input devices, which brought the input signal to the process. An electrically powered screwdriver has used with a torque range varying between 0.4 and 3.2 Nm to drive the screw into the hole for this experimental testing. An illustration this screwdriver present in Fig.1.

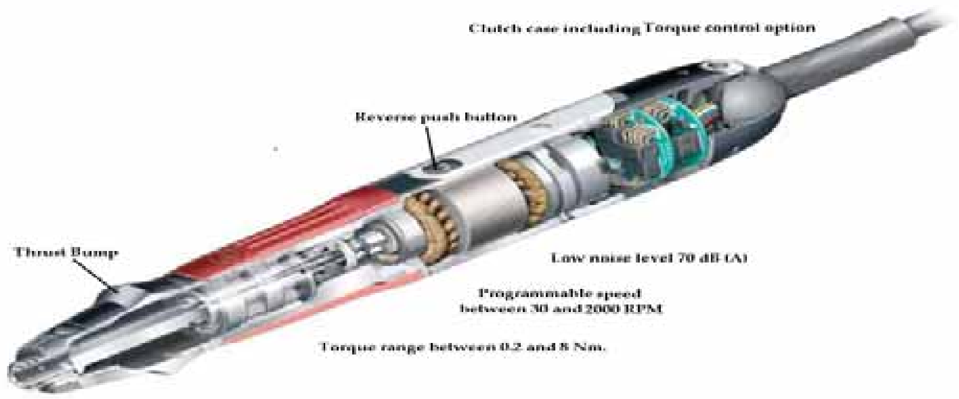

Clutch case including Torque control option
Reverse push button
Thrust Bump
Low noise level 70 dB (A)
Programmable speed
between 30 and 2000 RPM
Torque range between 0.2 and 8 Nm.

Figure 1. The illustration of screwdriver

2.2 The Instrument and Sensor Controller

The controller is main function to command the process that aims to control the instruments and sensors for the on-line operation. The integrated fastening process with GUI system has been employed and interfaced with the Data Acquisition (DAQ) card. The details of these devices are following.

2.2.1 Instrument and Sensor Controller

In Fig. 2, the rotary torque transducer has used in this experiment, which has attached to the shaft of the screwdriver between the end of screwdriver and the nut settings. This torque sensor is in position that can reduce the effects of inactive and friction associated of the gear and the drive motor and gears. This torque sensor is a measurement based on strain gauges with capable in measuring torque is in the range between 0 and 10 Nm with accuracy of 1% of full scale deflection.

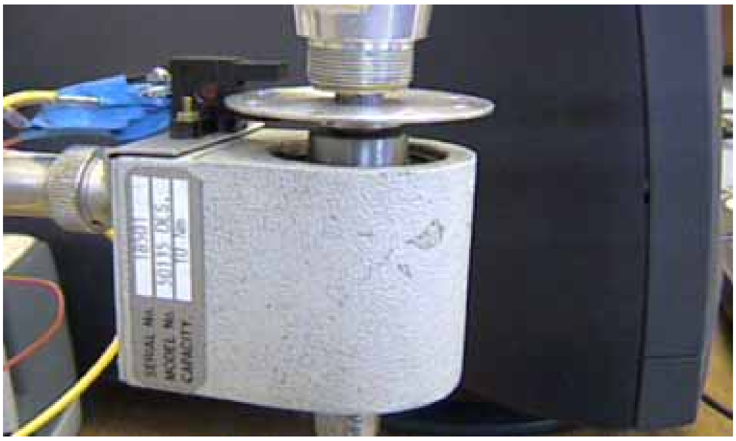


Figure 2. Torque Sensor

The Torque sensor was electronically interfaced with a digital torque meter that is produced and used of a strain gauge amplifier.

The voltage output of torque reading from the torque meter is proportional to the measured torque during screw insertion into the hole. Each output voltage has corresponded to the maximum transducer capacity of torque at 10 Nm.

2.2.2 Optical Encoder

An optical encoder has used to measure the rotation angle of the screw. This optical encoder consists of an optical switch and a measured disc.

The optical switch has functioned with a light sensitive device with a built-in amplifier, the shape and wiring diagram are presented in Fig. 3. It was mounted on the torque sensor casing.

The optical switch used in this test to be model of UZJ272. Its performance is to fix on the focus reflective/u-shaped type of the micro-photosensors low-cost.

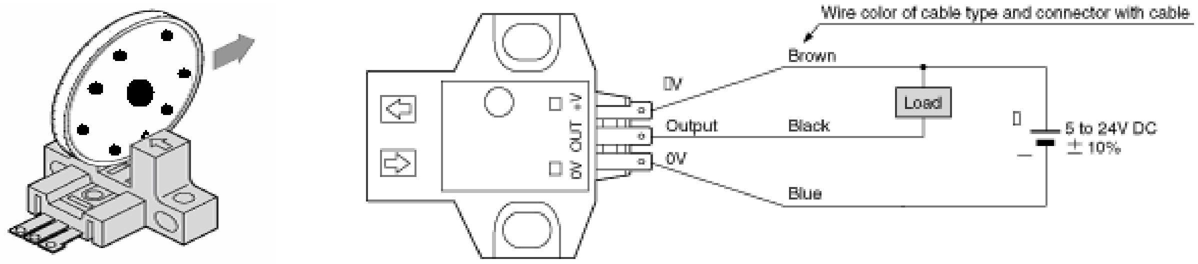


Figure 3.1 Optical Encoder

Figure 3.2 Typical wiring diagram

Figure 3. The application of Optical Encoder

A measurement disc has attached to the Torque sensor, which fixed at the shaft of the driver bit see Figs 2 and 8. This disc is perforated every 45° allowing 8 readings per revolution. Every time a hole in the plate is detected the opto-switch produces a voltage pulse by measurement strategy in Fig. 4.

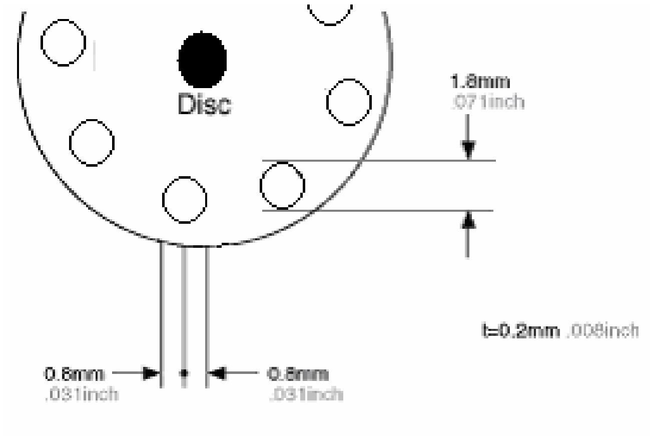


Figure 4. A measurement disc

The Optical encoder has a measurement resolution of 45°. This digital signal of a pulse voltage has been converted to the rotation angle (radial) during screw insertion. This application for reading signal on 8 holes as angle has present in Fig. 5.

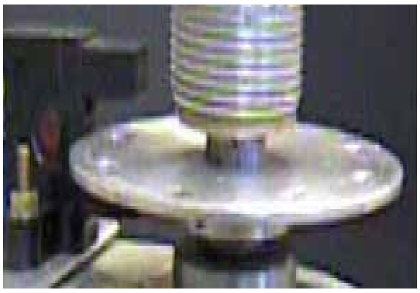


Figure 5. Optical Encoder (Opto switch and Disc)

2.2.3 The Reference Label Box or Connector Box

This reference label box is used to select the channel, port number, and type of signal for system to identify during on-line process. This box has connected with the SH68-68 cable to the Multifunction DAQ card see Figs. 6 and 7. The details of the Multifunction DAQ card has described in next.

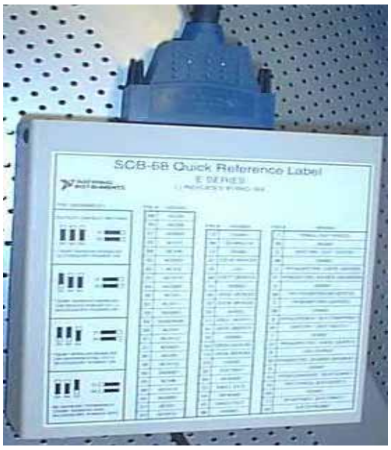


Figure 6. The wiring circuit in the Connector box

2.2.4 A Multifunction DAQ Card.

The multifunction DAQ 12 Bit Analog-Digital I/O card has used in this experimental test is product of National Instrument and Sensor (NI) with model "NI PCI 6024E". This is required as PCI bus, which is the low-cost E Series multifunction data acquisition device. It provides the full functionality of I/O 68-pin male 0.050 D-type with voltage output range of ± 10 volts.

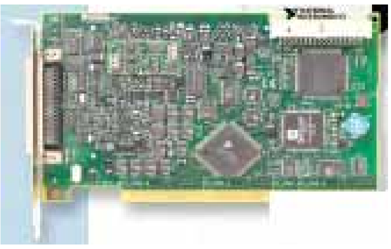


Figure 7 (a). A DAQ PCI card

Figure 7(b). A DAQ card with cable

Figure 7. Multifunction NI DAQ PCI 6024E card with application

In Fig. 7 has presented a multifunction DAQ that connected to the SH68-68 cable for install into a PCI slot on computer board. This card has used to archieved the signal from the sensors via the Connector box. The screwdriver attached the instruments with sensor controllers for this experimental test, shown in Fig. 8. The integrated component system (hardware) has applied for this experimental test of screw fastening system, which shown in Fig. 9.

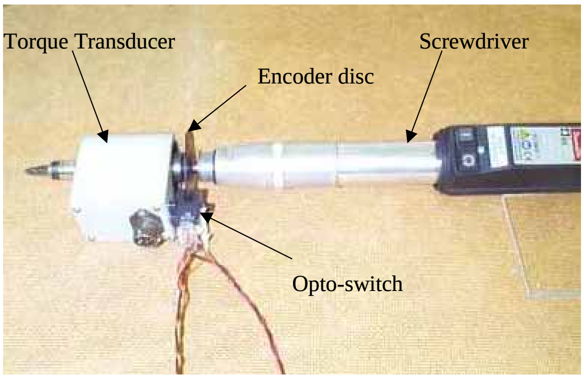


Figure 8. A screwdriver attached transducer and encoder

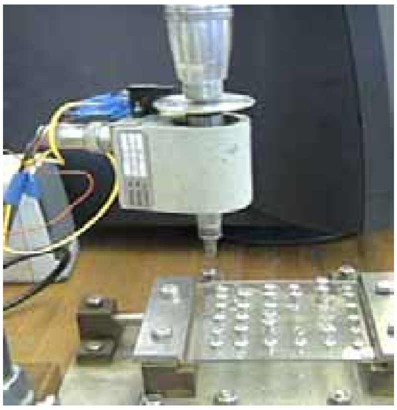


Figure 9. The screw fastening setup

3. Strategy for the On-Line Experimental Test

The experimental equipments are required to connect as the applied screw fastening software has implemented as the appropriated efficiency system with this integrated system has written using Graphic User Interface format (GUI) and Data Acquisition Toolbox of Matlab program to get signal from the torque sensor and optical encoder for Torque signature signal and rotation angle during the screw insertion. The captured signal has been applied the curve fitting and curve management techniques to identify the curve. The parameter estimation method has used the unknown parameters at this insertions. The details have presented the flowchart in Fig. 10.

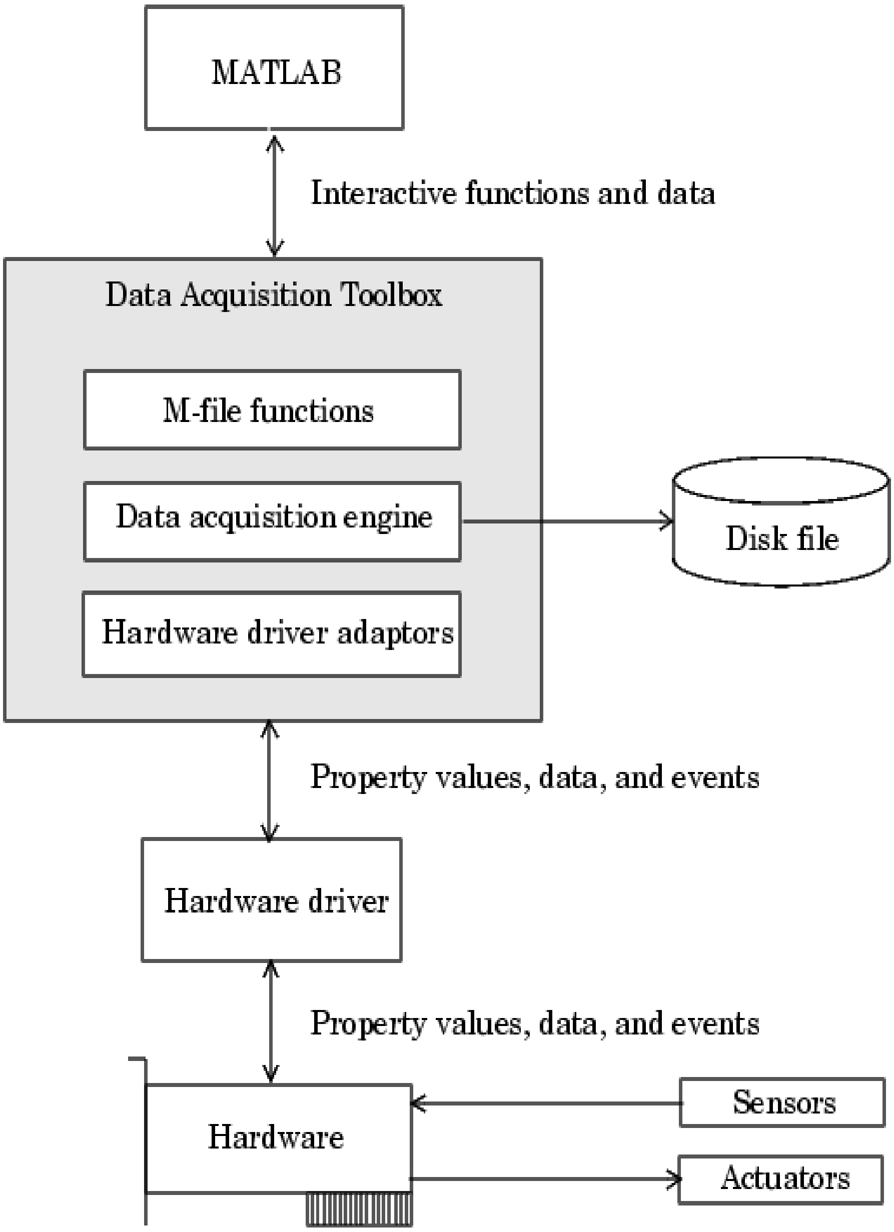


Figure 10. Flowchart of the on-line procedures

A flowchart in Fig. 10 presents the implemented procedure that applied the DAQ module to communicate between the screwdriver and the instruments and sensors.

The performance of the screw insertion identification tasks, the NRM technique is fed with torque-insertion depth signature signals. The screw insertions were performed with the use of a screwdriver. A torque sensor mounted at the tip of the screwdriver provides torque readings, and an optical encoder provides pulses, which are related to the screw rotation angle. The stream of pulses was integrated to determine the corresponding insertion depth. Model was developed and using the curve fitting technique to filter the signal. The experimental test rig prior to presentation to the on-line screw fastening process for the parameter identification has been described in next.

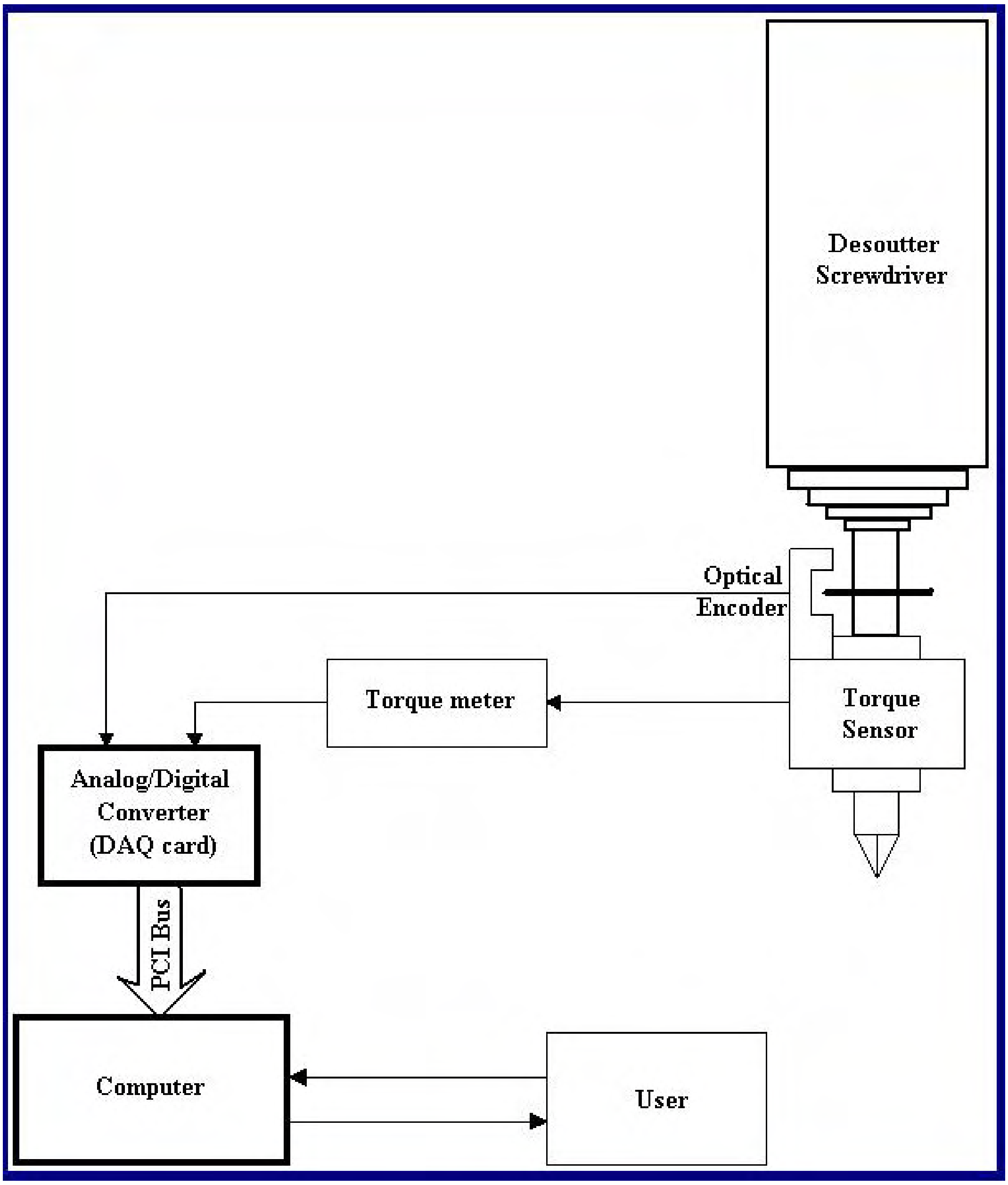


Figure 11. The overview of the experimental test rig

An online screw fastening process software has used to manage all instruments and sensors for the input task and interfacing with the screwdriver and linking with the monitoring based-on parameters estimation. The main functions of this control system have been implemented as GUI format. Reading the torque vs. time data from the torque transducer and the rotation angle vs. time data from the optical encoder. These online results can be presented the signal data in Fig. 12.

In this paper has used the monitoring technique based on parameter estimation to validate the experimental tests with simulation tests (theoretical model). The parameter estimation based on Newton Raphson Method has used to identify the unknown parameters. This estimation algorithm has applied to the torque signature signal after fitting the smooth curve by the curve fitting technique. This curve fitting technique has described more details in next.

3.1 Curve Fitting Technique on the Experiment

The smooth curve can be fitted and applied to the experimental signal. Using the interpolating polynomial of nth degree then we can obtain a piecewise use of polynomials, this technique is to fit the original data with the polynomial of degree through subsets of the data points.

This result is quite suitable approach as a least square fit algorithm. That has applied these fitting on points with the exactly points so close to actual curve when has been validated with theoretical curve.

Curve fitting to measure the signal data is a frequently occurring problem. These signals are two vectors x and y of equal size. The situation is aims to fit the data because being a dependency of y on x in form of a polynomial. The method of least squares can be used to solve the over constrained linear system which gives the coefficients of the fitting polynomial. Therefore, the higher degree polynomial has represented the dependency for this fitting technique.

For a given data set (x; y) - where x and y are vectors of the same size. The least square technique can be used to compute a polynomial for that best fits these data has used a polyfit function at the degree of the fitted polynomial (Klingajay & Giannoccaro, 2004).

An illustration problem on this technique has applied on experimentally with the relationship between two variables x and y that have recorded in a laboratory experiment in Fig. 12 (a). However, this captured curve has been fitted for a new validated smooth curve with the curve fitting algorithm has described in next.

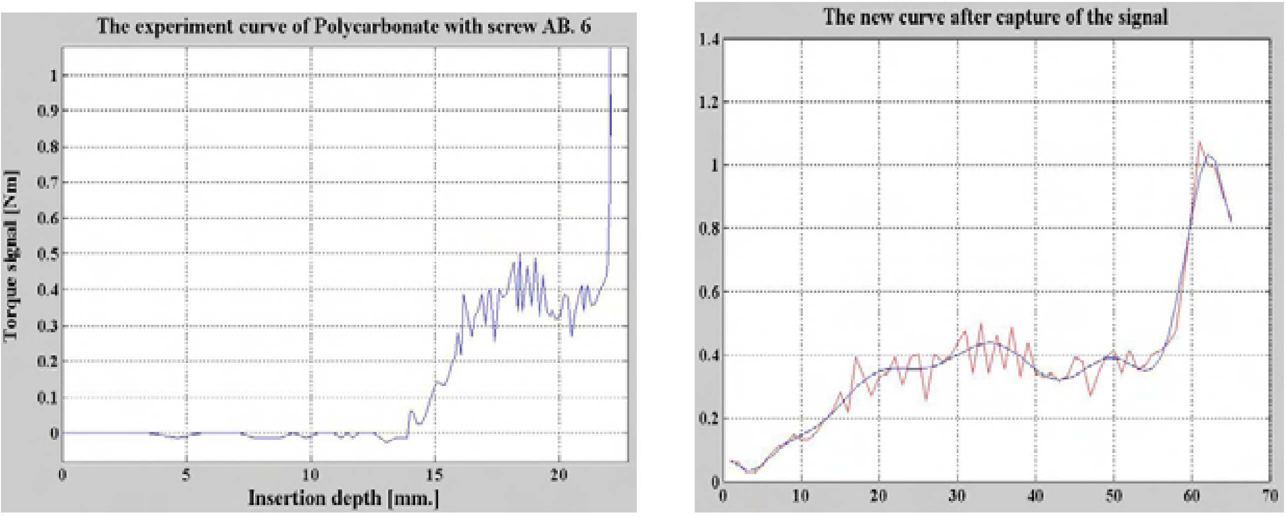


Figure 12 (a). An On-line signal.

