Multi Robot Systems

Recent Advances in
Multi Robot Systems

Edited by
Aleksandar Lazinica

I-Tech

Published by I-Tech Education and Publishing

I-Tech Education and Publishing
Vienna
Austria

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

First published May 2008
Printed in Croatia

A catalogue record for this book is available from the Austrian Library.
Multi Robot Systems, Recent Advances, Edited by Aleksandar Lazinica
p. cm.
ISBN 978-3-902613-24-0
1. Multi Robot Systems. 2. Recent Advances. I. Aleksandar Lazinica

Preface

To design a team of robots which is able to perform given tasks is a great concern of many members of robotics community. There are many problems left to be solved in order to have the fully functional robot team. Robotics community is trying hard to solve such problems (navigation, task allocation, communication, adaptation, control,…).

This book represents the contributions of the top researchers in this field and will serve as a valuable tool for professionals in this interdisciplinary field.

It is focused on the challenging issues of team architectures, vehicle learning and adaptation, heterogeneous group control and cooperation, task selection, dynamic autonomy, mixed initiative, and human and robot team interaction.

The book consists of 16 chapters introducing both basic research and advanced developments. Topics covered include kinematics, dynamic analysis, accuracy, optimization design, modelling, simulation and control of multi robot systems.

This book is certainly a small sample of the research activity on Multi Robot Systems going on around the globe as you read it, but it surely covers a good deal of what has been done in the field recently, and as such it works as a valuable source for researchers interested in the involved subjects.

Special thanks to all authors, which have invested a great deal of time to write such interesting and high quality chapters.

Editor

Aleksandar Lazinica

Contents

1

A Networking Framework for Multi-Robot Coordination

Antonio Chella, Giuseppe Lo Re, Irene Macaluso, Marco Ortolani and Daniele Peri
Dipartimento di Ingegneria Informatica, Università di Palermo
Italy

1. Introduction

Autonomous robots operating in real environments need to be able to interact with a dynamic world populated with objects, people, and, in general, other agents.
The current generation of autonomous robots, such as the ASIMO robot by Honda or the QRIO by Sony, has showed impressive performances in mechanics and control of movements; moreover, recent literature reports encouraging results about the capability of such robots of representing themselves with respect to a dynamic external world, of planning future actions and of evaluating resulting situations in order to make new plans. However, when multiple robots are supposed to operate together, coordination and communication issues arise; while noteworthy results have been achieved with respect to the control of a single robot, novel issues arise when the actions of a robot influence another's behavior.
The increase in computational power available to systems nowadays makes it feasible, and even convenient, to organize them into a single distributed computing environment in order to exploit the synergy among different entities. This is especially true for robot teams, where cooperation is supposed to be the most natural scheme of operation, especially when robots are required to operate in highly constrained scenarios, such as inhospitable sites, remote sites, or indoor environments where strict constraints on intrusiveness must be respected.
In this case, computations will be inherently network-centric, and to solve the need for communication inside robot collectives, an efficient network infrastructure must be put into place; once a proper communication channel is established, multiple robots may benefit from the interaction with each other in order to achieve a common goal.
The framework presented in this paper adopts a composite networking architecture, in which a hybrid wireless network, composed by commonly available WiFi devices, and the more recently developed wireless sensor networks, operates as a whole in order both to provide a communication backbone for the robots and to extract useful information from the environment.
The ad-hoc WiFi backbone allows robots to exchange coordination information among themselves, while also carrying data measurements collected from surrounding environment, and useful for localization or mere data gathering purposes.

The proposed framework is called *RoboNet*, and extends a previously developed robotic tour guide application (Chella et al., 2007) in the context of a multi-robot application; our system allows a team of robots to enhance their perceptive capabilities through coordination obtained via a hybrid communication network; moreover, the same infrastructure allows robots to exchange information so as to coordinate their actions in order to achieve a global common goal.
The working scenario considered in this paper consists of a museum setting, where guided tours are to be automatically managed. The museum is arranged both chronologically and topographically, but the sequence of findings to be visited can be rearranged depending on user queries, making a sort of dynamic virtual labyrinth with various itineraries. Therefore, the robots are able to guide visitors both in prearranged tours and in interactive tours, built *in itinere* depending on the interaction with the visitor: robots are able to rebuild the virtual connection between findings and, consequently, the path to be followed.
This paper is organized as follows. Section 2 contains some background on multi-robot coordination, and Section 3 describes the underlying ideas and the motivation behind the proposed architecture, whose details are presented in Sections 4, 5, and 6. A realistic application scenario is described in Section 7, and finally our conclusions are drawn in Section 8.

2. Related Works

In the past years, various classifications of multi-robot systems have been proposed. Dudek, et al., for instance, have proposed a taxonomy on communication mechanism and their cost to highlight that different multi-robot systems have very different capabilities (Dudek et al., 1996). The taxonomy proposed by Dudek takes into account some criteria, such as the number of robots in the collective, the maximum distance between robots such that communication is still possible, the communication topology, the composition of the collective, and the computational model of individual robots. Some of the works presented in literature achieve coordination among robots through distributed control, as in the case of the Alliance architecture (Mataric, 1997), where a robot increases the utility measure for the task that it is currently accomplishing while it decreases it for all other tasks; each robot then observes the behavior of its team-mates and selects the fastest achievable task. On the other side, the MARTHA project (Alami et al., 1998) assumes a centralized control to coordinate a team of autonomous robots for transport application in structured environment.
Parker in (Parker, 2003) presented a review of the main topic areas of research regarding multi-robot systems: biological inspired robot teams, communication, architectures and task planning, localization and mapping, object transport and manipulation, motion coordination, reconfigurable robotics, learning.
A large amount of research has been dedicated to the issue of communication in multi-robot systems. Several studies have been conducted to assess the benefits provided by communication on the performance of a robot team. Balch and Arkin conducted experiments with robots equipped with LED indicators signaling the state they were in (Balch & Arkin, 1994). The results indicated that communication considerably improves system performance. Simple communication strategies are preferable, because more complex approaches do not significantly improve results.
Mataric (Mataric, 1998) used communication to share data between robots in order to compensate for the limitations of direct sensory modalities. The proposed networking

framework allows the robots to enhance their perceptive capabilities by sharing the knowledge they own or the information provided by the sensor network. The features of such networks have been exploited, for instance, to allow each robot to detect people being beyond the range of robot sensors. This information is used to approach visitors and offer a guided tour of the museum.
Some of the works have focused on the issues related to fault-tolerance in multi-robot communication. Winfield developed ad-hoc wireless networking for collecting sensory data from a team of mobile robots (Winfield, 2000); the author also addressed the case where the ad-hoc wireless network is not fully connected, and rather it is partitioned into smaller sub-nets.
The scenario we are considering in the present work, however, involves only indoor communications, and an area that spans a building, so that full connection for the WiFi LAN can be reasonably assumed. Besides acting as a communication infrastructure for the robot team, the wireless LAN will also offer support for knowledge sharing; in particular, it will act as a backbone for exchanging information derived from locally deployed wireless sensor networks (WSNs).
Recently, this technology has been employed to tackle the task of closely monitoring and localizing moving objects in a structured environment (Akyildiz et al., 2002). Such networks are typically used for pervasive environmental monitoring through measurement of characteristic quantities, but each sensor node also has limited processing capabilities that may be exploited in order to carry on preliminary operations on raw data.
These features have been exploited by designing sensor nodes equipped with specialized hardware that enables them to compute a sufficiently accurate estimate of distances; such nodes have been successfully employed in the design of an indoor localization system (Priyantha et al., 2000). Such system may be employed as support for robots navigation and localization.
The synergy between wireless sensor networks and robotics has been analyzed for instance in (Moore et al., 2004), where the authors build a network of mobile sensors that can be controlled in order to collect samples of a distribution of interest, and also in (McMickell et al., 2003; Bergbreiter & Pister, 2003), where some general design, cost and scalability issues are discussed.