Figure 12 (b).A curve of fitted signal.

Figure 12. The noise reduction using curve fitting strategy.

In Fig. 12 (a) shows a captured curve from the experimental test during on-line insertion. The curve fitting strategy based-on Least Square technique has been applied to fit on this captured curve.

This a new smooth curve after fitting process has presented in Fig. 12(b). The on-line curve has been adjusted for the actual starting point of insertion then the new smooth fitted curve has presented the experiment torque signature signal that can be compared with the theoretical curve in Fig. 13.

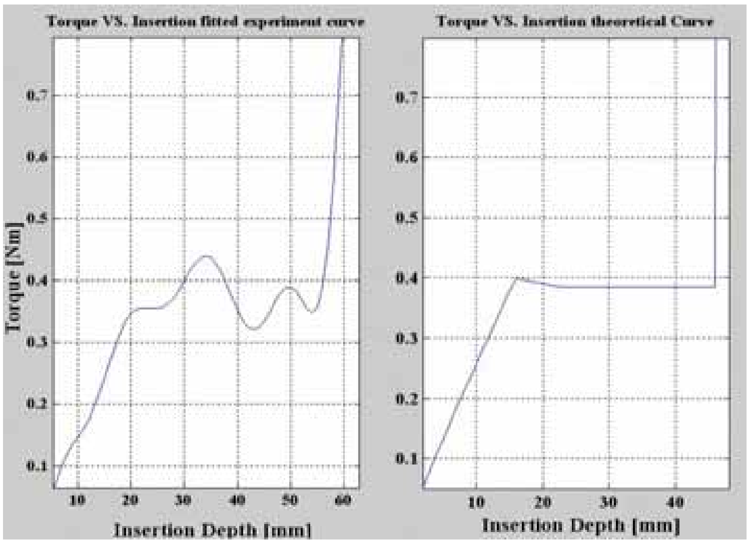


Figure 13. The theoretical and experimental curves.

3.2 The Experiment Curve Identification

The screw insertion process has been described on the required parameters depend on the each equation in the five different stages of insertion. Therefore, this NRM algorithm has applied with screw insertion process that required the exactly Displacement of depth, which has applied for those curves in each stage. Thus, the curve identification technique has become the important technique to employ for recognition the curves of the torque signature signal and displacement of depth of insertion.

Basically, the actual curve of the torque signature signal and rotation angle is the non-linear curve as being discontinue in each stage of insertions. These stages have divided the nature curves that could be generated from simulation tests or captured during experiment tests.

	Experiment Properties	
Material and Plate Properties	Material type	Polycarbonate
	Tap Hole Diameter (D_h)	2.0 mm (0.0020 Meters)
	Tensile Strength (σ_{UTS})	45Mpa
	Yield Strength (σ_Y)	45Mpa
	Elastic Modulus (E)	2.35Gpa
	Coefficient of Friction (μ)	0.19
	Tap (far) plate thickness (T_2)	3 mm (0.003 Meters)
Screw Properties	Screw type	AB No. 6
	Screw major diameter (D_s)	3.42mm (0.00342 Meters)
	Screw head diameter (D_{sh})	6.52mm (0.00652 Meters)
	Screw thread pitch (P)	1.19mm (0.00119 Meters)
	Screw taper-length (L_t)	2.94mm (0.00294 Meters)
	Screw total threaded-length (L_s)	9.67mm (0.00967 Meters)

Table 1. Parameters for insertion of Polycarbonate and screw AB 6

The screw, material, and plate properties in Table 1 have been used to produce the simulation screw insertion curve in Fig. 14(a).
This insertion curve has been identified the insertion stages of stages 2, 4, and 5, shown in Figs. (14 (b) - 14 (d)).

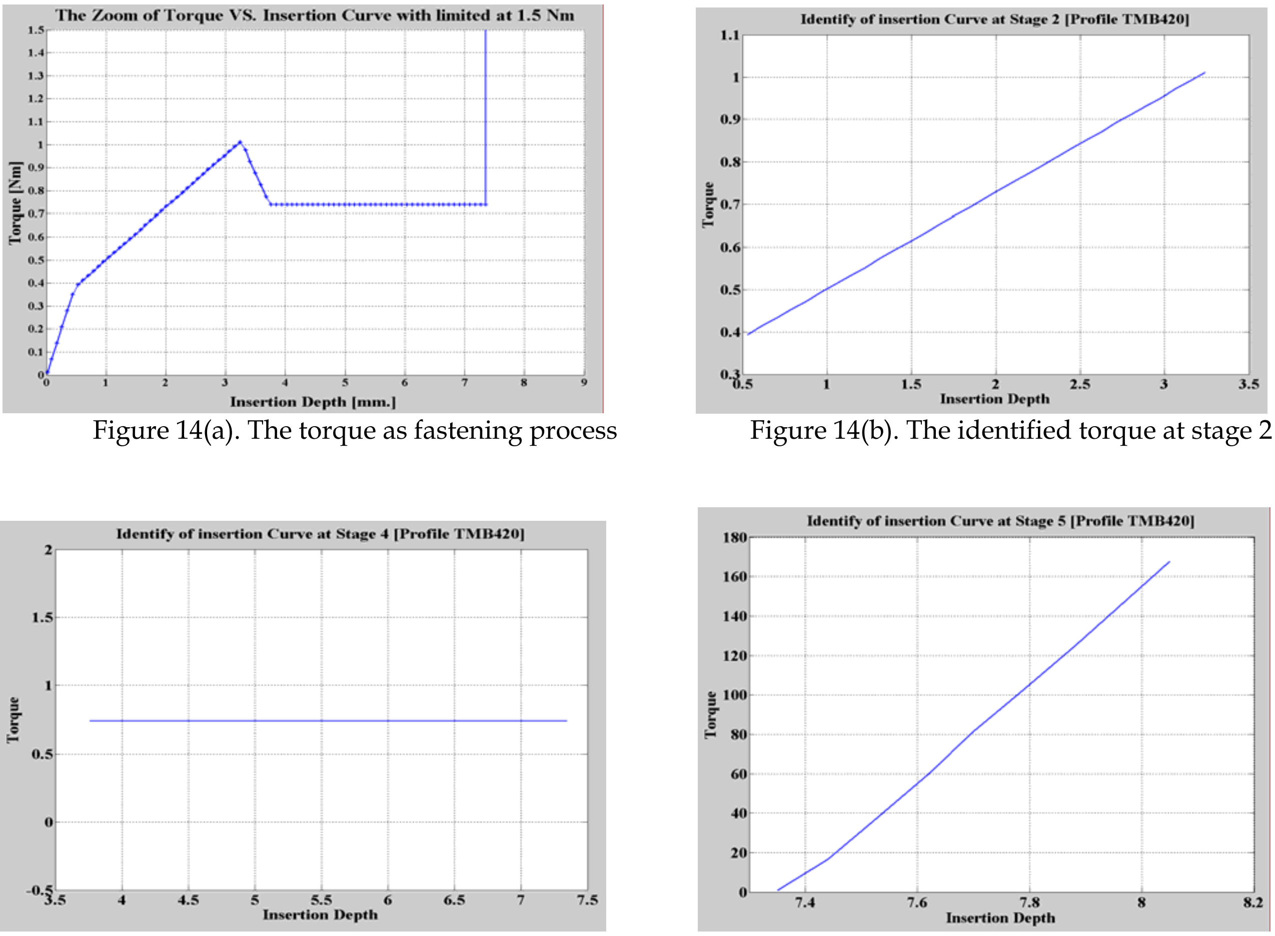


Figure 14(a). The torque as fastening process

Figure 14(b). The identified torque at stage 2

Figure 14(b). The identified torque at stage 4

Figure 14(b). The identified torque at stage 5

Fig. 14. The curve torque signature signal for screw AB6 with Polycarbonate.

The identified insertion curve at stages 2, 4, and 5 have shown in Figs. 14(b), 14(c), and 14(d) respectively. This behaviour presented to status of the screw that had turned from starting until the final of tightening been reached. These stage curves have used to estimate for the experiment of parameter identification. The methodology of parameter estimation has discribed in next.

4. Methodology of Parameter Estimation

In this paper present what is probably the most commonly used techniques for parameter estimation. The appropriable techniques for the particular attention in this paper devote to discussions about the Newton Raphson Methods is a choice of appropriate minimization criteria and the robustness of this application. The monitoring strategies have been applied the parameter estimation techniques to validate the screw insertion process in this experimental test.

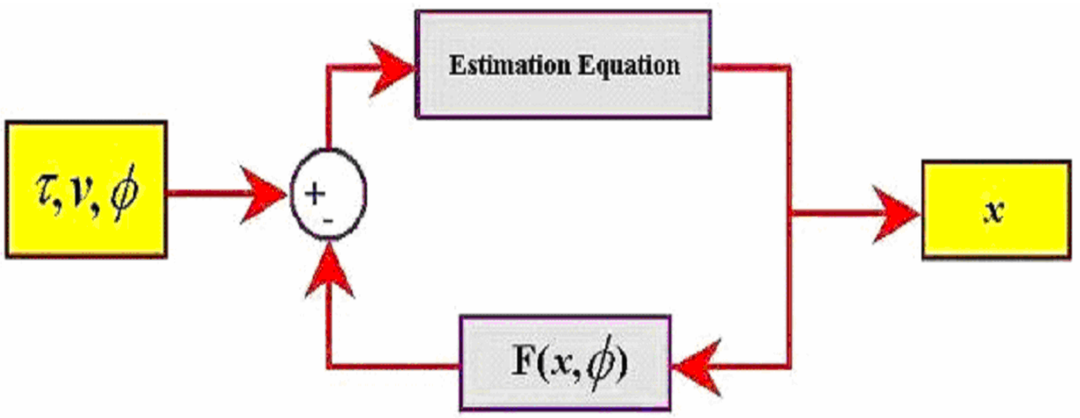


Figure 15. Estimation Parameters Scheme

Fig. 15, presents the scheme of the monitoring based on Parameter Estimation process. The main contribution of this paper is to apply the methodology for the estimated parameter application based on Newton Raphson Method (NRM) for screw fastening process with testing on the torque signature signal VS. insertion angle using the insertion stage 2. The model of monitoring has implemented with the estimation parameters Scheme in Fig. 15. In this Scheme, the required parameters have read the input data with contain both known and unknown parameters. A V vector consists of known parameters, the rotation angle (ϕ), a torque signature signal (τ), and the unknown parameter (vector xn) is an initialed value.

5. Analytical Model for Parameter Estimation

The focus on the experimental test parameter estimation based on NRM, which is the non-linear estimation parameter techniques that can solve problem over threaded fastening for screw insertion process. With parameters on the screw properties, plate properties, and material properties including friction and Ds, these techniques have been applied to predict two unknown parameters in this paper.

The two unknown parameters are required by the mathematical model can be estimated reliably on-line. experimental results without noise is presented to validate the estimation procedure in this paper. A self-tapping screw geometry and analytical model is mentioned in Ngemoh (Ngemoh, 1997), (Seneviratne et al., 1992), and (Klingajay & Seneviratne, 2002) that consist of five equations corresponding to each insertion stage:

$$\tau_1 = \frac{1}{\alpha} R_s A_c \sigma_{UTS} \cos\theta . \phi + \mu R_f K_f \sigma_f \cos\theta(\phi) \tag{1}$$

$$\tau_2 = R_s A_c \sigma_{UTS} \cos\theta + 2\mu R_f K_f \sigma_f \cos\theta(\phi - \alpha) \tag{2}$$

$$\tau_3 = \frac{1}{\alpha} R_s A_c \sigma_{UTS} \cos\theta + \mu R_f K_f \sigma_f \cos\theta(\phi_b + \phi) \tag{3}$$

$$\tau_4 = 2\mu R_f K_f \sigma_f \cos\theta(\phi_b) \tag{4}$$

$$\tau_5 = \left[\left(\frac{\mu\left(D_{sh}^3 + D_s^3\right)}{3\left(D_{sh}^2 - D_s^2\right)} \right) + \left(\frac{D_s + D_h}{4} \right) * \left(\frac{\pi\mu\left(D_s + D_h\right) + 2P}{\pi\left(D_s + D_h\right) - 2\mu P} \right) \right] * \left[\frac{K_{th} * K_{tb}}{K_{th} + K_{th}} \right] \tag{5}$$

Equations (1 – 5) can be written as

$$\tau = F(X, \phi), \tag{6}$$

Where ϕ is the screw angular rotation, X is the vector of system parameters:

$X = [D_h, D_r, D_s, D_{sh}, P, L_s, L_t, T_1, T_2, E, \mu, \sigma_Y, \sigma_{UTS}]$.

Where D_h = Tap Hole Diameter
D_r = Screw root diameter
D_s = Screw major diameter
D_{sh} = Screw head diameter
P = Screw thread pitch
L_s = Screw total threaded-length
L_t = Screw taper-length
T_1 = Tap (near) plate thickness
T_2 = Tap (far) plate thickness
E = Elastic Modulus
μ = Coefficient of Friction
σ_Y = Yield Strength
σ_{UTS} = Tensile Strength

Equations (1 -5) and the following variables $[\theta, \alpha, \beta, A_c, \phi_b, K_{th}, K_{tb}, R_s, R_f, K_f]$ are all a function of X, and are given and mentioned in (Seneviratne et al., 1992) and (Klingajay & Seneviratne, 2002). These equations have more details in Appendix. Thus given the system parameter vector $(D_h, D_r, D_s, D_{sh}, P, L_s, L_t, T_1, T_2, E, \mu, \sigma_Y, \sigma_{UTS})$, these parameters are important parameter that to be both variable and fixed parameters. These required parameters can be used to predict the torque signature signals. As the torque signature signal at stages 1 and stage 3 is very small value and too difficult to apply. Therefore, the torque signal at stages 2, 4, and 5 are used to investigate in this paper. An estimation parameter has been employed to predict the unknown parameters on the torque signature during screw insertion. As the torque signature signals are non-linear, therefore, the technique is applied with the multi-phase regression with all required parameters on the screw properties, plate properties, and material properties, including friction. The common methods often used is the Newton-Raphson Method. This methodology has been applied with the appropriate ability for this work on simulation tests and experimental tests (Klingajay et al., 2004).

This paper has used the mathematical model that is presented in (Seneviratne et al., 1992); (Seneviratne et al., 2001); (Klingajay et al., 2002). A model of the self-tapping screw fastening process has used the equation of five stages. Initially, the stage 2 equations, that is the torque required to drive the screw from screw engagement till initial breakthrough is used for parameter estimation in equation (2), can be rewritten as:

$$\tau_2 = f(\mu, \theta, \sigma_{UTS}, \beta, \phi, D_s, D_h, L_t, P\ ) \tag{7}$$

In conventional estimation theory, these parameters of screw insertion are generally determined using the technique of numerical method with the initial value and number of independent samples. The field of optimisation is interdisciplinary in nature, and has made a significant impact on many areas of technology. As the result, optimisation is an indispensable tool for many practitioners in various fields. Usual optimisation techniques are well established and widely published in many excellent publishing, magazines, and textbooks depend on their applications. For all their complexity, the algorithms for optimising a multidimensional function are routine mathematical procedures. An algorithm of Newton-Raphson Method (NRM) has developed and used for inversion purposes is presented. It achieves convergence in about nth iterations and produces exact values of the parameters depends on the number of unknown parameter that is going to apply. The curves of torque signature and insertion angle signal are simulated using the successful data from Ngemoh (Ngemoh, 1997) to validate in different screw, material, and plate properties.

The Newton Raphson Method (NRM) is based on the generalitazion with considering for the j^{th} function f_j of n functions $\{f_1, f_2, . f_n\}$ that define our system. This is to calculate the total derivative as sum of partial derivatives respect to the n variables $\{x_1, x_2, \ldots x_n\}$ of the functions (Klingajay & Seneviratne, 2003).

However, this model requires various parameters as input, before the torque signals can be predicted and used to automate the operation. Both equations (2 and 7) can be revised for the two unknown parameters as:

$$\tau_2 = F(V, x, \phi)$$

Where V is the vector of known parameters ϕ is the screw rotation angle, and x is the vector of two unknown parameters in this identification for two unknown parameters of Friction (μ) and Ds that can be followed:

$$x = \begin{bmatrix} \mu \\ D_s \end{bmatrix}$$

The NRM works by modifying the unknown parameters in order to force an error function to approach zero:

$$f(x, \phi) = \tau_2(\phi) - F(V, x, \phi). \qquad (8)$$

Rotation angle ϕ is chosen here as the independent variable, and the remaining parameters are investigated as a function of ϕ. The error function $f(x, \phi)$ is sampled at two distinct instances of ϕ, providing two independent equations, which can be solved in an iterative manner to find the two unknown parameters by generate two equations by sampling at two angle locations (Klingajay et al., 2003):

$$f_1(x, \phi_1) = 0 \text{ at } \phi_1$$

$$f_2(x, \phi_2) = 0 \text{ at } \phi_2$$

Let $\bar{x}(k)$ be the kth estimate of the unknown parameters. Applying the NRM, x can be calculated iteratively:

$$\bar{x}(k+1)=\bar{x}(k)-J^{-1}\begin{bmatrix} f_1(x,\phi_1)\\ \\ f_2(x,\phi_2)\end{bmatrix} \tag{9}$$

Applying Equation (9) iteratively until the error is reduced to a value close to zero will provide an estimate of the unknown parameters.

7. Test Results

The estimation algorithm is implemented and tested. The screw and material properties in Table 1 is used.

The integrated system has implemented in Graphic User Interface format (GUI) and linked-up with the Data Acquisition Toolbox in Matlab to get the signal from the torque sensor and optical encoder. The designed system has presented in Fig. 16, the estimated results have shown in Figs. 17.

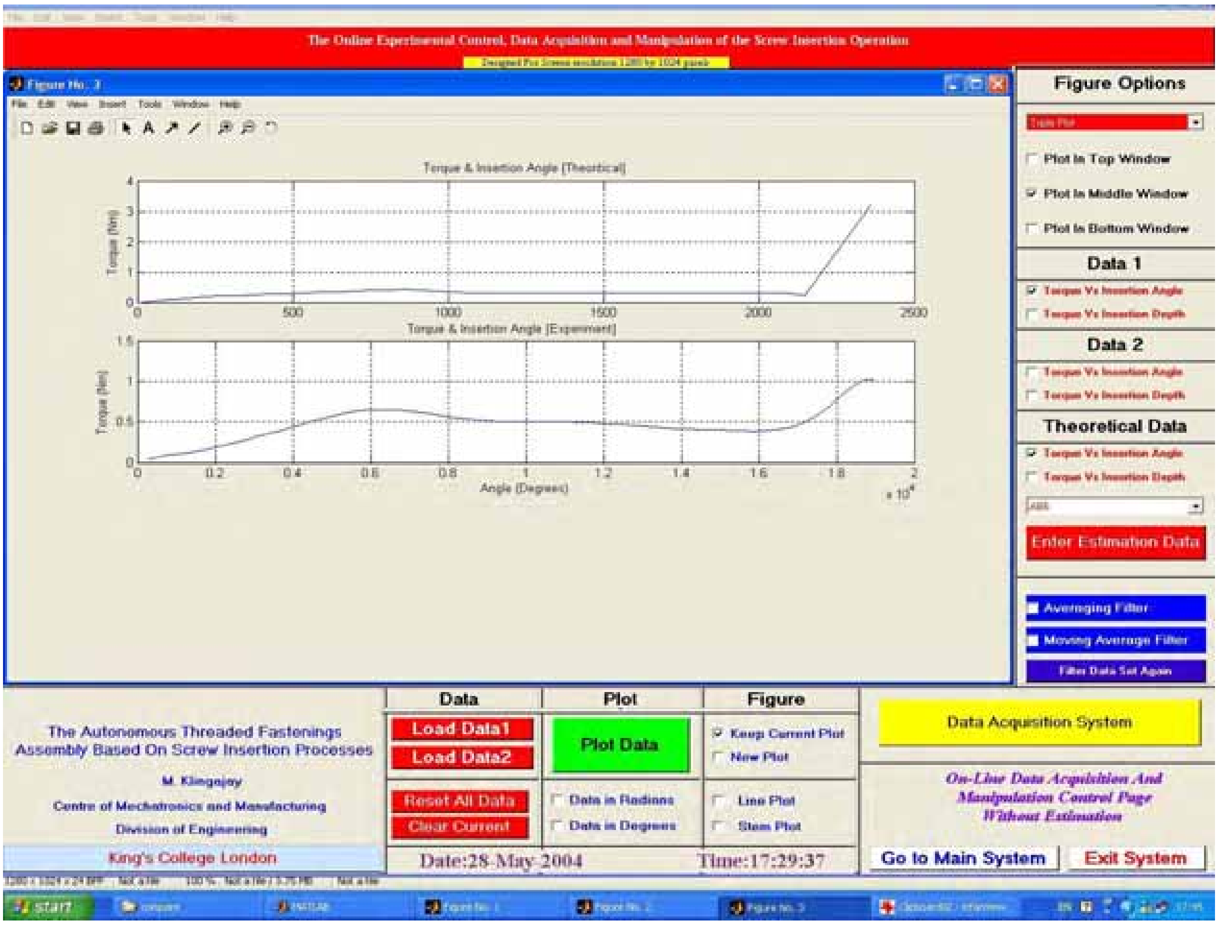
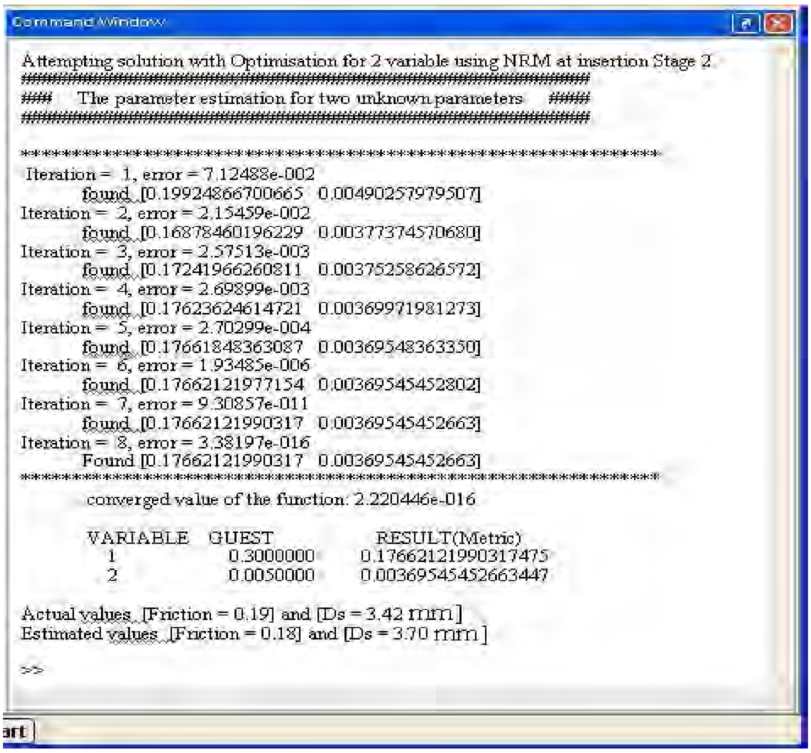