3. The RoboNet Framework

The design of the proposed framework has been mainly motivated by the experience developed in the context of the experiments conducted at the Archaeological Museum of Agrigento, Italy where findings from the close "Valley of the Temples", one of the UNESCO World Heritage Sites, are collected.
The purpose of the framework is the coordination of a group of robots moving in a structured indoor environment in order to manage automatically guided museum tours. Museum managers would also like to be able to provide virtual visits: tourists might thus be able to browse the exposed findings, for instance during off-peak hours, through web-based interface and could partially customize their visit by controlling the robots' actions. The quality of real visits, on the other hand, could be improved by careful planning, but this requires collection information and studying the tourist flows; moreover, it would be desirable to gather information also for surveillance purposes, so these are further *desiderata* for our framework. Finally, very tight requirements were posed by the museum board of

directors regarding severe limitations on the deployment of any intrusive hardware devices on the museum premises.

The RoboNet robot team may thus be regarded as a community of connected entities with the possibility of exchanging messages with each other, and of cooperating toward achieving a shared goal in order to find the solution to a common problem. Besides the benefits that are expected to emerge from cooperation, a major difficulty arises from the fact that all involved entities must achieve efficient coordination, while acting independently and autonomously from each other; moreover, an additional difficulty is represented by the need of providing fault tolerance to the whole system.

In our architecture, we assume that the role of coordinator is not statically assigned to a single external entity; rather, at a given moment in time, one robot will take up the role of team leader, with the possibility of releasing it in favor of one of the other components of the team.

Figure 1 shows the main components of the RoboNet framework, and represents the 3-tier architecture that has been devised to separate the main modules into functionally correlated layers. The lowest layer (the Communication Layer) is meant to provide basic connectivity by means of two different network technologies. A wireless sensor network is deployed in each room and will assist a close-by robot with self-localization tasks, by using a combination of radio frequency and ultra-sound signals, as will be explained later. Such WSNs do not form a connected network, and localization signals will not propagate across neighboring environments. Furthermore, additional wireless sensor nodes are present in the rooms; they are carried by visitors in order to provide a robot with proximity measurements, through computations performed based on RSSI signals. Those sensors will only need to be connected to a close-by robot that will thus compute a rough estimate of the distance of people the are supposedly following the tour it is guiding.

On the other hand, the WiFi backbone will provide connection among all robots, and will be used for exchanging messages related to coordination, or limited knowledge sharing.

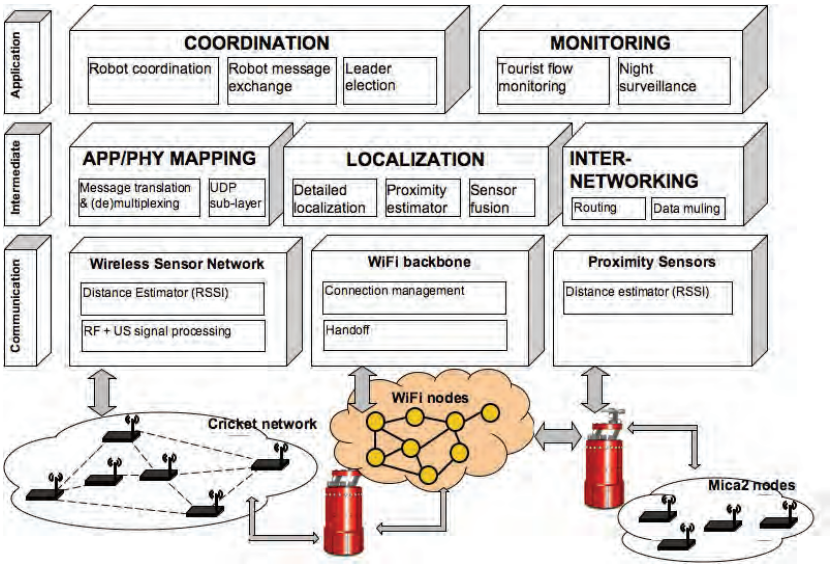


Figure 1: The RoboNet architecture

One layer up in the proposed architecture, the Intermediate Layer is where data previously collected are aggregated to extract information useful for localization or event detection purposes, such as signaling people abandoning the visit. This layer will also deal with specific networking issues, such as routing messages among robots, with special regard to messages to or from the coordinator entity.
Finally the uppermost layer consists of an application layer protocol that defines message formatting and exchanging. The algorithm ruling the asynchronous election of a leader among the robots in the team (in order to manage task assignments, among other things) will be implemented here.
The following sections will provide further details on each of the mentioned layers.