The Online Experimental Control, Data Acquisition and Manipulation of the Screw Insertion Operation
Designed For Screen resolution 1280 by 1024 pixels
Figure No. 3
File Edit View Insert Tools Window Help
Torque & Insertion Angle [Theortical]
Torque (Nm)
0
1
2
3
4
0
500
1000
1500
2000
2500
Torque & Insertion Angle [Experiment]
Torque (Nm)
0
0.5
1
1.5
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
Angle (Degrees)
x 10^4
Figure Options
Triple Plot
□ Plot In Top Window
☑ Plot In Middle Window
□ Plot In Bottom Window
Data 1
☑ Torque Vs Insertion Angle
□ Torque Vs Insertion Depth
Data 2
□ Torque Vs Insertion Angle
□ Torque Vs Insertion Depth
Theoretical Data
☑ Torque Vs Insertion Angle
□ Torque Vs Insertion Depth
ABS
Enter Estimation Data
Averaging Filter
Moving Average Filter
Filter Data Set Again
The Autonomous Threaded Fastenings Assembly Based On Screw Insertion Processes
M. Klingajay
Centre of Mechatronics and Manufacturing
Division of Engineering
King's College London
Data
Load Data1
Load Data2
Reset All Data
Clear Current
Plot
Plot Data
□ Data in Radians
□ Data in Degrees
Figure
☑ Keep Current Plot
□ New Plot
□ Line Plot
□ Stem Plot
Data Acquisition System
On-Line Data Acquisition And Manipulation Control Page Without Estimation
Date:28-May-2004
Time:17:29:37
Go to Main System
Exit System
start

Figure 16. The online integrated system for threaded fastening (GUI)



```
Attempting solution with Optimisation for 2 variable using NRM at insertion Stage 2
####################################################################################
####    The parameter estimation for two unknown parameters    #####
####################################################################################

**********************************************************************
 Iteration =  1, error = 7.12488e-002
        found  [0.19924866700665   0.00490257979507]
Iteration =  2, error = 2.15459e-002
        found  [0.16878460196229   0.00377374570680]
Iteration =  3, error = 2.57513e-003
        found  [0.17241966260811   0.00375258626572]
Iteration =  4, error = 2.69899e-003
        found  [0.17623624614721   0.00369971981273]
Iteration =  5, error = 2.70299e-004
        found  [0.17661848363087   0.00369548363350]
Iteration =  6, error = 1.93485e-006
        found  [0.17662121977154   0.00369545452802]
Iteration =  7, error = 9.30857e-011
        found  [0.17662121990317   0.00369545452663]
Iteration =  8, error = 3.38197e-016
        Found [0.17662121990317   0.00369545452663]
**********************************************************************
         converged value of the function: 2.220446e-016

         VARIABLE   GUEST              RESULT(Metric)
            1              0.3000000      0.17662121990317475
            2              0.0050000      0.00369545452663447

Actual values  [Friction = 0.19] and [Ds = 3.42 mm]
Estimated values  [Friction = 0.18] and [Ds = 3.70 mm]

>>
```

Figure 17. The experimental results estimating μ and Ds

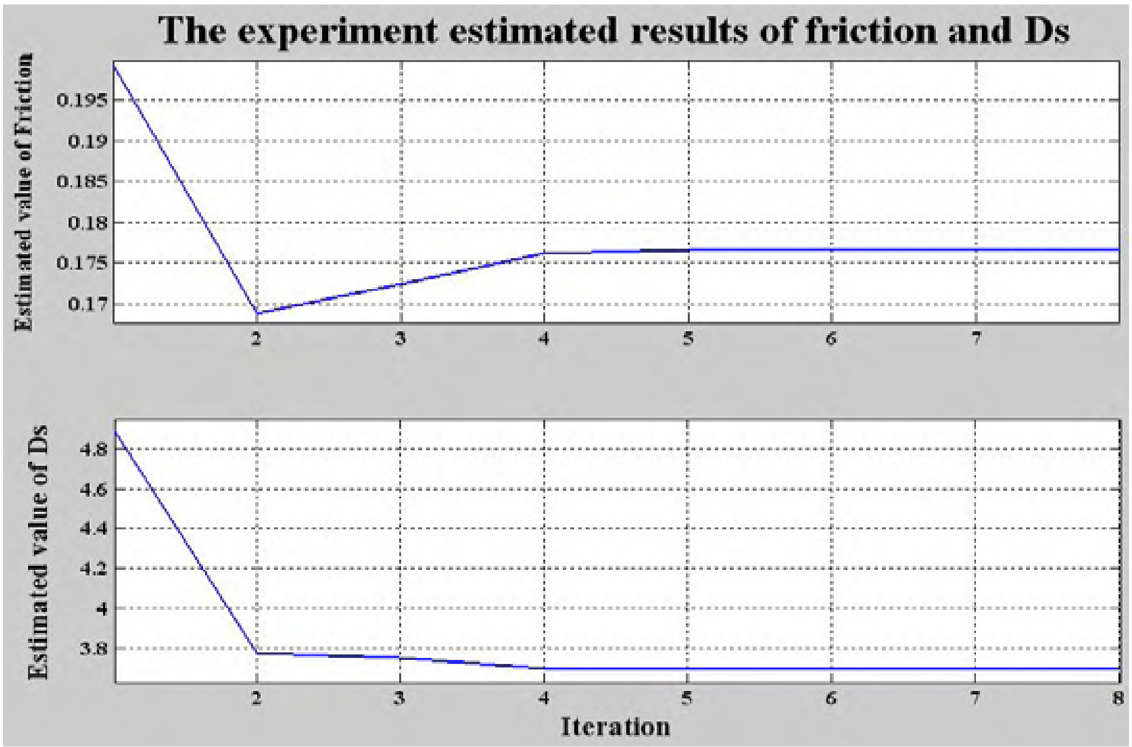


Figure 18. The estimated parameters without noise

Fig. 18, shows the estimated parameters of μ and Ds, the estimated values of these parameters are 0.18 mm. and 3.69 mm. However, these actual values of these parameters are 0.19 mm. and 3.42 mm. The percentage errors of these parameters of μ and Ds are 5.26% and 8.18% respectively, see Fig. 19 (Klingajay & Giannoccaro, 2004).

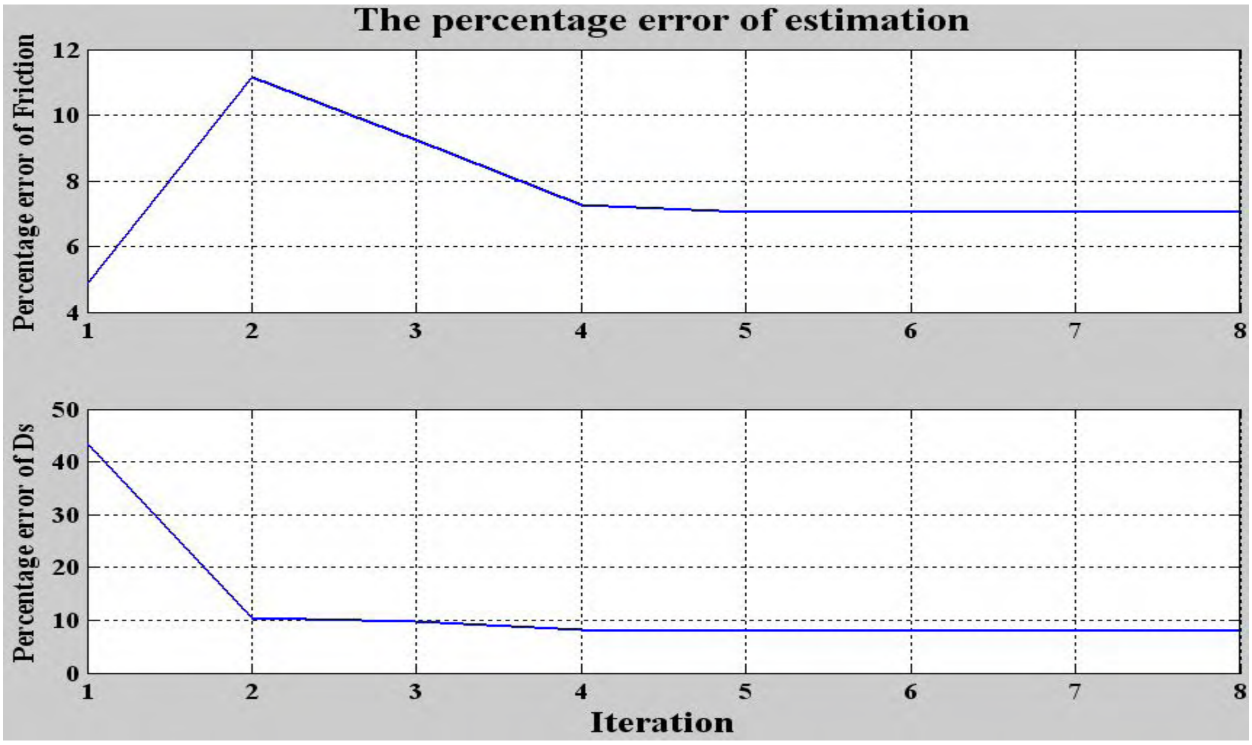


Figure 19. The percentage error of estimation without noise

8. Conclusion

The experimental tests have presented the validation of estimated parameters with the experiment data from [Ngemoh97] to validate. The test results has shown that the estimated parameters can be identified up-to four parameters for this experimental tests.

9. APPENDIX : Equations for Torque Signature Signal

The mathematical model given in [1,2] is concluded below.
The torque signature curve is shown by five differences stages (Klingajay et al., 2003):

Stage 1 - Screw Engagement $\{ 0 \leq \phi \leq \alpha \}$
$\tau_1 = 1/\alpha R_s A_c \sigma_{UTS} \cos \theta.\phi + \mu R_f K_f \sigma_f \cos\theta(\phi)$

Stage 2 - ***Screw Breaks through*** $\{ \alpha \leq \phi \leq \phi_b \}$
$\tau_2 = R_s A_c \sigma_{UTS} \cos \theta + 2\mu R_f K_f \sigma_f \cos\theta(\phi-\alpha)$

Stage 3 - Screw Full-Breakthrough $\{\phi_b \leq \phi \leq \phi_{b+} \alpha \}$
$\tau_3 = 1/\alpha R_s A_c \sigma_{UTS} \cos\theta(\phi_b-\phi+\alpha)+\mu R_f K_f \sigma_f \cos\theta(\phi_{b+}\phi-\alpha)$

Stage 4 - Screw Sliding $\{\phi_b+\alpha \leq \phi \leq \phi_t \}$
$\tau_4 = 2\mu R_f K_f \sigma_f \cos\theta(\phi_b)$

Stage 5 - Screw Tightening $\{ \phi \geq \phi_t \}$

$$\tau_5 = \left[\left(\frac{\mu_h \left(D_{sh}^3 + D_s^3 \right)}{3\left(D_{sh}^2 - D_s^2 \right)} \right) + \left(\frac{D_s + D_h}{4} \right) \left(\frac{\pi\mu\left(D_s + D_h \right) + 2P}{\pi\left(D_s + D_h \right) - 2\mu P} \right) \right] \left[\frac{K_{th} K_{tb}}{K_{th} + K_{th}} \right] \left[\frac{P\left(\phi - \phi_t \right)}{2\pi} \right]$$

10. References

ASM V.1 (1990). Properties and Selection: Irons, Steels and High Performance Alloys, Metals Handbook Vol.1, ASM International, 1990.

ASM V.2 (1990). Properties and Selection: Non-ferrous alloys Special-purpose Materials, Metals Handbook Vol.2, ASM International, 1990.

Bashford, D. (1997). Thermoplastics Databook, Chapman and Hall, London, 1997.

Bruno, L. G. (2000). Intelligent Monitoring of Screw Insertions, Ph.D.Thesis, University of London, 2000.

Fisher, R. A. (1959). Statistical Methods and Scientific Inference, Oliver & Boyd, London, Second Revised Edition, Hafner Publishing Co., New York, 1959.

Klingajay, M. & Seneviratne L.D.(2002). Friction Estimation in Automated Screw Insertions, Proceedings of International Symposium on Robotics&Automation (ISRA2002), 2002, Vol.1, pp.682-688.

Klingajay, M.; Seneviratne, L. D. & Althoefer, K. (2002), Parameter estimation during automated screw insertions, Proceedings of 2002 IEEE International Conference on Industrial Technology (ICIT'02), 2002, Vol. II, pp.1019-1024.

Klingajay, M.;, Seneviratne, L.D. & Giannoccaro, N.I. (2003). Optimization of four estimated parameters for autonomous threaded fastenings, Proceedings of 12th International Workshop on Robotics in Alpe-Adria-Danube Region, RAAD2003, 2003, Italy.

Klingajay, M.; Seneviratne, L. D.; Althoefer, K. & Zweiri,Y. (2003). Parameter Estimations for threaded assembly based on the Newton Raphson method, Proceedings of 2003 IEEE the seventh International Conference on Automation and Technology(Automation2003), 2003, Vol.1, pp. 864-869.

Klingajay, M.; Seneviratne, L. D. & Althoefer, K.(2003). Identification of threaded fasteningparameters using the Newton Raphson Method, Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2003), 2003, Vol.1, pp.5055-2060.

Klingajay, M. & Giannoccaro, N. I. (2003). Parametric Estimation of Autonomous Threaded Fastenings Operation based on Non-linear Least Square Method, Proceedings of the ISCA 16th International Conference: Computer Application in Industry and Engineering (CAINE 2003), 2003, USA, Vol. 1, pp. 211-215.

Klingajay, M. & Giannoccaro N. I. (2003), Comparison between Least Square & Newton Raphson for Estimation Parameters of an Autonomous Threaded Fastenings, Proceedings of 2003 IEEE International Conference on Industrial Technology (ICIT'03), 2003, Vol. 1, pp.163-168.

Klingajay, M.; Seneviratne, L. D. & Althoefer, K.(2004). Model Based Automation Strategies for Threaded Assembly Systems, Proceedings of International Federation of Automatic Control (IFAC), Problems in Manufacturing (INCOM2004), 2004, Brazil.

Klingajay, M.; Giannoccaro, N.I. (2004). The Automated Threaded Fastening Based on On-line Identification, International Journal of Advanced Robotic Systems (IJARS), December, 2004, Vienna, Vol. 1, No. 4, pp. 263-272.

Ngemoh, F.A. (1997). Modelling the Automated Screw Insertion Process, Ph.D.Thesis, University of London. 1997.

Seneviratne, L.D.; Ngemoh F.A. & Earles S.W.E. (1992). Theoretical Modeling of Screw Tightening Operations, Proceedings of ASME European Joint Conference on Systems, Design and Analysis, 1992, vol. 1, pp.189-195.

Seneviratne, L.D.; Ngemoh, F.A.; Earles, S.W.E. & K. Althoefer (2001). Theoretical modelling of the self-tapping screw fastening process, Journal of Mechanical Engineering Science (IMechE), Part C, Vol. 215:pp.135-154, 2001.

Verlag, C.H. (1993). Plastics for Engineers: Materials, Properties and Applications, Munich Germany, 1993.

Visuwan, P. (1999). Monitoring Strategies for Self- Tapping Screw Insertion Systems, Ph.D.Thesis, University of London, 1999.

Walsh, R. A. (1993). Machining and Metal working handbook. McGraw-Hill, 1994.

Multilevel Intelligent Control of Mechatronical Technological Systems

Tugengold Andrei Kirillovich, Ryzhkin Anatoliy Andreevich,
Lukianov Evjeny Anatolievich & Wojciechowicz Boleslav

1. Introduction

Up-to-day mechatronical systems intended for production are provided with new properties to meet the following quality requirements: fast action, accuracy, reliability and others. The new properties should fit the varying technological conditions, adapt to the set goals, consider the mechatronical objects state, that is perform non-formalized or difficult to formalize tasks. In these cases they should implement decision making and control functions close to the human brain ones. These functions are realized with the help of modern information technologies (Makarov and Lokhin, 2000; Burns, 1997; Millan and Yeung, 2003).

Many mechatronical systems of technological application are aimed at quality production to the required demands unity. As a rule the control is effected on completion of the process. So in metal-cutting machine tools working under the unmanned machining the product quality is evaluated by its parameter control after either completing the whole operation or some stage of it. As to the working parts movement control it is evaluated only in case of errors of disagreement during machining. The instability in performing working parts control can be explained by stochastic character of machining, varying external factors, improper initial and real positions of the working parts and tools, parts state and the tool cutting edge (Millan K., Yeung 2003; Tugengold A.K. 2001). It is therefore necessary to evaluate the process as a whole and chiefly the degree of product quality parameters to obtain the right choice of alternative control options. Due to the varying set of conditions the latter is difficult to obtain and is a complicated task though it is quite possible by using intelligent control systems.

2. Intelligent Control System of Technological Equipment

By using mechatronical systems a new approach to software under intelligent control can be realized. Initially at the first stage of receiving information the intelligent control synthesizes a great variety of behavior alternatives and possible modifications of controlled process parameters compared with the set program. Then at the next process stage in several steps the control system limits to min the variability of different alternatives. The final stage is implemented as a rule with the parameters of machining corresponding to the last stage of a set program, that is target ones. It allows to reach the synergetic effect it technological process realization and provide the appropriate quality result.

The intelligent control system the structure of which is given in Fig.1 is based on decision making in machining a part in accordance with current environment. The sequence it system operations are as follows:

- evaluation of the control object state , technical process, its result;
- correction of criteria and limitations;
- simulation and replaning movement program;
- correction of working parts movement control.

This sequence is effected by the control system with multilevel hierarchical structure which has a set of levels typical for mechatronical systems with intelligent control: tactical – 1, coordinative – II and organizing – III.

Level I have traditional schematic working parts movement object control to implement technical process.

Its distinguishing feature is to execute operating corrective control within transition regimes based on real state drives and machining. The goal is to decrease the error of the tool path trajectory, consequently increase the part accuracy by using neural networks.

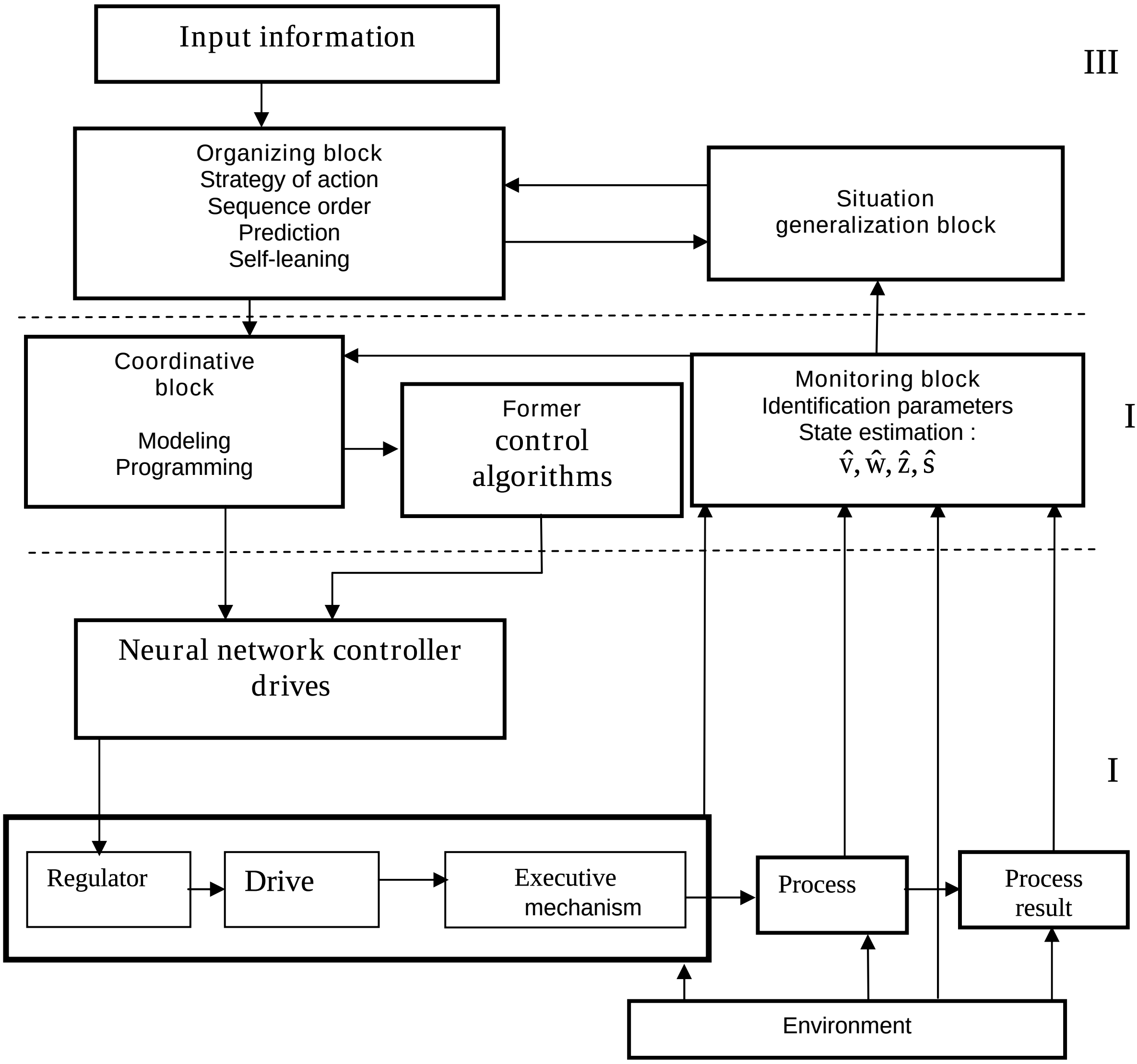


Figure 1. Structure of intelligent control system

Trajectory error minimization, occurring due to the dynamic process within the working parts speed change is included in to the intellectual control function at the tactical level. The preventing control is used for this reason, while the preventing signal is transmitted to the drive. The signal meaning depends upon the large number of the factors. (Many factors influence upon the signal meaning.) The main of them are the tool real position dynamic error, cutting force change, cutting edge variations, drive state.