4. The Communication Layer

The lower layer includes the physical communication modules, which supervise the communications among robots and between robots and wireless sensor nodes. Moreover, this layer contains the modules necessary for the low-level functions of the localization and proximity networks.
A wireless LAN will act as a backbone and provide connectivity to the robots, while limiting modifications to the environment to the deployment of few access points, with no cabling, in order to adhere to the previously mentioned requirements; robots will use the standard IEEE 802.11a (WiFi) protocol to communicate among themselves, with the possibility of using broadcast or multicast addressing schemes, besides simple unicast. Remote access from the Internet to the functionalities provided by the system will also be possible through this backbone network.
A specialized wireless sensor network is thoroughly deployed to assist robots in their self-localization phase, according to the specifications of the Cricket project (Priyantha et al., 2000), this localization network is composed of Mica2 motes mounting an Atmega 128L 8-bit processor with 8 kBytes of RAM, 128 kBytes of FLASH ROM, and 4 kBytes of EEPROM and equipped with a CC1000 RF transceiver and an ultrasonic transmitter and receiver.
A few of those nodes are located at fixed positions in each sub-environment, while another one (named *listener*, in Cricket terminology) is carried by each moving robot. The same kind of nodes will also be carried by tourists, but, rather than acting as listeners, will only exploit their RF transceiver in order to provide proximity estimates.
In ideal conditions, the energy of the RF signal decreases with the square of the distance from the emitter, and this information could be used by a receiver to estimate its distance from the source of the signal as a function of the strength of the received signal. However, radio propagation is likely highly non-uniform in real environments, so received signal strength indicators (RSSIs) suffer from noisy measurements and distance predictions using signal strength are somewhat imprecise.
Nevertheless, proximity measures may still be used as an inexpensive means for the robots to approximately estimate how many tourists are "following" them and to plan the guided tour accordingly. Moreover, the same proximity sensors may be placed at specific spots (e.g. at each passage between contiguous sub-environments) in order to gather statistics about tourist flows by storing the IDs they sense over time for surveillance purposes.

Since robots are also equipped with standard WiFi cards in order to communicate with the above mentioned backbone network, they can communicate with Mica2 nodes on one side, and with the WiFi access points on the other, and are thus natural candidates for harvesting data from proximity sensor nodes (which do not form a connected network).

5. The Intermediate Layer

Data originating from the different kinds of networks described above will be processed at the middle layer of the RoboNet architecture, in order to convert application-layer messages into physical-layer ones, also adapting them to the requirements of the specific network that will carry them, to assist robots in the localization phase, and to extract proximity information about visitors.

Our framework achieves coordination among robots by a specifically designed application-layer protocol whose messages are carried by the WiFi backbone.

Application-layer messages will travel over UDP, managed by a simple loss recovery mechanism implemented as a thin sub-layer; using TCP would have led to unacceptable latency, and bare UDP would have suffered from the high loss rates typical of a wireless network such as the one considered here. After a message is sent, an ACK is expected and, if a pre-defined timeout expires before the ACK is received, the sender tries a retransmission and waits for twice the timeout; in case of further failure, a loss is simply notified to the upper layer.

Localization is realized by combining data coming from the analysis of internal robots' parameters and from the Cricket infrastructure. As described in (Balakrishnan et al., 2003), the most recent version of the Cricket framework implements a Kalman filter to assist a moving device in tracking its position. Cricket mobile nodes initially estimate distances from nodes at known positions by measuring the time difference of arrival between RF and ultrasound signals; the Kalman filter uses a predictor to estimate of the node's current position and corrects this estimate taking into account the difference between the predicted and the sensed distance. A covariance matrix reflects the filter's confidence in the state vector.