The availability of priory information from intelligent control about current and future movement area parameters allows realizing a preventing control. To determine the control value position error information, difference of drives speeds from planned ones, cutting force evaluation is used. Forming a compensating control is performed by static three ply neural network with direct transmission signal. As efficiency criterion of correcting control for transient regimes the minimum criterion for space limited by ideal and real trajectories is taken. Fig. 2 shows tool movement trajectory without correction (Fig. 2a) and with correction (Fig. 2b). Max trajectory error is defined as the normal from ideal speed changes point to real movement trajectory and amounts as Δ_t=0.047 mm without correction and Δ_t=0.014 mm with correcting control. Thus application of neural network preventing correction allows to trajectory error in transient process by 60-80%.

Level II includes monitoring blocks, a coordinator and a former with corresponding. Monitoring block tasks are to provide information about space vector parameters necessary for decision making. The block identifies parameters under control of environment state, object control, technological process and its result, diagnose these states and evaluate them. The evaluation obtained by block expert system tells about state qualitative changes.

On the basis of these data the expert system of coordinator formulates decision-making about necessity of changing in working pasts movements in program and forming algorithms of their realization. At this level situation evaluation of qualitative system state necessary to achieve coordination control is implemented. Coordinative block executes situation simulation based on qualitative models describing object control functioning with regard to monitoring results.

It plans the sequence of working parts movement adequate to the current situation. Impact controllers at tactical level are formed based on accepted sequence and algorithms of control. The equipment control is implemented in accordance with chosen decisions and algorithms. To reduce errors in form and size of parts caused by dynamic errors in electromechanical drives a neural network controller for drive feeds is applied.

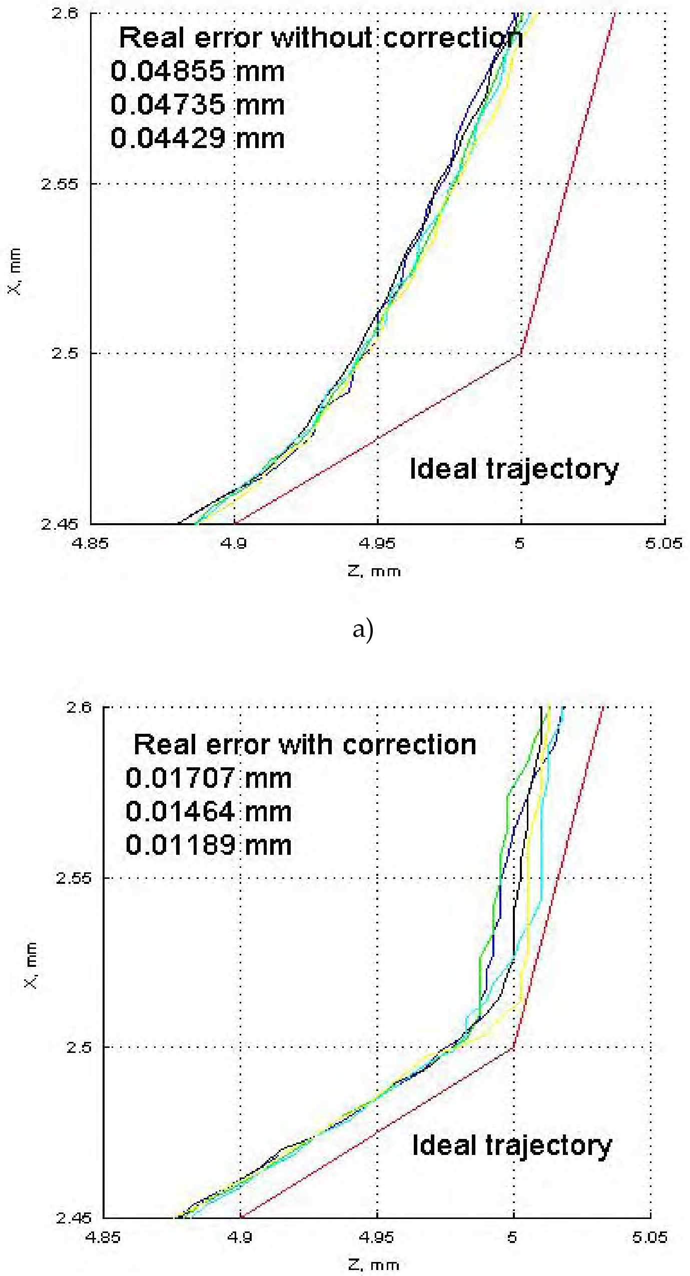
Level III (expert system of organizational block) executes action strategy choice of mechatronical object: rearranging the stages of action (compared with initial program) to achieve the main goal; current goals functions; corresponding to forming criteria bases and limitations.

New action strategy takes as its basis expert generalization of synthesized macro models of the current situation in technological process. The possibility of achieving each stages goals and target result is obtained by procedures of predicted evaluations.

Besides, level III includes the checks of achieved goals, learning methods with accumulation of knowledge about the character of processes (technological and within the object itself) and control models necessary to perform these purposes.

Generalization block implements information analysis, provided by monitoring block generalization procedure uses the image of numerical values of internal and external parameters and their balanced relation as quality notions as well as their group analysis. The goal is a consideration of a current set of parameters that might be suitable for any former situation or combination of states in order to establish any differences if they are any them make appropriate decisions in an organizing block. As an example of such decisions they may be such as decision about limited accuracy of machining next part; necessity of changing the passes number during machining the finest surfaces, etc.

At should be noted that the functions performed at Levels II, III cannot be formalized analytically since it is necessary to use not numerical but qualitative evaluations and notions. For processing, accumulation and utilization know ledge dynamic expert systems are applied working in real time scale and capable to improve mechatronical object behavior due to the in-built algorithms of learning and self-learning.



a)

b)

Figure 2. Dynamic trajectory error of servo drives. (a - without neuro networks control unit; b – with neuro networks control)

At tactical level I state vector of working parts of mechatronical object $\mathbf{x=x(t)}$, $\mathbf{t_H<t<t_K}$, where $\mathbf{t_H}$ is start time of the process or its stage, $\mathbf{t_K}$ is finish time of the process or its stage, becomes dependent not only on impact control vector **u**, but on dynamically varying properties of the elements taking part in the process and also on specific dynamic phenomena accompanying the process. Thus when cutting materials by machine tool mechatronical system the essential influence factors are instability of machined materials characteristics, tools, non-uniformity of machining allowances for parts; auto vibrations and others. And in mechatronical rolling systems in metallurgy there are changes in friction coefficient between rolls and rolling material, in stresses of deformed materials and others.

Mechatronical object parameters, the process, the process result and environment are difficult to measure; if impossible for some parameters. Often one can judge about them only by indirect route. For this reason monitoring block evaluates the whole system state according to parameters vectors to be measured: v – mechatronical object considers, **w** – process, **z** – process result, **s** – environment. This evaluation of technological system state is conducted by block expert system based on knowledge considering the professional operator's experience. Expert system solutions are not well defined state models **v**, **w**, **s**, **z** respectively and impact extents of different factors on process results.

Coordinative block based on data obtained from monitoring block (Level II) evaluates disagreement between real and planned object behavior, process and attained result. Controlling action model to perform executive mechanisms of planned behavior controlled coordinates, with the obtained state evaluation results can be presented by non-linear differential equations of the type.

$$u = A(q, p) + b(q, \dot{q}, \ldots, q^{(r-1)}, p),$$

where u - is m measuring control vector;

q is executive mechanisms controlled coordinates vector ;

$q^{(r)}$ is the later time derivative;

p is the n parameter measuring vector, evaluated as

$$p =< \hat{v}, \hat{w}, \hat{z}, \hat{s} > .$$

Fuzzy logics formalisms unstable unities form this equation mechanism. The 2nd and 3rd level of machine mechatronical as system the decision making process includes the following stages:

1. detail surface mixed by machine accuracy differentiation information sufficiency
2. evaluation according to the corresponding initial half-finished surface;
3. decision-making on insufficient information obtaining method during the machining;
4. Data and measurement result evaluations new stages of the controlling program for expediency evaluation decision procedures record;
5. determining factors evaluation to prevent possible prevailing influence upon the insufficient surface machining;
6. variants' strategic planning based upon the prognosis;
7. machining variants tactic planning;
8. controlling action model forming program stages correction;
9. passes results evaluation ; decision – making on the further surface machining;
10. decision on the final surface machining validity; results and possible ways to reduce the machining time.

Stated the 5 - 6 stages planning corresponding to the different level generalization. For instance, at the 5th stage decision – making search is effected generally, forming, new pass sequence to obtain the final surface without all the technological parameters indication , while at the 6th stage only the next pass with the error compensation variant workout is formed , that is the selected tactic task is solved in details.

3. Controlling Expert System Environment

At the designing of the environment fitting an all-level system control hierarchy of expert system, the following requirement execution was envisaged:

- structure formation unified method of the data and knowledge bases at a certain subject field;
- high – speed decision making, providing the expert system real-time work
- knowledge replenishment realization and self-education during the functioning period .

Besides at the expert system at the subject field under consideration has to work with the input information, having substantial uncertainty. That is why knowledge image models and decision making methods, based upon the unstable unities and unstable logics, are used.

Original expert system (Expert 2.0) environment contains neural network elements, where each separate "node" is an independent neural network responsible for the local task solving. The node being an element of the general neural net, receives signals (data, decisions) from the preceding nodes, effects their data analysis and depending upon the result activates the definite signals at its output.

Description space is formed to differentiate the situations (processes, events). At this very space the situations may have different parameters but belong to the same classes, occupying the definite area at the sign space. The situation can be referred to a definite class according to the input information set at the definite moment of time. Input signal analysis functions according to which neural node makes its decision, combine verification and summing up.

Neural nodes realization program allows the expert system to change the neural net structure and weight indices depending upon the accumulated experience – either confirmation or rejection the taken decision appropriation while the weight indices are consequently update. This corresponds to the program principle ability to self – educate. The initial educating set (input data – result) can be taken from theoretical reasons and expert qualified personal practical.

Conception base for the expert system program – module architecture design and conclusion (making) methods is a flexible structure of a program configured neural net.

Expert system environment is based upon the object – oriented programming principles and includes the following components:

- Project Manager executing all the system constituent components controlling functions;
- expertise manager, controlling expert system environment work during the expertise , as wall having expertise recording function;
- knowledge base structure editor aimed at structure creating and modification. Using subordinate objects (elements editor, connection editor, junction adjuster) it effects setting of the elements, forming the knowledge base structure .

- element editor for the forming structure setting
- node adjuster effecting educating function for logical conclusion making ;
- elements- any objects, effecting structured functions;
- knowledge – base node – elements producing any logical conclusion;
- elements responsible for the system adequate information supply.

Self – learning process proceeds as following. Expert system environment chooses one of the examples obtained during the execution process and being "Input data –result" set sends real data do the node receiver and lets the node make any conclusion. Then the obtained conclusion is compared to the example result and should they did not coincide, the "weight indices" correction of the given node is effected. Thus the base example skipping is effected up to the time the node starts making right decision for each example. At this stage the self – learning process is completed.

She system possibility to self – educate using the examples, knowledge base real time correction refer to the given method advantages. That is provided due to the fact, that each expertise effected, adds one more example to the accumulated example base, which are used for the system self – learning, précising the function of the situation area division. The education method based upon the input example has the following advantage, as it does not require expert rule replenishment to make the conclusion, as this work is effected by the environment independently.

Designed expert system environment is realized in a form of a program, written in object – oriented language Delphi 5.0. OLE technology usage simplifies the change implementation process into the software and allows the system to develop gradually without total overwork, if any initial requirements substantial changes occur.

4. Action Results Prognosis

Machining resulting accuracy is characterized by the set of the output parameters $y_1, ..., y_n$. Vector parameters $\mathbf{y} = \{y_1, ..., y_n\}$ changes occurs under the influence of the planned action sequence with the object and under the influence of the many factors, those record complexity and stochastic makes consider the parameter changes as any random function $\mathbf{y}(g,t)$, $t \in T$, $g \in Q$.

There Q – that is random events multitude;

T – pass multitude (or time or any determined changed parameter).

At the probability space Q(F,P), where F is a Q, P sub multitude algebra, P is a probable measure, the random function:

$$y(t) = \{y_j(t)\}_{j=0}^{n},$$

may be considered as general model of the parameter change process. The peculiarities of the parameter change process are explicit inflexibility and stochastic.

The action result prognosis task under the inflexibility of the random process is closely connected with the random function extrapolation, that is classically revealed the following way. There are giving, $\mathbf{z_w(t)}$, $\mathbf{t \in T}$, and "non – observed" random process $\mathbf{z(t)}$, which is statistically connected with $\mathbf{z_w(t)}$. At the $\mathbf{t \in T_p}$, moment, where $\mathbf{t = \{t_0, ..., t_k\}}$ and $\mathbf{T_p \in T}$, the "observed" process $\mathbf{z_w(t)}$ is known. It is necessary to assess the "non-observed" random function $\mathbf{W(t)}$ for the future time moment $\mathbf{t = t_{k+1}}$, $\mathbf{t_{k+1} \in T \,|\, T_p}$, using the known $\mathbf{z_w(t)}$ realization. At the control – diagnostic interval the prognostic process may be observed under the measurement error **e(t)** condition. The probable assessment of the planned action at the next stop can be taken as one of the variants. The random process model under the half uncertainty condition and at expert process evaluation can be presented as following

$$y_i(t) = a_{i,j}\cdot\{f_i(t)\}, \quad i=0, ..., n \tag{1}$$

where $a_{i,j}$ - random value; $\{f_i(t)\}_{j=0}^{m}$ - determined pass function.

The model treats the random process division on the determined base, characterizing the trend. Degree, exponential and other function may be used as a basis (as cutting, e. g.).

The prognosis parameter is influenced essentially by the random character process as cutting, for instance the fluctuating influence of which cam be treated as convertible changes. Taking into consideration all the stabilizing factors the random process of the parameter changes may be approximated by the equation

$$y(t) = Y(t) + F(t), \tag{2}$$

where Y(t) – unstable; random process of the parameter changes;

F(t) is a stable, random fluctuation process, initiated by inner and external action.

Since the random process parameter value z(t) measured differs from genuine y(t) value for some random value e(t) – that is the measurement error, then

$$z(t) = y(t) + e(t),\ t\in T_p\in T.$$

The true measurement evaluation, with the occasional error, may be obtained by well-known math statistics procedure, presuming e(t)'s independence, uniformity normal conditions. The measurement process is presumed to be discreet and continuous. The math classical statistical methods form the problem solving algorithm basis – that is, the least squares, maximal to life, optimal filters, etc.

From the point of realizations simplicity the optimal filtration method usage is of great interest. The task peculiarity lets to be limited by the linear optimal filters aimed at unstable sequences extrapolation.

The multi - purpose use Kalman - Bucy filter is easily realized on a computer due to the recurrent representation form. The results obtained are optimal in the approximate square meaning being competent effective and a unbiassed estimator. The random process under consideration may be described by the difference equations of the type

$$x_t = F(t, t-1)\cdot x_{t-1}, \tag{3}$$

where F(t, t – 1) is trans missing matrix, characterizing $\mathbf{x}_t$ structure, while the observation should meet the

$$z_t = P_t\cdot x_t + e_t, \tag{4}$$

relation requirement where P_t - is the limitation matrix of x_t observation; e_t - is the observation error.

To solve the y(t) prognosis problem solving one should bring the (1) model in conformity with (3) expression. The transformations are effected on the state space expanding basis, that is new coordinates introduction into the technical task state. Matrix F(t, t – 1), P_t and vector x_t structure, occurs as a result of y(t) model transformation.

The well – known relations, determining Kalman - Bucy filter and transformed to the state of form the 3 4th equation solving:

$$X_{t+1|t} = X_{t|t-1} + W_t(z_t - P_t \cdot X_{t|t-1}), \quad (5)$$

$$W_t = K_{t|t-1} \cdot P^T(P_t \cdot K_{t|t-1} \cdot P^T_t + R_t), \quad (6)$$

$$K_{t+1|t} = K_{t|t-1} - W_t \cdot P_t \cdot K_{t|t-1}.$$

Where: Xt+1/t is a prognosis observation evaluation X (relative mat expectation X) for the time moment t+1;

Xt/t-1 is the moment t X prognosis evaluation;
Kt/t-1 is a Xt/t-1 vector component covariant matrix;
Kt+1/t is Xt+1/t vector component covariant matrix;
Wt is a corresponding filter transmission matrix;
Wt is a y(t) observation result at the t moment;
R is a e(t) measurement error covariant matrix.

As a result of the 5 6 correlation prognosis calculation the

$$M[X(t_k + 1) | t_k] \text{ и } D[X(t_k + 1) | t_k],$$

meaning is obtained as experience math expectation evaluation and X dispersion.

5. Conclusion

The structure under consideration as a whole meets the following intellectual controlling systems requirements:

- Intellectual functions plenitude due to the realized procedures of the artificial intellect strategic, coordinating and executive level methods.
- Methods actions directions, connected with given task completion and the decision – making search.
- Cardinal perceived information stream expanding (due to the very object and environment evaluation) and variety alternatives of behaviors.
- Planning and controlling at the decision – mating system hierarchy.
- Reciprocal subordination of the regulation task, action planning and behavior strategy choice as a form of parallelly acting reverse connection. System state image synthesis intellectuality due to the expert evaluation; image level ranging, situation generalization and prognosis.
- Evaluation of the factors, having no data obtained by the direct measurement; experience basis presumption formation.
- Controlling system function preserving at the half or total intellectual loss, with any process realization quality detriment.

Thus, intellectual controlling realization at the upper decision making level (methods and order to obtained the set detail parameters), combined with the drive neural network controlling lets gradually increase detail production accuracy and quality.
The system adequate transformation under the changing technological conditions is supplied by this structure. The intellectual system for complex mechatronical objects controlling under the uncertain conditions constructing principles were taken as a basis at the design level, including expert system technologies, fuzzy logic, neural network structures.

Since the monitoring block as a part of a subsystem of intelligent control is considered to be important for research it is suggested that the authors do further research and publishing in this field.

6. References

Burns R. (1997). Intelligent manufacturing, Aircraft Engineering and Aerospace Technology Volume 69 · Number 5 · 1997 · pp. 440-446 © MCB University Press · ISSN 0002-2667

F. Meziane, S. Vadera, K. Kobbacy, N. Proudlove (2000). Intelligent systems in manufacturing: current developments and future prospects, Integrated Manufacturing System 11/4 (2000), p.218-238, MCB University Press [ISSN 0957-6061]

Makarov M., Lokhin V. (2000). Artificial Intelligence and complex Objects Control/ Ed. by 1. The Edwin Mellen Press. Lewiston, NY, 2000.

Millan K., Yeung (2003). Intelligent process-planning system or optimal CNC programming - a step towards complete automation of CNC programming, Integrated Manufacturing Systems 14/7 [2003] 593-598 MCB UP Limited [ISSN 0957-6061]

Tugengold A.K. (2001). The principle of intelligence control in mechatronical systems.//6-th International conference on advanced mechanical engineering & technology, AMTECH-2001, vol.3, Bulgaria, p.20-25.

A Robot System for High Quality Belt Grinding and Polishing Processes

Bernd Kuhlenkoetter & Xiang Zhang

1. Introduction

Grinding and polishing are standard operations in the material processing. They are automated with the help of industrial robots in order to relieve human from laborious tasks and unpleasant environments and elevate the profitability of production nowadays, specially in the sanitary fitting industry. However, the systems known at present are adapted quite costly to other part geometries and operation cycles and are therefore economically applicable only for large batch sizes. Frequent changing of the robot program and system parameters also increase the cycle time and the cost of the whole manufacturing process. This problem will be more outstanding when the operating surface is a free-formed surface with very complicated geometrical shapes.

This paper describes a robot system which is specially developed for grinding and polishing free-formed surfaces with high quality and high efficiency due to the linkage of innovative robot technology, simulation technology and artificial intelligence methods. The robot system combines some different levels of automation, manual operation, partial automation and full automation. Highly labor-concentrated jobs will be done full automatically, for example detecting errors on the workpiece surface within the manufacturing process. Some modules are partial automated to help the operator plan the schedule and design the program. For operations that rarely occur a manual interference will be sufficient. A reasonable remaining of the manual work can keep the low costs of the robot system which is also acceptable for the enterprises. The robot system is flexible and suitable for the manufacturing of small and medium batch size, which is a difficult task for current robot-controlled grinding and polishing systems.

The automation in this robot system focuses on two parts. One is the quality assurance and the other is the process model that suggests the choice of the technical parameters for a high quality processing. To obtain a high quality surface, the robot system should have more accurate motion programs in the process optimizing. The new software system is developed exactly for this purpose. The software helps the operator design the grinding and polishing paths precisely and efficiently with CAD data of the grinding/polishing surface. In addition, the simulation of the grinding process will assist the operator to get more reasonable manufacturing paths and adopt the optimized parameters to the manufacturing process in order to obtain a high quality surface. The process model is specially for the grinding process that studies the relation between the various grinding parameters and the final amount of materials removal from the workpiece surface. The result of the process model makes the simulation of the grinding process possible. The operator can analyze qualitatively what the surface is like before the workpiece is really ground. This can direct the operator to modify the grinding path in advance and therefore reduce the rejection rate of the production processes.