Following the directions of (Smith et al., 2004), we provide each moving robot with an *active* localization device (the listener), so beacon nodes, whose location is known, estimate distances to the listener based on an active transmission from the listener itself.

Since a mobile device in the active mobile architecture sends simultaneous distance estimates to multiple receivers, its performance is arguably better than with the passive mobile system in which the listener obtains only one distance estimate at a time and may have moved between successive estimates.

As the listener is bolted to the robot whose movements are constrained in two dimensions, we modify the original EKF used to track the mobile device (Smith et al., 2004) considering a state vector composed by the two locations coordinates in the plane and the heading direction (θ). Moreover to take into account the information provided by the control data (i.e. the translational velocity v_t, and rotational velocity ω_t applied to control the robot), we adapted the prediction step of the EKF including a velocity motion model (Thrun et al., 2005). The state prediction at time t is:

$$\overline{\boldsymbol{\mu}}_t = \boldsymbol{\mu}_{t-1} + \begin{pmatrix} -\frac{\hat{v}_t}{\hat{\omega}_t}\sin(\vartheta) + \frac{\hat{v}_t}{\hat{\omega}_t}\sin(\vartheta + \hat{\omega}_t \Delta t) \\ \frac{\hat{v}_t}{\hat{\omega}_t}\cos(\vartheta) - \frac{\hat{v}_t}{\hat{\omega}_t}\cos(\vartheta + \hat{\omega}_t \Delta t) \\ \hat{\omega}_t \Delta t \end{pmatrix} \quad (1)$$

where $\overline{\boldsymbol{\mu}}_t$ is the predicted state at time *t* and $\boldsymbol{\mu}_{t-1}$ is the state vector at time *t-1*; $\hat{v}_t$ and $\hat{\omega}_t$ are the translational velocity and the rotational velocity respectively, generated adding Gaussian noise to the motion control $\mathbf{u_t} = (v_t\ \omega_t)^T$.
The estimated pose provided by the EKF is integrated with the one generated using a previously developed particle filter algorithm (Thrun, et al., 2005).
In particular the posterior distribution of the robot pose is computed as a weighted sum:

$$p(x_t \mid z_t, u_t) = w_{wsn,t}\, p(x_t \mid z_{wsn,t}, u_t) + w_{rs,t}\, p(x_t \mid z_{rs,t}, u_t) \quad (2)$$

where $p(x_t|z_{wsn,t},\ u_t)$ is the normal posterior distribution computed with the EKF taking into account measurements provided by the WSN ($z_{wsn,t}$) and $p(x_t|z_{rs,t},\ u_t)$ is the posterior distribution computed by the particle filter given the robot sensors measurements ($z_{rs,t}$).
The weights $w_{wsn,t}$ and $w_{rs,t}$ are a measure of the uncertainty of the corresponding pose estimates. In particular $w_{wsn,t}$ is the likelihood $p(z_{rs}|\ x_t)$ of the measurement model (Thrun et al., 2005), while $w_{rs,t}$ is computed as the inverse of the trace of the covariance Σ_t of the posterior distribution estimated by the Kalman filter.
The original localization algorithm may thus be customized to our scenario allowing better performance.

6. The Application Layer

Robots belonging to the team need to agree on an initial representation of the world they operate in; moreover, they are not supposed to perform tasks independently from each other, and need some degree of coordination. The uppermost layer of the proposed architecture deals with providing the robots with proper representation of the environment, and implements a robust task assignment algorithm.
The present work builds upon a previous experience on the same topic (Chella, et al., 2007); in the referenced work, the proposed coordination mechanism relied on a central coordination unit that contained a complete knowledge of the environment and supervised the task assignment job through an auction mechanism. On the other hand, in the current version we abandoned this centralized approach as it introduced a potential bottleneck and a single point-of-failure, and instead we devised a fully distributed control where each robot is a potential coordinator, and this role will be assigned dynamically through a simple election mechanism.
We assume the backbone network is fully connected, which appears reasonable given the communication range of WiFi devices and the relatively short distances of an indoor environment. At the Intermediate Layer, the internetworking subsystem, transparently to

the robots, builds a communication structure that allows addressing each of the robots singularly; additionally, multicast addressing is possible.
At the Application Layer, communications among robots occur via the backbone network that acts as a bus, and a simple algorithm is implemented for electing one of the robots as the temporary coordinator, following well-known techniques from related literature (Santoro, 2006). Roughly speaking, the algorithm guarantees that, at any given time, exactly one of the robots will act as the coordinator for the whole. When a leader must be elected, all robots will broadcast a "coordinator election request" message on the communication network; each message will contain the sender's identification number, and the highest ID will eventually be used to have all robots agree on an elected coordinator. As will become clear in Section 6.1, the particular addressing scheme employed in our framework allows for some flexibility with respect to each robot's ability of acquiring and releasing the coordinator role.
Such algorithm is clearly fully decentralized, and it can be proved that it is also asynchronous and fault-tolerant. Further details can be found in (Santoro, 2006).
In the following, the coordinator will be indicated by the term Robot Team Leader (RTL); it is important to point out that this role is not statically assigned to one of the robots, but will be taken up by several of them in the course of operations.
For the purpose of the operations of the RTL, the environment is modeled as a topological map representing the connectivity and accessibility of the different regions in the environment, similarly to what described in (Chella, et al., 2007). The environment is thus split into a collection of sub-environments connected by passages, and the relations of connectivity between the different sub-environments are captured by a connectivity graph representation: a node corresponds to a sub-environment; each sub-environment is univocally identified; an arc between two nodes exists if the corresponding sub-environments are connected.
The condition of accessibility for each robot is modeled by partitioning the connectivity graph and considering the subgraph representing regions reachable by the robot.
A visit is described in the model by the sequence of sub-environments to be showed to visitors. For the reasons stated above, a visit generally needs to be split into a sequence of sub-visits each one guided by a robot. The RTL governs the assignment of sub-visits to robots by an auction mechanism.
When a guiding service is needed the RTL is notified by either the ticket counter or a robot, with the former case happening only once at the beginning of a visit, while the latter may arise several times during the visit. In fact, every time a robot is not able to reach the next sub-environment, another robot must be found to lead the visitors group throughout the remaining part of the visit.
When a robot decides to cease guiding the group, for instance because it cannot physically reach the next sub-environment, it communicates the remaining part of the visit to the RTL. This mechanism allows for the inclusion of reasons other than the sub-environment connectivity into the management of guided tours; for instance drawn batteries as well as other malfunctions could be easily circumvented, given other robots were allocated to the same sub-environment. In practice, a request from a robot guide that ceased a visit is identical to those issued by the ticket counter for a new visit and both are encoded with the same message kind in the protocol.

When the RTL receives such a message it starts the auction by multicasting the request for a task to the robots able to reach the first sub-environment of the visit. The robots then reply to the RTL with their bid expressed as an estimate of the time needed to start the requested visit. This estimates accounts for the time to get to the starting point as well as the remaining time of the current visit, if any.
The current RTL collects the bids and sends the robot with the best bid a task assignment message. Encoded in the message, beside the sequence of sub-environments to be shown, are the IDs of the visitors. The receiver can then reply to accept the task thus ending the auction, otherwise another bidder must be chosen by the RTL.
A visit may end naturally when the last sub-environment has been visited or when the robot guide senses no visitors in its proximity.
The leadership alternation approach we adopted allows to carry out an intrinsically fault-tolerant system with respect to the failures of a single central unit, while in the meantime the benefits of a centralized coordination approach are maintained.
Each robot is able to perform as a leader as soon as it is selected. To this aim, both the topological map of the whole environment and its partitions (representing the accessibility of sub-environments for each robot) are initially provided to all the robots and represent the *innate* knowledge of the system.
The proposed framework has been designed to allow for an easy extension process of the innate system knowledge with new information acquired during run-time. In particular, we analyzed the possibility of modifying the auction mechanism by sending the request for a task assignment to a subset of the robots that are able to perform it. Such subset can be selected by performing a statistic of the past bids made from the robots in a similar case.
The evolution of the system knowledge through learning, combined with the mechanism of leader alternation at the same time point out the issue of knowledge sharing. To this aim the proposed framework allows to verify different approaches:

- knowledge sharing and updating every time the leader changes;
- no knowledge sharing.