In the next section, the grinding and polishing using industrial robots are briefly introduced. The system used and the key technologies in the system will be described in detail in section 3. In section 4 we give an account of our work to simulate the belt grinding process which is a very important part in the whole system. The sub-system of defect detection and classification is introduced in the section 5. We end with a summary and some main future plans of the system.

2. Grinding and Polishing in the Manufacturing Chain

It is the aim of grinding and polishing processing to improve the forming and dimensional accuracy as well as the surface finishing. Both processes play an important role as they are at the end of the product chain where processing errors lead to high rejection rates of the parts.

Specially in the sanitary fitting industry grinding and polishing are widely used to manufacture a free-form surface. Semi-finished workpieces are normally manufactured by casting. Through the subsequent grinding and polishing a high-quality shining surface is produced. In this case, the dimensional accuracy of the workpiece is only playing a secondary role. Fig. 1. illustrates five phases of producing a water tap fitting from the rough workpiece to the final product.

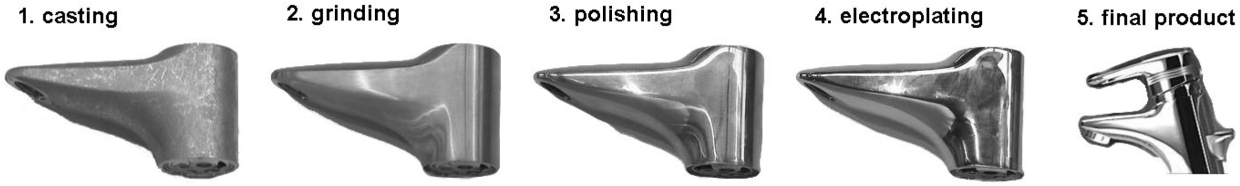


Figure 1. Producing steps in the manufacture of fittings. (1) casting, (2) grinding, (3) polishing, (4) electroplating, (5) end product

Traditionally grinding and polishing processes of such free-form surfaces are done by manpower. A worker holds the workpiece, feeds it towards the grinding/polishing machine and moves the workpiece along the paths that are decided subconsciously by experience, see left side of Fig. 2. In this manufacturing process, the worker has to suffer from the unpleasant environment, e.g. dirty air and loud noise. Additionally the job is hard and monotonous work so that the people cannot concentrate on it for very long time, so they cannot guarantee constant quality of the product. Nowadays, industrial robots are introduced into the manufacture of free-form surface grinding and polishing to minimize the costs while optimizing the quality. The robot, replacing manpower, holds the workpiece and moves along the paths that are predefined by technicians, see right side of the Fig. 2. To implement this process using industrial robots, the most important is to plan the grinding/polishing paths and generate the robot programs. Another practical problem to be handled is the occurring of errors in the previous casting phase. The casting process is characterized by high resulting dimensional and geometrical tolerances as well as quality fluctuations like blowholes and pores. These strongly variable starting conditions of grinding lead to unprofitable rejection rates and a very high manual testing procedures in the automated grinding and polishing processing. What is even more difficult for the realization of an automated solution is the fact that errors are only detectable after a part of the fine processing has been done and that the sensitive and strongly shining surfaces are difficult to measure.

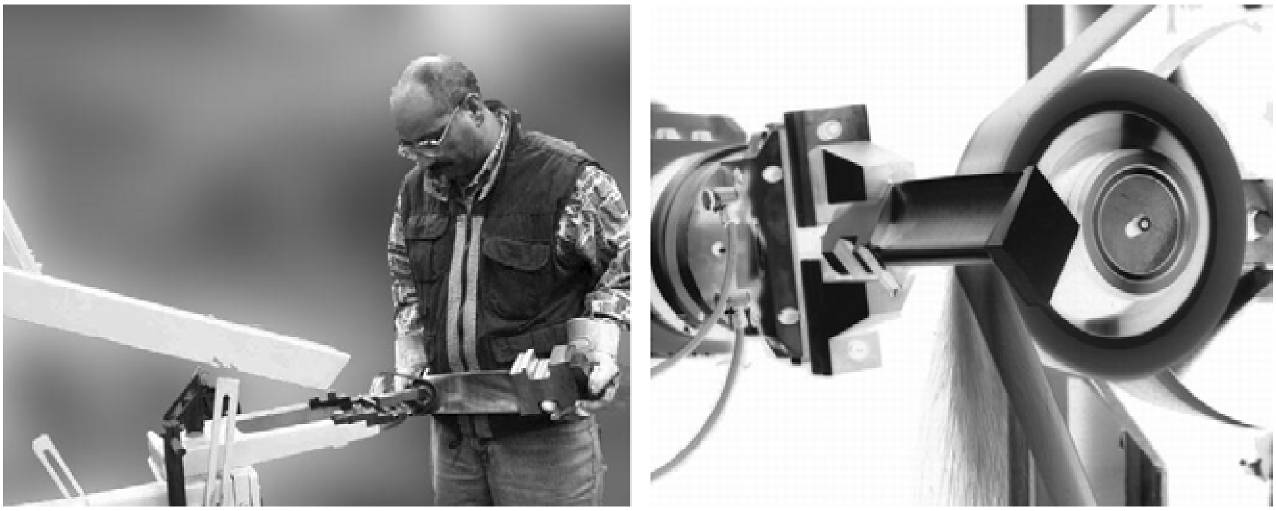


Figure 2. Manual grinding process and robot aided grinding process

The robot aided automation solutions known at present in the fields of grinding and polishing are especially and successfully used in the sanitary fitting industry. The profitability of these systems has been granted in the past years despite high national wage costs. With the advancing globalization of the markets, it is now faced with a high competition from low-wage countries where the grinding and polishing is done manually. Besides the grinding and polishing processes, the subsequent steps of manufacturing are also threatened to be shifted abroad. The cost of the total manufacture process should be reduced to face global competition. However, the high time and cost requirements for programming and optimizing have a negative effect on the profitability of industrial robot aided grinding and polishing cells (Schueppstuhl, 2003). Compared to conventional robot tasks these high requirements result from the clearly more complex, comprehensive and more accurate motion programs.

These requirements have of course an even more negative influence if new programming or adaptations become frequently necessary (Cabaravdic, 2003). The main reasons for this are an unfavourable ratio of the batch size to the variety of modifications and the occurring fluctuations in the process due to workpiece tolerances, and other errors in the upstream manufacturing process. The required level of automation of the procedures depends on the occurring frequency of the necessary optimizing work. For rarely occurring process malfunctions a manual interference is sufficient, while for more frequent occurrences a full automation must be realized. The techniques to be developed for a manual use differ considerably from a fully automated one. In manual procedures the focus lies on the efficient interaction with the operator whereas a full automation needs the development and integration of a complex measuring method, a data processing and process control.

3. Innovative Robot System

One aspect of the project aims at further developing the process specific offline-programming system, as approved in practice, in order to reach a higher efficiency than manual programming and optimizing procedure. The nowadays systems are designed for a universal use and similar to complex 3D CAD-systems what regards their layout and operation. Processes that do not need an extra path or parameter optimizing, such as palletizing, assembling or varnishing can be programmed efficiently with such software systems by high-qualified engineers and technicians in the planning department. However, there are no appropriate tools available for the grinding and polishing to optimize the phase directly at the robot cell.

The use of a conventional offline-programming system in grinding or polishing usually fails because of its complexity in connection with an unsuited qualification of the operators as well as lack of process specific functions. Therefore a high demand for according developments arose in this context. The intended system should adapted itself to small batch sizes and the case in which lots of modifications must often be carried out. The operator of the robot is in the centre of the decision-taking and should be given PC-based deciding guidance for the next step and according tools for an efficient program optimizing.

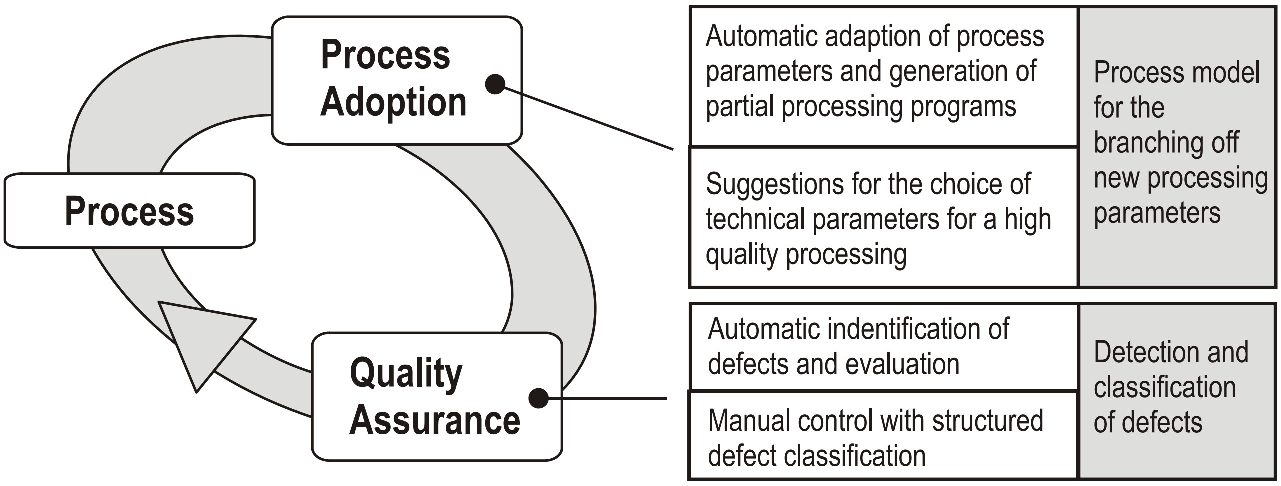

Process Adoption
Process
Quality Assurance
Automatic adaption of process parameters and generation of partial processing programs
Suggestions for the choice of technical parameters for a high quality processing
Process model for the branching off new processing parameters
Automatic indentification of defects and evaluation
Manual control with structured defect classification
Detection and classification of defects

Figure 3. Central aspects of the system

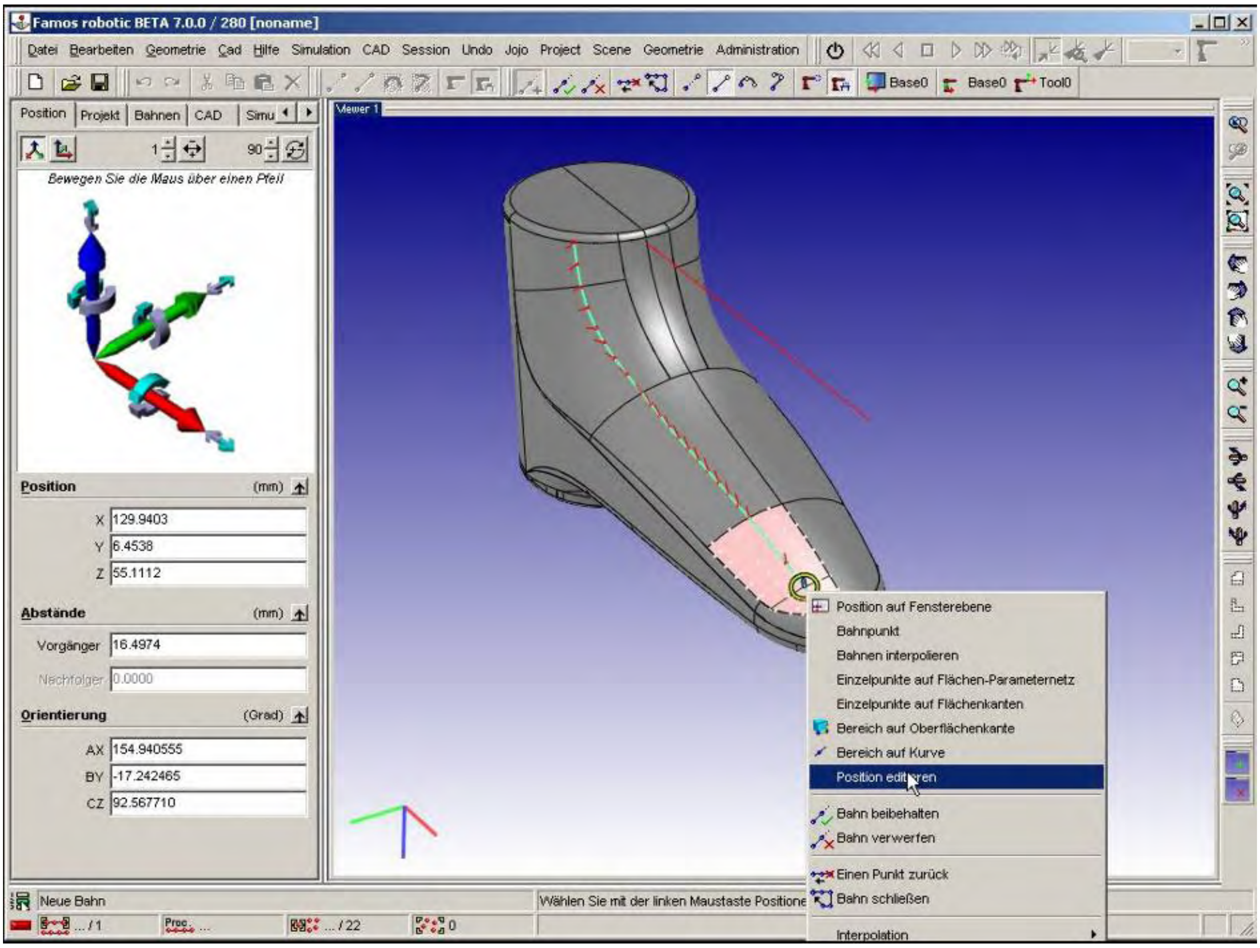
Another aspect lies in the consideration of disturbing influences that frequently occur and which must be detected and compensated automatically. In practice, the most common disturbing influences are the existing defects on the semi-product from the casting phase or caused directly by the grinding processes.

Therefore, the error detection and classification as well as the deduction of parameter optimizing strategies through measuring methods and process control must be realized fully automatically, see Fig. 3.

A software system has been developed for a workshop programming of robot systems that provides an intuitively operable graphic 3D-user interface. The software provides process specific optimizing tools, specially for grinding and polishing. The software accepts the CAD model of the workpiece or its surface as input and helps the operator to generate the grinding or polishing path conveniently and practically. The software can generate the according robot programs for users.

With this software, the time required to design the paths is greatly reduced compared to the universal off-line programming system. It is very suitable for small or medium batch size manufacture and flexible to the requirements of frequent modifications of workpieces or paths. Fig. 4. shows the simple generating of grinding paths on the surface of the workpiece. Additionally, the software is extended by an adaptive consulting centre for the allocation of errors, causes of errors and compensation strategies and an internet connected process-know-how-database.

This software fills the gap between the multi-functional, however complex, offline-programming systems used in the planning department and the inefficient possibilities of the robot control used by the operator for the program optimizing.



Bewegen Sie die Maus über einen Pfeil

Position (mm)

X 129.9403

Y 6.4538

Z 55.1112

Abstände (mm)

Vorgänger 16.4974

Nachfolger 0.0000

Orientierung (Grad)

AX 154.940555

BY -17.242465

CZ 92.567710

Neue Bahn

Wählen Sie mit der linken Maustaste Positione

Figure 4. User-orientated offline-programming and simulating system (Carat Robotic Innovation, 2004)

A higher tolerance towards changing starting qualities of the workpiece is achieved by combining image processing/measuring systems, grinding and polishing process models, and adaptive control technique and intelligent software components. The "see and evaluate" of processing errors on high shining surfaces is even difficult for the untrained human eye.

Furthermore, errors in the workpiece material in the process chain of rough grinding, finish grinding and polishing can be often detected only after a part of the processing has been done, or at the end.

Therefore, another requirement has arisen for developing a fully automatic working process chain for the industrial robot-aided grinding and polishing based on the measurements of an image processing system so that an optimum surface quality is obtained despite fluctuating starting conditions. Fig. 5. below illustrates the working flow of the automatic detection and classification system. The central hardware is a closed vision system that can provide suitable illumination and take photos of the workpiece surface. The software of the system is responsible for finding the defect position and classifying the defects by analysing the grayscale photos taken by the vision system.

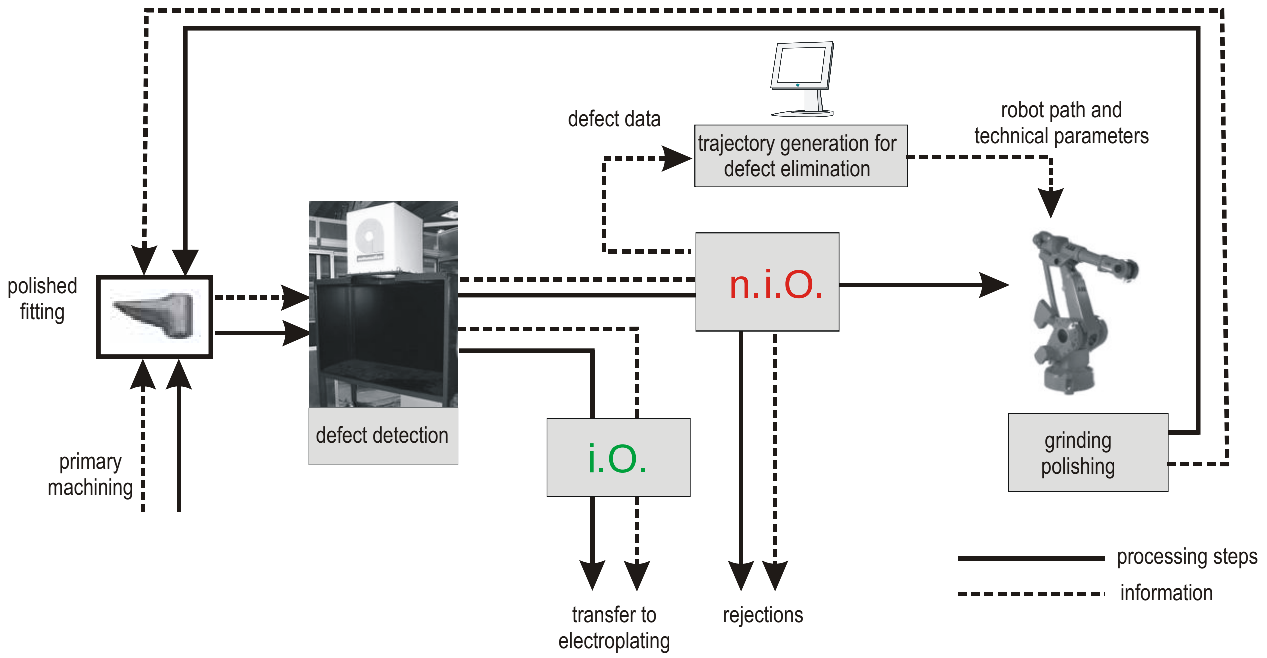


Figure 5. Flow chart of the fully automated process chain

After polishing, the workpiece is transferred to the vision system to evaluate. The workpiece will go to the next manufacturing step when no defects are found on the surface.

A workpiece that has remediless defects will be rejected or discarded while the rest is ground and polished according to the individual defect data. The retouched workpieces are fed into the automatic vision system again until they are either accepted as qualified or discarded as defectives. The compensation machining process for detected errors on the surface is already realized in the system.

4. Simulation of Belt Grinding Processes

The operator can design the grinding and polishing path easily by using the software as shown in Fig. 4. However, it is a highly experience-related job to decide the paths on the workpiece surface. A skilled operator generates the paths and according programs that result in a good surface quality and low rejection rate while an untrained operator reaches a high rejection rate that damages the profitability of the robot system. The individual operator produces inconsistent standard of design and causing a various surface quality of the workpieces. Therefore, the belt grinding process and robot system should be studied in detail with a process simulation being necessary at the same time. Both path planning and parameter selection of the robot system will benefit from the results of the simulation afterwards.

4.1 Introduction of Belt Grinding Processes

Belt grinding is a machining process with a geometric indeterminate cutting edge. The grinding belt (cutting tool) consists of coated abrasives and is attached around at least two rotating wheels.

The part to be ground is pressed onto one of these wheels, which is the contact wheel. Fig. 6. shows the belt grinding process. The material is cut off under non-permanent touch between workpiece and abrasives.

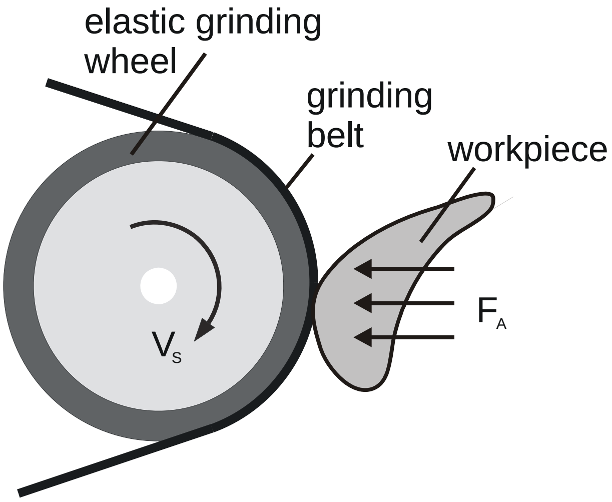


Figure 6. Belt grinding process with elastic wheel

The variant of the belt grinding with elastically deformable contact wheel is especially suitable for the finishing of free-form surfaces because the elastic contact wheels allow a flexible processing through their adaptation to the part surface. Compared to other finishing processes, belt grinding with elastic contact wheel is characterized by a higher removal rate with comparable surface quality. The most important advantages of this process variant are:

- The elasticity of the contact wheel leads to a better form adaptation to the surface of the workpiece compared to a rigid contact wheel and therefore produces a higher quality of sculptured surfaces.
- The compensation of the infeed and orientation error of workpieces or tools is possible to some degree.
- During a long phase the grinding belt wear factor is approximately constant. Thus the belt wear does not affect the machining process in this phase.

4.2 Removals in Belt Grinding Processes

The most important point in simulation is to calculate the removals from the workpiece surface at a discrete point of time. The simulation of the grinding process is more difficult than those precise operating processes, e.g. milling, because the removal of the surface cannot be obtained directly by calculating the swept volume through CAD data and tool shape. The cutting tool in belt grinding is the grinding belt that is coated by hard abrasives. In machining, the grinding wheel rotates and the belt rubs and strikes the workpiece surface.

Because shape and distribution of the abrasives on the belt are non-uniform and rather disordered, the belt grinding processes are considered as a cutting processes with an indeterminate cutting edge. Another problem is the elasticity of the tool (grinding wheel) that can cause a strong force variation between the grinding wheel and the workpiece. Calculating removals is therefore not a purely geometric computation, but a quite experience-based process. Besides the material of grinding belt, quite several other parameters can simultaneously work on the final removal from the workpiece surface, for example elasticity of grinding wheel, temperature and so on. The removal from the workpiece surface is a function of a group of factors.

To model this, Hammann presented a linear experiential formula (Hammann, 1998; Schueppstuhl, 2003):

$$r = K_A . k_t . \frac{V_b}{V_w . l_w} . F_A \qquad (1)$$

where γ is the removal; K_A, a combination constant of some static parameters; k_t, the grinding belt wear factor; V_b, the grinding velocity; V_w, the workpiece moving velocity; l_w, the length of the grinding area; F_A, the acting force. Using this model, one can do many experiments making only one factor variable and all other factors unchanged at the same time. Then, the experimental results can be combined to determine the coefficients in the model. The parameters in the model are all one-valued which means all inputs of the model are indicated by only one value, for example the acting force F_A is the global force between the workpiece and grinding wheel. The global grinding model is sufficient to the operating of simple shapes. But it is obvious that they are incapable to free-form surface grinding because not only the total removal but the local removals distribution (removals at small sub-area) are neccessary to be known.

Removals distribution results from the contact force distribution in the grinding process. The detailed local force distribution information (not only the global force FA) should be obtained before the removals distributions are considered. Normally, the Finite Element Method (FEM) is the traditional way to calculate the contact forces according to the initial contact situation. The FEM deals with this contact problem as an ideal Signorini contact problem (Krause, 2001) which solves an optimization problem by a contact energy minimization principle. Considering the process locally, the surface of the treated workpiece is split into a number of "finite elements" (Hammann, 1998; Schueppstuhl, 2003), assuming constant distribution of the contact pressure and cutting speed at each element. Hence, the contact pressure must be calculated for each element. The overall contact pressure distribution for one contact situation is then given by all local element pressures. Local removals can be calculated through a multi-factor statistical analysis in the second modeling step. Fig. 7. illustrates the framework of the process model for free-form surface grinding. First of all, the geometry and elasticity information are constructed as the initial contact situation which is the input of the FEM model. Then, FEM works out the distribution of force between the workpiece and the grinding wheel. Finally, the distribution of force together with other process parameters are given into the local multifactor analysis model to calculate removals distribution.

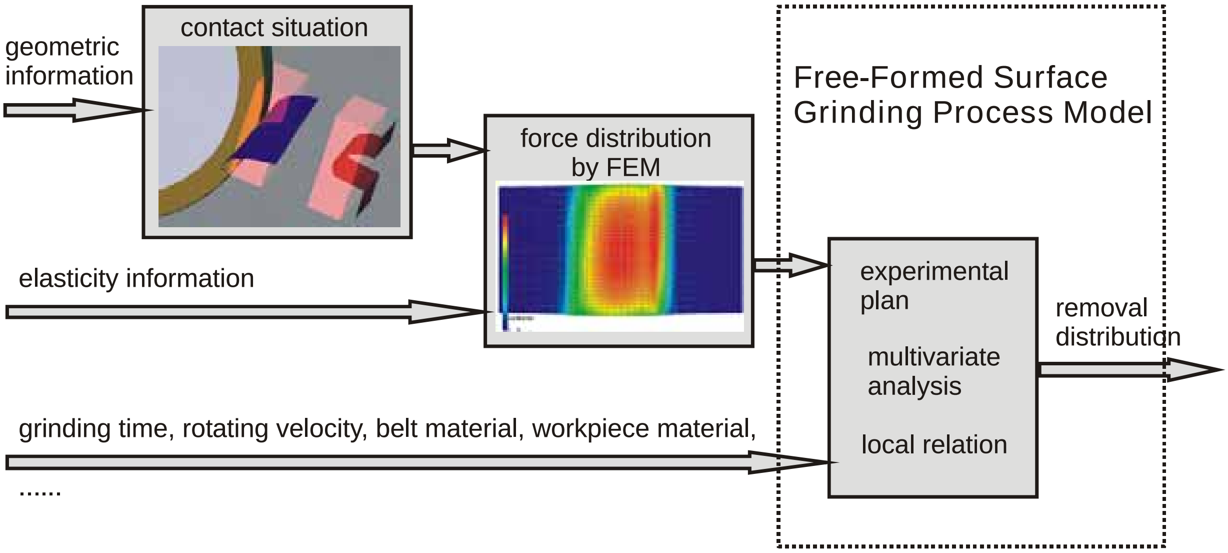


Figure 7. Process model for free-formed surface

4.3 Simulation of Robot Controlled Belt Grinding Processes

Fig. 8. is a flow chart of simulation processes (Schueppstuhl, 2003; Kreis& Kneupner, 2001a, 2001b). The planned path is known at first and the grinding process is divided into some discrete time intervals. At any time point, the initial contact situation can be reached by the current CAD model and the path. The next step is to calculate the force distribution and then get the removals distribution by the process model. The current CAD model is updated and moves to the next time interval until the path end. In this simulation framework there are four important parts:

- Initial contact situation modeling
- Force distribution calculating
- Process model
- Workpiece model

Additionally, a direct control system should be implemented if an on-line control of the robot is required (Kneupner, 2002; 2004).

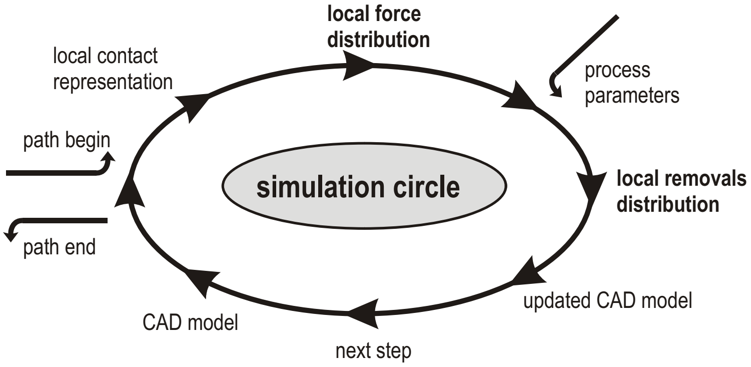


Figure 8. Simulation circulation

While simulating, the removal is calculated by the process model. Especially for surfaces with a small radius, the change of the surface can be dramatic. Therefore it is not possible to calculate a complex geometry using a so-called swept volume which interpolates the removal between two different removal calculations. Instead, the shape of the workpiece is changed directly after one removal calculation and the next calculation is based on the new geometry. An important effect which should be taken into account in simulation processes is the so-called “cut loose” effect (Kneupner, 2004). It means that the tool will be not in contact with workpiece any more after a specific time of grinding if the workpiece is not moved because all material in between will be worn out. While this happens, the actual removal rate will slow down. It is clear that such a process can only be simulated if the workpiece model is updated after a removal calculation. Thus, the time interval should be small enough to neglect the decreasing removal rate because of the “cut loose” effect. Moreover, a fast calculation is essential for practical reasons. The cycle time is driven by an external cycle time. As it is possible to calculate the position of the tool in relation to the workpiece with a high accuracy within the robot interpolation cycle time, we use a multiple interpolation time cycle. This is an amount of time small enough for neglecting the shape-change within a calculation. The Height Model is put forward in our project to describe the initial contact situation specially considering the characteristics of belt grinding process, assuming that workpieces for grinding are idealistically hard and stiff without deformation and that the grinding wheel is made of soft material with a known elasticity, which at the same time is also a prerequisite of the Signorini contact problem. So when the grinding wheel contacts the workpiece, it deforms according to the geometry of the workpiece surface and actual infeed. To describe such a situation, Height Model divides the contact area into a mesh where then the initial contact situation of grinding wheel can be encoded as a group of so-called Heights, which are intervals between the base plane and deformed surface of the grinding wheel at all mesh points (see the right part of Fig. 9.). The base plane is always vertical to the infeed direction and has invariable distance to the wheel center. The normal vectors of all contact points on the deformed surface are also recorded for later use.

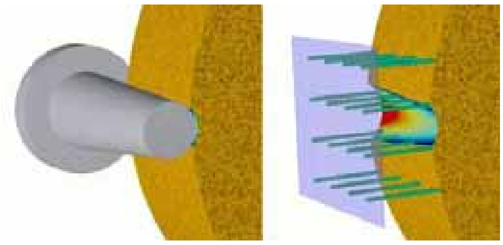


Figure 9. Height Model

The Height Model is actually a discrete description of an initial contact situation in which mesh size is a control factor for different precision requirement. In this way, each contact situation can be described by a heights matrix H, which has the following form:

$$H = \begin{pmatrix} h_{11} & h_{12} & \cdots & h_{1n} \\ h_{21} & h_{22} & \cdots & h_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ h_{m1} & h_{m2} & \cdots & h_{mn} \end{pmatrix}$$

where m, n is the mesh size in two directions.

The contact situation between workpiece and tool can change very fast during the processing of sculptured surfaces, causing a strong removal variation. This fact asks for quick calculation of force distribution in real time. The traditional way to calculate the force distribution is resorting to the FEM that consider the contact problem as an ideal Signorini problem. Blum and Suttmeier (Blum et al. 2000; 2003; Suttmeier, 2001) worked out a FEM model. The FEM model has to achieve the optimization solution through iterating steps each time when a new contact situation is presented. Thus, it is very time-consuming. Although the model used an optimized mesh discretization, it still requires about 15 minutes for calculating one contact situation. This is far away from the demand for real time simulation. The force distribution calculation is the practical bottleneck of the real-time simulation. To meet this, a new force distribution calculation model (Zhang et al., 2005a; 2005b) is worked out to accelerate the calculation. The idea is that the neural network or Support Vector Machine serves as an approximation of the FEM model to learn the non-linear mapping from the initial contact situation to force distribution. The expensive iteration process is shifted in the training process of the model, but not in run time resulting in a dramatic reduction of the calculating time which makes the real time simulation of the grinding processes possible.

After the local forces are known, it is time to decide the parameters in the process model. The experiments and previous research (Koenig, 1996; Meyerhoff, 1998; Hammann, 1998; Schueppstuhl, 2003) showed that an overall mathematical description of the belt grinding process is not possible because a complete list of the influential factors cannot be determined exactly.

In the process model, only eight parameters are selected out which are:

1. force floc (N/mm2)
2. rotating rate of grinding belt vb(m/s)
3. grinding time ts (s)
4. local radius of the workpiece rloc (mm)
5. the grain size of grinding belt kb
6. belt tension fb (N/mm2)
7. contact length lc (mm)
8. material of the workpiece mw

The influential factors of belt grinding can be determined only statistically due to the geometrically indeterminate cutting edge. The parameter values must be in a reasonable range in order to predict the removal precisely enough.

Statistical design of experiments is applied for the modeling of local relations. An essential aspect of the statistical design of experiments is the fact that several factors are varied simultaneously from a single experiment to the next one. In order to implement that, a experimental plan is used by the statistical design of experiment. With the help of such an experimental plan, more complex relations can be modeled. In our case a full factorial experimental plan with $2^8 = 256$ single experiments is used for all the 8 influential factors on two factor levels.

The local removals as the response variables are measured after every individual experiment using a sensor device. The experiment data are listed in the Table 1.

The influence of wear is minimized by the use of wear resistant grinding belts. A heat influence can be minimized through a suitable selection of grinding times. The grinding time for every individual experiment must not be too long so that the grinding belt is not heated up. This experimental plan has been done only partially and is still in work.

	f_{loc} (N/mm^2)	v_b (m/s)	t_s (s)	r_{loc} (mm)	k_b	f_b (N/mm^2)	l_c (mm)	m_w
1	0.01	10	5	30	P60	0.1	5	1.0037
2	0.5	30	15	200	P120	0.4	50	brass

Table 1. The technical data for all parameters

4.4 Simulation Examples

Four parameters are used to simplify the definition of the workpiece surface, the infeed x, the local curvature (the reciprocal of the radius) in y and z direction and the turning angle. The simulation results can imply how this local parameters act on the local removals. Fig. 10(a). illustrates the simulation result with different infeeds. The length of the pillar on the workpiece indicates the grinding infeed and colors on the workpiece surface denote the local removal distribution. The infeed starts with zero at the beginning of the path and reaches the maximum at the end of the path. The removal is growing when the infeed increases. This is because a bigger infeed leads to a larger deformation of the elastic grinding wheel and bigger acting forces. Fig. 10(b). illustrates the simulation result with different local curvatures in y direction. The length of the pillar on the workpiece indicates the local radius of the workpiece surface. The green pillar means the local surface is concave and the red pillar means the local surface is convex. The infeeds are the same along the grinding path. It shows that the convex part has a bigger removal than that of the concave part which conforms to the practical experiments. Fig. 10(c). is the result with different local curvatures in z direction. Fig. 10(d). is the simulation result of grinding along a simple equal-infeed path. The workpiece has a small turning angle that changes the local contact situation between the workpiece and grinding wheel, so the removals are also not distributed uniformly.

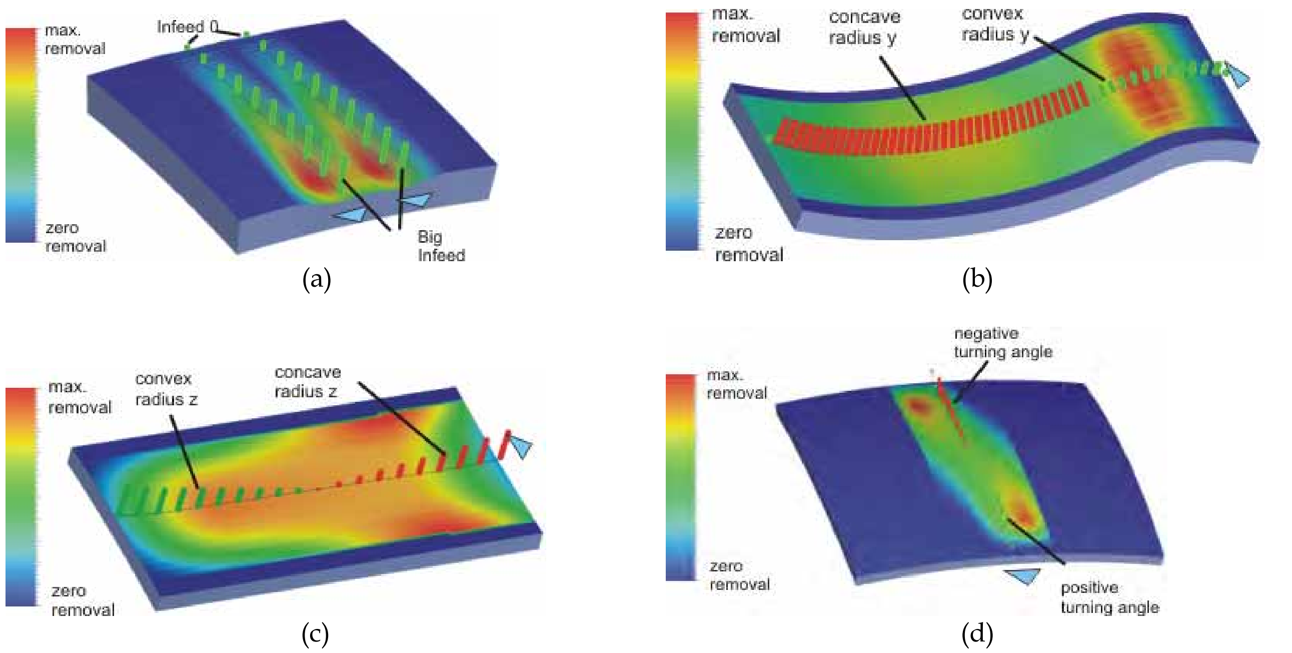


Figure 10. Simulation results (a) with different infeeds, (b) with different longitude local radii, (c) with different latitude local radius, (d) with turning angle (Schueppstuhl, 2003)

5. Defects Detection and Classification on Workpiece Surface

As mentioned in the section 3, another troublesome work is to find and identify the defects on the surfaces that are caused in the previous casting phase. In the past, this job was done by well-trained workers. They checked the surfaces for potential defects and classified them into pre-defined categories. In order to reduce the wage cost, this process is also attempted to be replaced by an automatic defect detection and classification with a vision system.

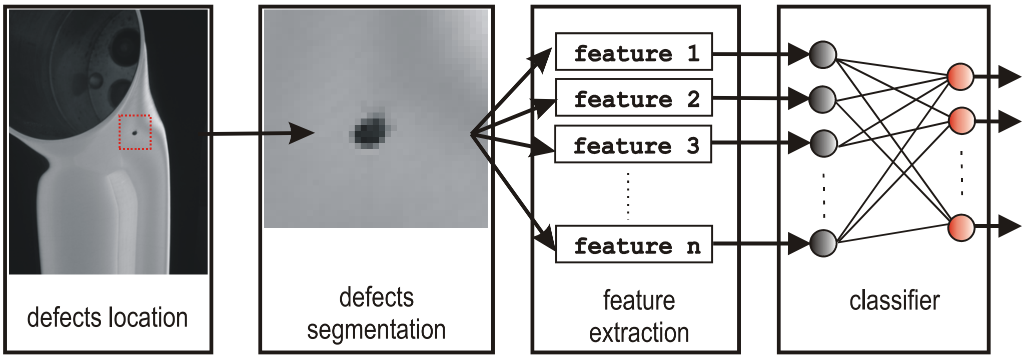


Figure 11. The framework of the automatic identification and classification system

The planned classification system mainly consists of four parts: defects location, defects segmentation, feature extraction and classification. The four parts in the system act sequentially. Fig. 11. shows the framework of the classification system.

At the beginning, the system detects and locates the defect in the grayscale bitmap obtained from the vision system. Before encoding the bitmap into some meaningful feature values, the area of the defect in the image should be pre-processed and segmented. In this phase the separate defect image is obtained and is ready for feature extraction. The feature extraction is the most important part in the system. It defines the rules of describing and expressing the defects inside an image in a form that a classifier can understand. At first, those features should be found which are useful to differentiate one defect from another,.

The features are not limited to the shape feature, but can be the texture features and some statistical features of the segmented image. Then, an effective way should be found to encode such selected features, specially in a mathematical way. After the data pre-processing, the feature data are applied as the training data to the classifier. Many artificial intelligent techniques have the capability of multi-class classification, e.g. k-NN, MLP network, RBF network, SVM, etc.

The vision system adopt a specific illumination method to get a light color surface lying on the dark background. A small window is then applied on the grayscale picture of the surface. By analyzing the grayscale distribution in each small window, the defects can be located precisely because the defects are of deep color and surrounded by the light color surface. Fig. 12. shows 5 different defects selected from the total 19 classes that have been found on the workpiece surface.

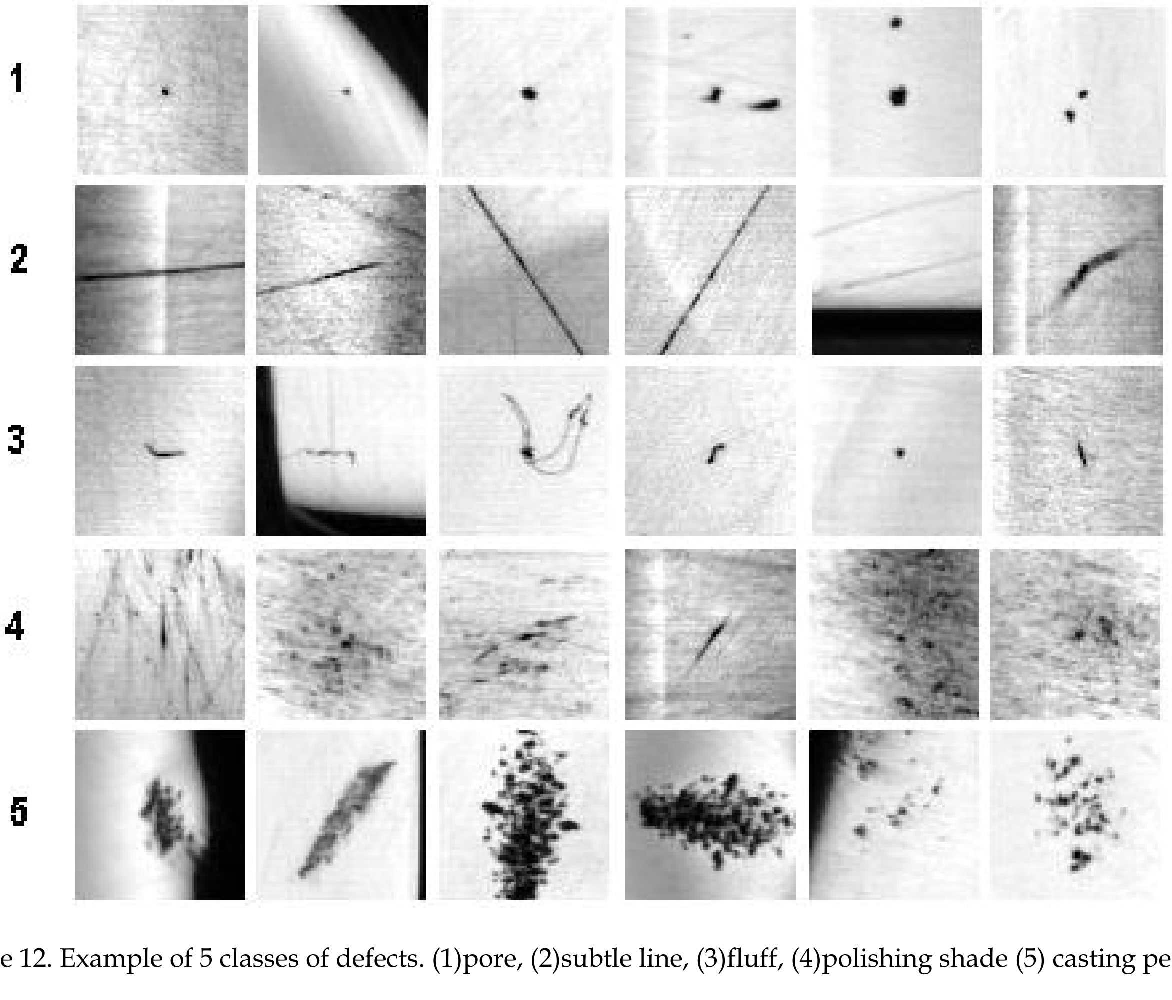


Figure 12. Example of 5 classes of defects. (1)pore, (2)subtle line, (3)fluff, (4)polishing shade (5) casting peel

Technicians differentiate the defects by experience. Actually such experience involves some rules. For example, a small black solid circle in the white background may be the "pore" defect; a long stroke on the surface may be a "subtle line" defect. In other words, a technician draws a conclusion by combining some attributes of the defect in the segmented image, e.g. size, grey scale distribution and so on. All these attributes used by technicians (no matter intentionally or subconsciously) to discriminate defects are called valid features of the defect.

The length and width of the defects are the simplest valid features. It is very easy to see that such size information are the reason for a technician to differentiate the defects. However, not all features used by human beings are as straightforward as the shape features such as length and width. The fact is that you use some features to classify but you do not know you use them as a base. From the result of some other similar projects (Lampinen&Smolander 1994; Iivarinen&Visa 1998), three kinds of features are normally extracted to identify and classify the object, shape, texture and some statistical features of the image, e.g. grayscale histogram. It is not obligatory to include all those features as the input to the classification process. It depends on the project itself to decide which features to include.

Two basic principles should be hold in mind:

1. The features extracted are sufficient to separate different defects.
2. The features that is not helpful to discriminate the defects are not extracted to the classifier.

The first principle is a precondition of the system to function. Different defects are represented by features of different values. Think of the classifying of an apple and a pear, for example. Only by the shape feature it is not possible for people to tell whether it is an apple or a pear. It does not comply with the first principle because the apple may have the same and resemble the outer shape of the pear. The second principle is to ensure the generalization of the classifier.

Those features that are not helpful to distinguish the objects are not selected best. For example, the weight is not a feature that should be used because an apple is possibly heavier or lighter than a pear. Unnecessary features act like noise in the system. They distract the classifier from the right way and influence the performance of the classifier. The quantitative measurements of features are called descriptors. The way to measure shape are called shape descriptors while texture features are expressed by texture descriptors. There are many different descriptors available. Descriptors design and selection is an important research topic in the classification system. The current system uses only some simple shape descriptors, e.g. length, breadth, compactness and roundness of the defect. The right classification ratio is only about 30% now. There are three reasons for the low classification ratio.

The first one is the lack of the effective descriptors to account for the defects. The second reason is that the system did not use the up-to-date intelligent classifier and the last one is that there are two many pre-defined classes of defects (19 classes). Some kinds of defects are so similar that even skilled people can not differentiate them from each other. For example, "polishing shade" is similar to the "subtle line". When only analyzing the grayscale pictures, the "pore" defect is difficult to separate from the "fluff" defect by the operator in some cases, even though they are different in reality, see Fig. 12. The classification of defects is still under development. It is supposed to be able to reach an 80% right classification ratio by well-presented feature extractors and tailored classifier in the future.

6. Summary

Due to the tendency of the globalization of markets and competition from low wage countries, it is profitable for the industry to further automate the robot-controlled grinding and polishing system, especially in the sanitary fitting industry. The high automation level of the robot system not only relieves the human being from laborious tasks, but also elevates the efficiency of the whole manufacturing process by producing a steady product surface quality. A framework of such a robot system is put forward in this paper. The automation of the robot system focuses on two parts. One is to facilitate the grinding and polishing paths planning processes. Software is developed to help the operator to generate the grinding and polishing paths quickly and easily. It offsets the incapability of universal off-line programming systems for grinding and polishing. With this new software the tasks that had to be done by high-specialized staff before can now be taken over by normal operators. To help the operator to design the paths, the belt grinding process and its simulation are carefully studied. Another key point of the automation is to automatically find and classify the defects on the workpiece surface into some pre-defined classes in order to apply an according compensating machining process. The detection of the defects is already realized while the classification should be improved in the future work.

This research and development project was funded by the German Federal Ministry for Education and Research (BMBF) within the national "Research In Production For Tomorrow" Programme and managed by the Project Agency for Production and Manufacturing Technologies (PFT), Forschungszentrum Karlsruhe

7. References

Blum, H; Suttmeier, F.T. (2000) An adaptive finite element discretisation for a simplified signorini problem. Calcolo, 37(2):65-77.

Blum, H; Schroeder, A; Suttmeier, F.T(2003). A Posterori Error Bounds for Finite Element Schemes for a Model Friction Problem. Conference Proceedings of Simuation Aided Offline Process Design and Optimization in Manufacturing Sculptured Surface, pp 39-47, Witten-Bommerholz, Germany.

Cabaravdic, M; Kneupner, K; Kuhlenkoetter, B. (2003) Methods for efficient optimisation of robot supported grinding and polishing processes. International Conference on „Trends in the Development of Machinery and Associated Technology", Barcelona, Spain.

Hammann, G.(1998) Modellierung des Abtragsverhaltens Elastischer Roboter-gefuehrter Schleifwerkzeuge. PhD thesis, University Stuttgart.

Iivarinen J; Visa, A. (1998) An Adaptive Texture and Shape Based Defect Classification. In the Proceedings of the 14th International Conference on Pattern Recognition, vol. 1, pp. 117-122, Brisbane, Australia

Lampinen, J; Smolander, S. (1994) Wood defect recognition with self-organizing feature selection. Intelligent Robots and Computer Vision XIII: Algorithms and Computer Vision, Proc. SPIE 2353, pp. 385-395.

Kneupner, K. (2002) Directcontrol. Ein Programmierkonzept fuer Roboterzellen. In Conference proceedings of Robotik, Ludwigs-burg, Germany.

Kneupner, K. (2004). Entwicklung eines Programmier und Steuerungskonzepts fuer Robotersysteme auf der Basis eines Umwel-tmodells. PhD thesis, Dortmund University.

Koenig, W. (1996) Fertigungsverfahren Band 2 --- Schleifen, Honen, Laeppen. VDI Verlag.

Krause, R. H. (2001) Monotone Multigrid Methods for Signorini's Problem with Friction. PhD thesis, Freie University Berlin.

Kreis, W; Kneupner, K. (2001a). Ein Simulationsgerechtes Prozessmodell fuer das Freiform Bandschleifen. Dortmund Germany.

Kreis, W; Kneupner, K. (2001b) Simulation des Bandschleifprozesses. Frontiers in Simulation, Simulationstechnik 15th. Symposium, pp 517-522,. Paderborn Germany.

Meyerhoff, M. (1998) NC-Programmierung fuer das kraftgesteuerte Bandschleifen von Freiform-flachen. PhD thesis, University Hannover.

Schueppstuhl, T. (2003) Beitrag zum Bandschleifen komplexer Freiformgeometrien mit dem Industrieroboter. PhD thesis, University Dortmund.

Suttmeier, F:T. (2001) Error Analysis for Finite Element Solutions of Variational Inequalities. PhD thesis, Dortmund University.

Reconfigurable Mechatronic Robotic Plug-and-Play Controller

Johan Potgieter, Jonathan Zyzalo & Olaf Diegel

1. Introduction

In today's manufacturing community, flexible manufacturing systems (FMS) are becoming important. There is a global trend towards flexible manufacturing systems with an increasing number of countries competing for a share of the world market. FMS are increasingly required for companies to meet the demand for high quality, reasonably priced products. FMS allow a manufacturer to quickly change processes or operations to produce any product, at any time, while striving to keep the processes economical.

To remain competitive companies should be replacing or upgrading old, obsolete methods, processes, and systems with the latest advances in manufacturing technology. This competition within the manufacturing industry has led to a focus on producing high quality parts quickly and accurately. The quality of products, along with the increased productivity necessary to compete globally, has led more and more manufacturers to introduce advanced manufacturing technologies in their factories. In order for these advanced manufacturing technologies to be accepted by manufacturing companies, they need to be able to implement the new technologies and their processes as quickly as possible so as to keep manufacturing down-time to a minimum. This allows the manufacturer to respond quickly and with increased flexibility to market needs. Such is the need for FMS.

Important to FMS is computer-based technology. Since the advent of the microprocessor, computer-based technologies have made it possible to improve productivity, reduce manufacturing costs, and produce better, uniform quality goods. Manufacturers who introduce computer-based technologies into their manufacturing environment are able to increase their productivity and in turn increase their market share.

Many manufacturers have made the change over to either completely or partially automated systems and processes. The development of the microprocessor has seen the use of robots increase in many applications. FMS generally consist of a number of automated machine tools and materials-handling systems. These systems are flexible enough to reconfigure their elements or processes in order to produce a product. Robotics finds itself to be an important part of FMS due to the flexibility of robotic arms (Krar & Arthur, 2003).

The past few decades has seen a large increase of automated machines in the manufacturing environment. It is therefore important to know exactly what a robot is, and what makes it different from other automated machines, as robots come in many shapes and sizes. The word robot comes from the Czech word "robota" meaning work, servitude or forced labour. The most widely accepted definition of a robot was given by the Robotic Institute of America in 1979:

"A robot is a reprogrammable, multifunctional manipulator designed to move materials, parts, tools, or specialized devices through various programmed motions for the performance of a variety of tasks."

George C. Devol designed the first programmable robot in 1954. Its first use in a manufacturing industry was in the General Motors plant, in which its function was to extract heated die-castings from die-casting machines and to perform spot welding on car bodies (Robotics Research Group, 2003). Since this first application of robot arms in the American automotive industry, robot arms have improved tremendously and found many more applications. Most modern industrial robot arms are capable of various types of materials handling, manufacturing processes, assembly, sealing and painting. Robot applications are now also found in science, research, engineering, and medicine.

Today, there are many companies manufacturing industrial robot arms. A problem that has arisen from the many available robot arms is a lack of standardisation. There is little or no interoperability between different manufacturers' systems and between the different generations of automated machinery. Most robots and other systems are custom built and difficult and expensive to upgrade. The main disadvantage of robots is therefore that a dedicated controller is required to control the robots actuating systems. This proves costly and usually makes interfacing with the robot complex due to hardware and software conflicts. It also reduces the flexibility of the machine.

Robotics is an interdisciplinary field that ranges in scope from the design of mechanical and electrical components to sensor technology, computer systems and artificial intelligence (Fu et al., 1984). Mechatronics brings together the disciplines of mechanical engineering, electrical and electronics, information technology and software engineering. It involves knowledge of technologies such as sensor and measurement systems, drive and actuation systems, analysis of system behaviour, control systems, and microprocessors (Bolton, 1999). Mechatronics, therefore, encompasses all the areas and skills needed in the field of robotics.

One aspect of mechatronics research is the ability to look for modular, low cost solutions for the control and flexibility of systems. This is an area in which mechatronics is able to aid the field of robotics by an investigation into an adaptive modular robot actuating system. This becomes a logical solution to the problem of interfacing automated machinery at a computer-based level, and is referred to as Computer Integrated Manufacturing (CIM). At the PC level, "plug-and-play" has become the benchmark for devices, allowing ease of use and increased flexibility. Upgrades have simply become a matter of updating software. Another benefit of a modular actuating system is that it becomes easier and quicker to replace parts than to get them repaired. This reduces manufacturing down time. Instead of a manufacturing companies going out and buying a whole new robot system, they may be able to salvage old robot arms that have been out of service simply because they are too expensive to repair.

2. Background

2.1 A Brief History of Robotics

During the decade of the 1960s, one of the most important advances in technology that has aided robotics occurred. The first integrated circuits, combining multiple transistors and other components within a single chip, started to appear. Development of electronic technologies continued throughout the decade (Bradley et al., 2000). The 1970s saw a boom in robot designs mainly due to the increased development of integrated circuit technology. Large scale integration (LSI) was allowing 10,000 components onto a square centimetre of chip. This led to the introduction of the first 8-bit microprocessor. Before this, robots were reprogrammable but blind, deaf, and had no logic. Robot developers were now able to get their robots to perform more difficult tasks such as welding and assembly processes. In 1978, Unimation launched the PUMA which stood for Programmable Universal Machine for Assembly.

The 1980s saw a second generation of robots that had sensing capabilities. This was due to further advances in microchip technology. Intel introduced the 32-bit microprocessor in 1980. In 1981, IBM introduced the first personal computer. Dedicated microprocessors began to be used to control the operation of an increasing range of systems (Bradley et al., 2000).

The 90s built on the advances in electronics and other technologies. Robots were given mobility and autonomy. They were able to make intelligent decisions on their own. Electronics and processing power has led to the development of application specific integrated circuits (ASIC). This has allowed the development of robots that have intelligence and, as a result, have increased the performance and reduced costs of manufacturing (Bradley et al., 2000).

Robotic design has simplified the automation of complex manufacturing processes. There are now many more applications of robots outside the manufacturing industry. Senses such as sight, touch, and hearing have been added to allow robots to perform more complicated and sensitive tasks. With these advances in the state of modern technology, robots now have many applications in science, engineering, space, and medicine.

2.2 PUMA 560 Robot

As a workhorse of the robotic industry, it is useful to have a good knowledge of the original PUMA 560 series robot arm. In the original PUMA 560, each joint is driven by a 40V brushed permanent magnet DC motor, with the motors for the bottom three joints rated at around 160W and the motors in the wrist rated at around 80W. Each of the first three joints (waist, shoulder, elbow) are also equipped with a 24V electromagnetic brake which must be released before that joint can operate. These brakes stop the arm from collapsing when the power is removed. All the joint motors were fitted with 250 line sin-cos incremental encoders giving position feedback to the controller.

The controller consisted of a DEC LSI-11/02 computer, and six joint controllers that consisted of a Rockwell 6503 microprocessor, a digital-to-analog converter (DAC), and a current amplifier. The DEC LSI-11/02 computer was used to compute the setpoints that made up the desired robot joint level position. The setpoint was updated by the LSI-11/2 every 28ms. Each update cycle, new setpoints for each joint are transferred from the LSI-11/2 to one of the six Rockwell 6503 microprocessors. The microprocessor received and acknowledged the new setpoint and performed interpolation between the current setpoint value and the new one. This was the computed joint error. Every 0.875ms the microprocessor would read the incremental encoder and then update the error. The error-actuating signal would be converted to a current signal using the DAC, and the resulting current would then be sent to the amplifier to move the joint (Fu et al., 1984).

This control scheme is essentially a PID control loop. One of the main disadvantages of this type of control, as will be noted later on in the discussion section, is that the feedback gains are constant and pre-specified. The feedback gains cannot be updated for varying payloads. Since inertial loading and the effects of gravity dependant on the position or velocity of the robot arm, the simple PID control scheme does not perform very well (Fu et al., 1984).

The original control system was contained in a large racking box along with a terminal system for operator control and programming of the robot. The motor and brake control signals to the robot arm, and the encoder and potentiometer signals coming back from it, were transmitted along an 80 core, 5-metre umbilical cable.

The PUMA 560 series robot that was donated for the project came with the entire arm intact. The robot was a six-axis revolute robot arm. It had a complete wiring loom and the 80 core umbilical cable with connectors. However, there was no power supply for the motors, no information as to the workings of the encoder circuits on the motors, and no controller hardware. These essential parts of the project required investigation and development before the control hardware was to be implemented.

3. Hardware Development

From the specifications of the PUMA 560 robot, it was found that a 40V DC power supply was needed to power the motors. The motors were also equipped with 24V electromagnetic brakes, so a 24V DC supply was needed. A 5V logic supply was also necessary to power all the encoder circuits on the motors and the microprocessors used.

3.1 Full-Wave Power Supply

Electrical power in New Zealand is distributed as 50Hz AC with a nominal voltage of 230Vrms. Transformers are used to obtain any other AC voltage. Two circuit diagrams of a full-wave power supplies are shown in the Figure 3.1. Only the secondary coils of the transformers are shown.

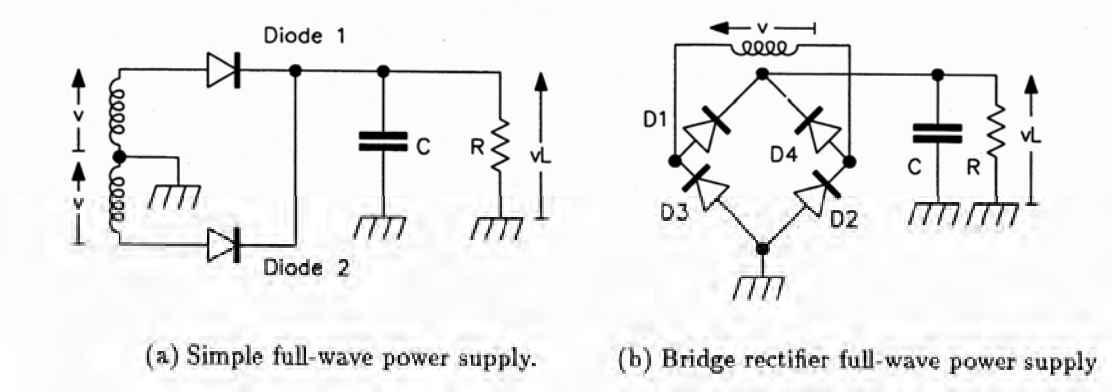

Diode 1
Diode 2
v
v
C
R
vL
v
D1
D4
D3
D2
C
R
vL
(a) Simple full-wave power supply.
(b) Bridge rectifier full-wave power supply

Figure 3.1 Full-Wave Power Supply Circuits (Bold, 2000)

Figure 3.1(a) uses a transformer with a centre-tap that is connected to ground. The other ends of the secondary coil are each connected to a rectifying diode. Both halves of the secondary coil winding have the same voltage across them, but they are opposite in polarity. When the top of the transformer is positive, diode1 conducts to charge the capacitor. When the bottom end of the transformer is positive, diode 2 conducts and also charges the capacitor. Diode 2 can be thought as inverting the negative goes portion of the AC output voltage of the transformer.

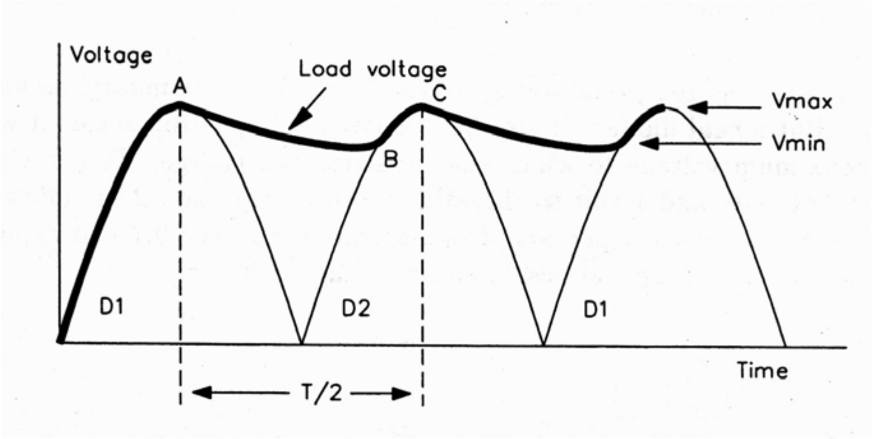


Figure 3.2 Waveforms observed with full-wave power supplies (Bold, 2000).

Figure 3.2 shows the resulting waveform created by the circuit. Diodes 1 and 2 are active in the portions of the cycles label D1 and D2 and conduct alternately. If the capacitor is initially uncharged and the input voltage goes positive, the capacitor charges up to the peak positive voltage at point A. The transformer drives current into both the load and the capacitor. When the transformer voltage falls below the peak voltage, the capacitor begins to discharge through the load giving the portion of the waveform from A to B. The load voltage falls until point B where it intersects the rising of the waveform again. At this point the capacitor charges again and so the cycle repeats. The voltage drop between point A and point B is known as the ripple voltage. Using a larger rated capacitor can smooth this ripple.

The following equations show how the capacitor and transformer voltage can be calculated.

$$C \approx \frac{I_{DC}}{2V_R f} \tag{3.1}$$

$$V_{rms} = \frac{V_{DC} + \frac{V_R}{2} + 1}{\sqrt{2}} \tag{3.2}$$

Where C is the capacitor value and Vrms is the voltage required from each transformer winding. VR is the ripple voltage, f is the frequency of the waveform (50Hz), and VDC is the required DC voltage. The one added to equation (3.2) is to compensate for the small voltage drop that occurs across the diodes when they each conduct. Generally this drop is 0.7V for most power supply diodes (Bold, 2000).

Figure 3.1(b) shows the more common bridge rectifier circuit. The secondary voltage is shown in the upper part of the circuit diagram. This circuit does not require a transformer with a centre-tap (which are more expensive). When the transformer secondary voltage is positive to the left, diodes D1 and D2 conduct and charge the capacitor. When the transformer voltage polarity reverses, diodes D3 and D4 conduct using the other half of the AC voltage input waveform. The analysis of the circuit is the same as the first, except that two diodes conduct at once. This means that equation (3.2) needs a two diode drops subtracted from the secondary transformer voltage. This is done by increasing the voltage output from the transformer by two. The new equation becomes:

$$V_{rms} = \frac{V_{DC} + \frac{V_R}{2} + 1}{\sqrt{2}} \tag{3.3}$$

To construct the power supply for the PUMA robot required the purchase of a transformer. In the electrical industry transformers are specified by their VA rating. The VA rating is the product of the voltage required on the secondary coil and the current required for the application. An assumption was made about the amount current required for the power supply base on the equation for power and the power rating of the motors. The calculation was as follows:

$$P = VI = I^2R = \frac{V^2}{R} \tag{3.4}$$

$$P_{Motor} = 40V \times I$$

$$I = \frac{P_{motor}}{40V} \tag{3.5}$$

The maximum current necessary for the power supply becomes the sum of currents if all the six motors are going. For the larger 160W motors the current that is drawn at 40V is 4Amp and from the small 80W motors it is 2Amp. This results in a sum of 18Amp for the case where all the motors are drawing their maximum. So for the transformer it was decided to go with the secondary coil rated at 20Amps.

Now the ripple voltage needed to be found so as to work out the equation for the transformer output voltage. The capacitor that was used for the power supply was a 43,000mF electrolytic capacitor, which was salvage from a prevent power supply application. Equation (3.1) is arranged like:

$$C \approx \frac{I_{DC}}{2V_R f}$$

$$V_R = \frac{I_{DC}}{2Cf}$$

$$V_R = \frac{20}{2(43{,}000 \times 10^{-3})(50)}$$

$$V_R = 4.7 \times 10^{-3} V$$

This gives a very small ripple voltage of only 4.7mV and shows that the capacitor does a very good job at smoothing the power supplied.

To work out the AC output voltage of the transformer, equation (3.3) could now be applied.

$$V_{rms} = \frac{V_{DC} + \frac{V_R}{2} + 1}{\sqrt{2}}$$

$$V_{rms} = \frac{40V + \frac{4.7 \times 10^{-3}}{2} + 1}{\sqrt{2}}$$

$$V_{rms} = 29.0V$$

It was found out that in the electrical industry, 230V to 32V transformers are a common standard. So the calculation was reworked to find out how much DC voltage this would supply to the motors. The result was a voltage of approximately 44V DC, which was within the limit of the motors. Thus a transformer with a VA rating of 640 (32V×20A) was ordered.

3.2 Switch Mode Power Supply

The logic power for the encoder circuits and the microprocessors was provided using a computer ATX switch mode power supply unit. These are very cheap (if not free) and can be taken from any old PC just as this one was. This gives a very convenient logic supply as it provides a +5V, -5V, +12V, and –12V. The potential difference between the +12V and –12V rails also supplies the 24V necessary to switch the electromagnetic brakes of the three larger motors.

Four switches were required for test purposes. One was to turn on the 40V DC supply, one to turn on/off the brakes, and also one to turn on the logic supply. Finally, a high-power switch for the emergency stop, which cuts power supplied to the entire system.

3.3 Motors, Encoders, and Potentiometers

The motor, encoder, and potentiometer assembly of each joint did not come with any wiring or circuit diagram. External diagnostic testing of the circuits was not possible as the encoder circuit is very sensitive to incorrect polarity. The assembly was disassembled so as to gain a better understanding of the wiring and physical layout of the assembly. Table 2 lists the wires coming off the motor, encoder, and potentiometer assembly, their colour and corresponding functions.

Wire Number	Wire Colour	Function
1	Violet	+Vpower
2	White	-Vpower
3	Orange	-EncA
4	Green	+EncA
5	Yellow	+EncB
6	Light Blue	-EncB
7	Green	Index
8	Red	+5V encoder/pot
9	Dark Blue	+Vbrake release
10	Gray	Brake return
11	Black	0V ecoder/pot
12	Green/White	Vpot

Table 2. Motor Assembly Wiring

One of the major difficulties in the project had to do with the encoders. Based on an investigation into the workings of incremental encoders, a square wave output was expected from the encoder circuits. However, due to age of the encoders, only a sine wave signal was measured and this output signal was small, at only 30mV. As the sine wave was analogue, the signal required digitisation. This required an analogue-to-digital converter (DAC). However, for the microprocessor to count the encoder increments, it required digitisation into a pulse train rather than a digital representation of the analogue signal that the DAC would provide. Converting this analogue signal into a pulse train proved very time consuming.

The cost of replacing these encoders with modern industrial encoders was looked at. However, the cost of the new encoders was deemed too expensive in comparison with making up a signal conditioning circuit from discreet components. Using knowledge of electronics and signal processing, a circuit was designed. The encoders used for each motor were quadrature encoders. Two pairs of signal lines were provided but it is the differential output of a pair of signal lines that provide the encoder line information. Amplification of the signal was accomplished by using a standard LM741 operational amplifier to amplify the differential of each pair of signal lines. But the conversion of the amplified signal into a pulse train was not possible. It was not until an encoder circuit from a PUMA 560 Mark II robot was obtained, that the problem could be overcome.

On the PUMA 560 Mark II model, this problem with the encoder sine wave output was corrected by placing the circuitry within the encoder. Upon inspection, it was found that a voltage comparator IC was used to obtain the square wave output. Once a voltage comparator IC was purchased, the LM339, it did not take long to get the desired result for the analogue-to-pulse train circuit. The circuit is shown in Figure 3.3.

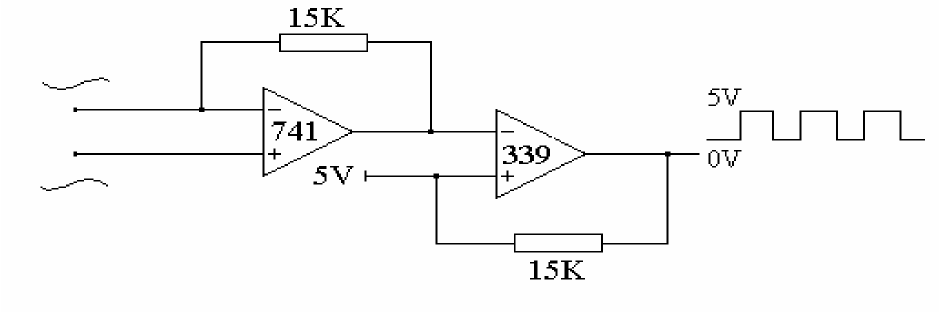


Figure 3.3 Encoder output conversion circuit

This circuit was implemented on each of the encoder lines of the six motors. Therefore twelve of these circuits needed to be made. They were all first implemented using matrix board. But because of errors in the circuit, the high density of the circuit, and other design problems, a more modular design was created using an electronic PCB design program. This design was then etched onto a PCB.

3.4 Modular Design

One of the objectives of the project was to create the system that was modular. The idea behind this was so that, when a part of the system is not working or broken, it can easily be replaced by another module. When looking at existing modular industrial systems, ie. PLCs, it was found that most use a standard 19” racking system. Modules slide to position and plug into the back of the rack or are daisy-chained to other modules already on the rack. For this project, the modules that slid into the rack were the control cards. The rack used was a standard 19″ rack system that was donated by a lighting company.

The cards that slid into the rack used to be lighting dimmer control cards. All the lighting components were removed, except the gold terminals and PCB used to plug the card into the rack. These cards were then fitted with all the components necessary for the control of two motors. Each control card module contained a BrainStem Moto 1.0 and two Devantech MD03 H-bridge Motor Drivers. With each control card module it was therefore possible to control two motors. The BrainStem offeres several control options that make the card flexible enough to control almost any two motors rated below 50V and 20Amp. Because the BrainStems can also be daisy-chained using an I^{2}C bus, extra control cards can be placed anywhere on the bus. All that is required is for each modules I^{2}C bus address to be set up on the BrainStem before "plugging" the card in.

All the inputs and outputs were routed to the back of the module so that each module simply plugged into the rack. The I/O from the BrainStem was wired to the back with ribbon cable and a female DB-25 pin connector was used to plug the I/O into the rack.

The rack had several buses on the back of it. These were used for the 5V and 0V logic supply, the 40V and 0V motor power supply, and the I^{2}C bus. The rack was modified for the added male DB-25 pin connectors used by the cards.

The connection of the umbilical cord to the PUMA 560 required consideration because the connector used was an old, high density connector which was no longer commercially available. An industry standard Wieland connector was used instead. The motor wires were separated from the signal and logic wires and were given their own connector.

Wieland connectors are available in a variety of pin options. A 12 pin Wieland connector was used for the motor wiring and a 24 pin Wieland connector for the signal and logic wiring. Wieland connectors use screw-in terminals rather than crimps that make wiring more flexible and convenient. The Wieland connectors thus add to the modularity of the system by giving it a generic plug so that other robot umbilical cords or machinery using motors, can be easily plugged in.

4. Hardware Control

4.1 H-Bridge Motor Drivers

The direction in which a permanent magnet motor turns depends on the direction of the current flowing through the armature coil. To reverse the direction of the motor, the current through the armature coil needs to be reversed. This could be done by physically changing over the supply terminals to the motor. This is highly impractical, so an electronic circuit, called an H-bridge, is used to do this and is shown in Figure 4.1.

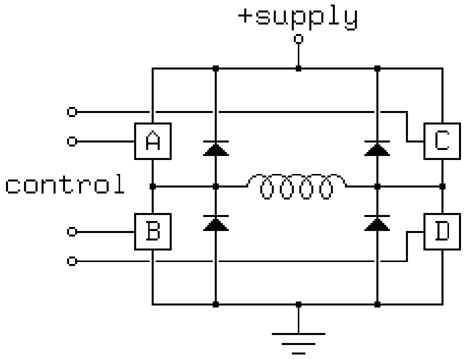


Figure 4.1 H-Bridge Circuit (Jones, 2001)

In the figure, A, B, C, D are transistor switches. The switches used in the H-bridge must be protected from the voltage spikes caused by turning the power off in a motor winding. This is usually done with diodes, as shown in the figure.

With four switches, the basic H-bridge offers 16 possible operating modes, 7 of which short out the power supply! However, only the modes that give the direction of the motor are of interest. The Forward mode has switches A and D closed (see Figure 4.2). Reverse mode has switches B and C closed. These are the basic modes required by an H-bridge that allows the current through the motor armature coil to be changed.

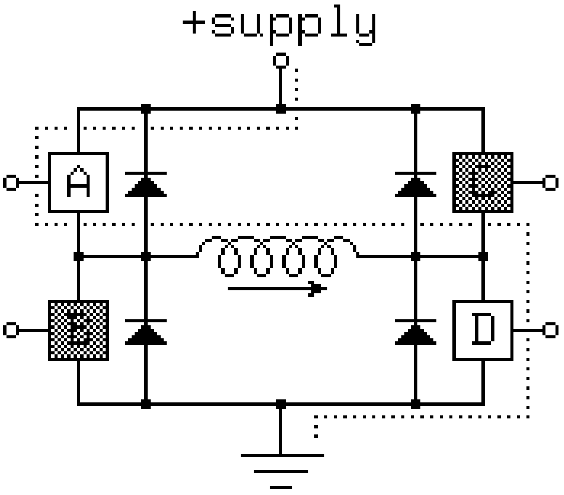


Figure 4.2 H-Bridge Circuit in Forward mode (Jones, 2001).

The H-bridges used for the project were the Devantech MD03 as they offered the voltage and amperage requirement of the system. Rated at 50V 20Amp, the MD03 H-bridge is a medium power motor driver. It has high degree of flexibility with several modes and control options. Each MD03 requires a 5V logic supply and a separate motor voltage supply that can be anything from 5V to 50V DC.

The MD03 has four control modes. It can be controlled using a standard I^{2}C bus, 2 analogue modes, and a remote control mode used for control directly from an RC receiver. The mode of interest to the project was one of the analogue modes, the Analogue Mode – 0V-5V. This mode is useful because it can also be controlled using the digital TTL logic levels which come from the microprocessor. The SDL input of the MD03 is used to indicate the direction, logic 0 for reverse direction and logic 1 for forward direction. The SDA controls the amount of voltage sent to the motor. 0V indicates no power and 5V indicates full power. This mode allows a pulse width modulation (PWM) signal to be used instead of an analogue voltage on the SDA input. A resistor/capacitor filter on the MD03 generates an analogue voltage from the input PWM signal.

PWM is a technique used by most microprocessors and other controllers to control an output voltage at any value between the power rails. It consists of a pulse train whose duty cycle is varied so that it creates variable “on” and “off” states. The average output value will be approximately the same percentage as the “on” voltage. So, in the case of MD03 H-bridge mode that was used for the project, a 0% duty cycle represented 0V supplied to the motor, a 50% duty cycle represented half of the supply voltage available to the motor, and a 100% duty cycle indicated maximum voltage.

4.2 PIC Motor Control

The motors of the robot were controlled using a PIC18F252 microprocessor. This PIC has been implemented in a module called the BrainStem Moto 1.0 by the Acroname company. Acroname has been able to implement all the features of this PIC microcontroller into a small motor control package. The main features of the BrainStem Moto 1.0 are as follows:

- 40MHz RISC processor.
- 2 motion control channels with PWM frequencies from 2.5kHz-5kHz.
- 1 dedicated 10-bit ADC.
- 1 dedicated digital I/O line.
- 1 Mbit I^{2}C port.
- I^{2}C routing.
- Status LED.
- 11 1k TEA file slot and 1 16k TEA file slot.
- 368 byte of user RAM.
- RS-232 serial port communication.

Mode Descriptions	
PWM	Enables you to vary the amount of power sent to the motor. This is a useful starting point to verify motor and h-bridge operation. This mode does not provide position or speed feedback.
PWM - Encoder	This mode works like PWM, but provides encoder input for measuring motor velocity. This mode also includes the ability to set an input offset value and encoder sampling rate.
PWM - A/D	This mode works like PWM, but provides input from an A/D channel. The value is displayed but not used as feedback for motion control. This mode is the starting point for doing Back-EMF control as it lets you configure your system to suit your motors.
A/D PID	This mode uses the feedback from an analog channel to determine the position of the system. A PID control loop attempts to maintain the desired position.
A/D Velocity PID	This mode uses the feedback from an analog channel to determine the velocity of the system. A PID control loop attempts to maintain the desired velocity.
Encoder PID	This mode uses the feedback from an encoder to determine the position of the system. A PID control loop attempts to maintain the desired position.
Encoder Velocity PID	This mode uses the feedback from an encoder to determine the velocity of the system. A PID control loop attempts to maintain the desired velocity.
Off	Disables all PWM and control outputs and sets the channel's digital IO pins to be inputs.
Step	Provides stepper motor control. This mode requires a different H-Bridge specifically designed for stepper motor control. This bridge component will be available as an optional driver bridge in the future.

Table 3. Mode Descriptions (Acroname, 2003)

The Moto module is pre-programmed with seven different modes. All of these modes allow for custom settings and configuration depending on the type of motors that are used and the type of sensors used to monitor feedback from the motors. The following Table 3 lists the modes available with a brief description of what they do.

The modes of particular interest for the PUMA 560 robot's motors were the Encoder PID and Encoder Velocity PID modes. The Encoder PID mode makes adjustments to the position of the motor based on feedback from the motor's encoder. PID control is used in an algorithm on the PIC to determine how much PWM is applied over time to the motor in order for a desired position to be reached and maintained. Proper selection of PID gain constants minimise oscillations of the motor (Acroname, 2003). The following Figure 4.3 shows how the overall flow of the control loop is done.

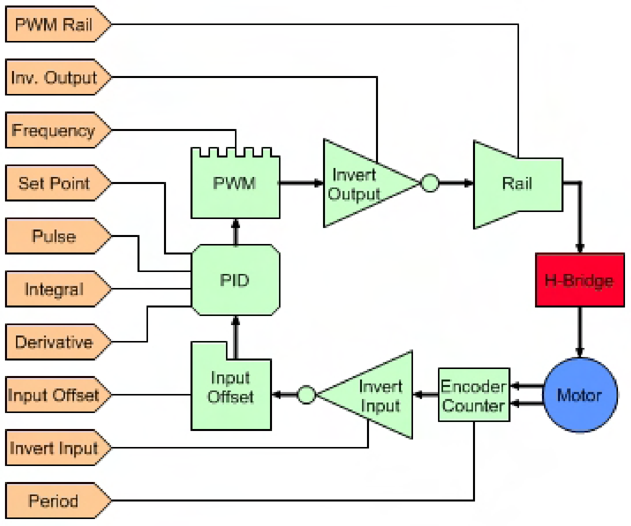


Figure 4.3 Basic logic flow of the Encoder PID mode (Acroname, 2003)

The Encoder Velocity PID works similarly to the Encoder PID mode, however, instead of moving to a given setpoint and maintaining that setpoint, the mode allows motion at a constant setpoint speed and maintains that speed. This mode did not prove very practical for the project implementation, so it was not used. Instead, encoder PID mode was used for the project.

The other important feature of the PIC is the use of reduced instruction set computing (RISC). The Moto responds to a limited set of pre-programmed commands. Acroname provides a reference library on its website with information of how these commands work.

4. Software Development

4.1 PC Graphical User Interface

The Graphical User Interface (GUI) was developed using a common object-orientated programming language, Visual Basic 6.0. The GUI was developed to test the functionality of the system. It communicates with the microprocessors through a serial cable connected a com port on any PC. The GUI communicates with each of BrainStems through the first BrainStem Moto module, which acts as a router. Each packet of data sent from the PC contains an address for the respective BrainStem the data is meant for. If the packet is not addressed to the first router, it sends the pack on the I2C bus until it reaches the appropriate module. The main GUI serves the following major functions:

- Communication management of packets between the BrainStems and the PC.
- Inputs for setpoints for each motor that is to be controlled.
- Allows manipulation of the settings of programs running on the PIC. In the case of the BrainStem, this includes such things as PID control settings, mode selection, PWM, register monitoring, etc.

5. Discussion

When the entire system was put together, adequate control of the robot was possible. Overall the system performed well enough to get controlled motion of the six joints of the PUMA robot arm. The system was also subsequently used to control a CNC lathe. There were, however, a few problems with the final system. The system also requires some further enhancement to achieve better control of joint positioning.

The main problem occurred with the logic power supply. Every now and then the logic power, which supplies all the encoder circuits and microprocessors, would fail to power the most essential parts of the system. This would result in unpredictable behaviour from the robot arm and make it unsafe. It was found that the grounding of the logic power supply was floating causing differences in the ground of the system components. By using the ground of a bench-top power supply, this problem was solved.

The rest of the problems with the final system were a result of the PID control method. The PID control was not able to account for the effects of inertia and gravity because the feedback gains of the PID algorithm were fixed. This meant that the system had a very low level of repeatability and accuracy. The Encoder PID mode of the BrainStem Moto 1.0, although working well for the wrist joints, did not perform very well for the larger joints. When a new setpoint was entered into the PID control loop, the control algorithm would output maximum voltage to the motor until it neared the new setpoint based on the feedback from the encoders and the PID gains. It did this without accounting for the effects of inertia and gravity. In practice this meant, for the shoulder and elbow joints especially, that the joint would move more rapidly in the down direction that it would in the up direction. To correct this problem, a velocity control method needs to be explored.

Another problem that exists with the BrainStem Moto 1.0, is that it only has a 16-bit setpoint number and this is reduced to 15-bits as the most significant bit is a directional bit. Due to the resolution of the 250 line incremental encoders, only limited movement of a joint can be completed with each command sent to the BrainStems.

This leads on to another problem with the BrainStem program. It is not able to detect when a movement is complete so that the next move of the robot can take place, i.e. there are no flags set to indicate a new setpoint has been achieved. This problem was overcome, in Visual Basic, by introducing a timer delay between selected movements.

A problem also occurs when the motors are powered at high speeds. At high motor speeds the PIC fails to read the encoder pulses. This results in uncontrolled motion of the joints. In some cases the encoder circuit that was developed for the incremental encoders fails to supply the pulse train signal to the BrainStems. This may be because there was a frequency limitations on the ICs used in the encoder circuit. This problem could also arise from aliasing. This would depend on the rate at which the PIC samples the input signal from the encoder circuit. The effect of this problem is that robot must operate at reduced speeds in order to maintain control.

The only other item of note is that the robot joints require calibration every time the robot is turned on. This is the purpose of the potentiometers in each motor. The potentiometers are input directly to the analogue inputs of the BrainStem Moto 1.0 that provides a 10-bit analogue-to-digital conversion of the potentiometer value. This value gives an indication of the position of the robot when it is powered up.

As these problems are overcome, further planned development of the system is to integrate it with RobotWorks, a computer aided manufacturing (CAM) package. RobotWorks is an add-on package for SolidWorks, which is a commonly used CAD program. The idea being that path planning for the manufacture of various parts and components is all done by RobotWorks. This will remove the complexity of working out programmed motions for the robot from the user and, in effect, creates a virtual manufacturing environment.

6. Conclusion & Future Work

The objective of this project was to use a Mechatronics systems approach to develop a modular, mechatronic, plug-and-play controller for the control of an n-axis robotic system. The actuating system of the six-axis, revolute PUMA 560 series robot arm was first investigated to gain an understanding of the mechanical, electronic and software system requirements. The system was then successfully developed and included the construction of a power supply for the robot and a generic modular controller for the robot actuating system. Interfacing of the controller with a PC was also developped so that the system could be successfully controlled and tested. This was all achieved at a low cost thanks to the mechatronics approach that was used.

The system still needs further development, and future work includes the development of the system so as to integrate it with a computer aided manufacturing (CAM) package. The project did, however, effectively demonstrate that a sound mechatronic systems approach could be used to develop a working, low-cost system and a relatively short time frame.

The project also demonstrated the flexibility of a modular plug-and-play mechatronic system as the modular controller cards were used to control both the Puma 560 control arm, and a CNC lathe with only software changes required to configure the cards to each machine.

7. References

Acroname. (1994). BrainStem: Moto. , *Available from:* http://www.acroname.com *Accessed:* 2003-11-14.

Bold, G.E.J. (2000). *Transistor Electronics. (5th Ed).* Auckland, University of Auckland.

Bolton, W. (1999). *Mechatronics: Electronic control systems in mechanical engineering. (2nd Ed).* Addison Wesley Longman, ISBN 0582357055, Malaysia.

Bradley, D., Seward, D., Dawson, D. & Burge, S. (2000). *Mechatronics and the design of intelligent machines and systems.* Stanley Thornes, ISBN 0748754431, United Kingdom.

Fu, K., Gonzalez, R. & Lee, C. (1984). *Robotics: Control, Sensing, Vision and Intelligence.* McGraw-Hill, ISBN 0070226253, Singapore.

Jones, D. (2001). Basic Stepping Motor Control Circuits. *Available from:* http://www.cs.uiowa.edu/~jones/step/circuits *Accessed:* 2003-11-14.

Krar, S. & Arthur, G. (2003). *Exploring Advanced Manufacturing Technologies.* Industrial Press, ISBN 0831131500, New York.

Robot Electronics. (2001). 20A H-Bridge MD03: MD03 Documentation. *Available from:* http://www.robot-electronics.co.uk, *Accessed:* 2003-11-14.

Robotics Research Group, (2003). Learn More: History. *Available from:* http://www.robotics.utexas.edu/rrg/learn_more/history, *Accessed:* 2003-11-10.

Stanford University: Computer Science Education. (1998). Robotics and Motion Planning. *Available from:* http://www-cse.stanford.edu/classes/sophomore-college/projects-98/robotics, *Accessed:* 2003-11-10.