In the first case, whenever a new leader has to be elected, the current leader broadcasts the new information to all the robots, so achieving a common updated and consistent knowledge representation.
The second approach will lead to different leaders, depending on the information they received during their leadership. This approach may for instance allow for easy checking of the performances of different learning algorithms as a function of the dynamics of the acquisition of new knowledge about the environment in the course of time.
Moreover this approach allows to bound the drawbacks due to the over fitting produced by some learning algorithms. For example one leader can decide to leave out a robot from the auction related to a certain task. The diffusion of such information could produce the erroneous exclusion of this robot from future auction even if the temporary failure that affected the robot has been solved.
Although such issues can be addressed by performing a periodic query of the robots state, the local nature of certain information allows to naturally fit the system to variable conditions. However a global update of the system knowledge can be periodically performed to provide each robot with the possibility of evolving and growing its own knowledge.

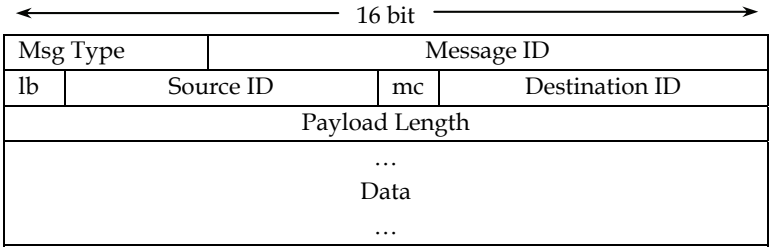


Figure 2: The message format

6.1 The Message Exchange Protocol

We devised an application-layer protocol, adapted with minimal changes from (Chella, et al., 2007) in order to allow the RTL to send and receive requests from robots and the robots to communicate with each other. Messages will travel on the WiFi network and have the format specified in Figure 2, which shows a fixed 6-bytes header followed by a variable length payload.

Message Type	Description
`LDR_SEL`	used during the leader election phase
`RBT_PING`	ping message to check for liveliness
`RBT_PONG`	response to ping
`SVC_SIG`	used to signal the RTL about a new visit to be assigned
`SVC_REQ`	used by RTL for starting the auction
`SVC_REPLY`	used by a robot to participate in the auction, and signal its service response time
`TASK_ASS`	used by RTL to assign a task to a robot
`TASK_ACC`	used by a robot to confirm acceptance of a task
`SYNC_REQ`	used by a robot to exchange its state with another robot
`SYNC_REPLY`	reply to `SYNC_REQ`

Table 1: Message types

The semantics of carried data depend on the specific message type that is specified in the 4-bit "Msg Type" field; this is more than sufficient to identify one of the different types we are currently using for messages, as will be detailed in the following. Messages traveling across the network will be uniquely identified by the combination of a 12-bit "Message ID" and of a "Source ID".

Each robot is statically assigned an ID, which will be also used to identify the source of a message, indicated in the "Source ID" field as a 7-bit integer (the first bit is reserved for the leader election mechanism). The next 8 bits carry the message destination information; this field is 7-bit long, with 1 preceding bit reserved for multicast groups management; when set to 1, it indicates a multicast address (in order to identify specific groups of robots), while referring to individual robots otherwise. This "Destination ID" field contains the unique identifier of any addressable entity in the RoboNet framework, namely every individual robot or any predefined group of robots. The address `0000000` is reserved and will be used in certain circumstance by the RTL to identify itself instead of its natural address.

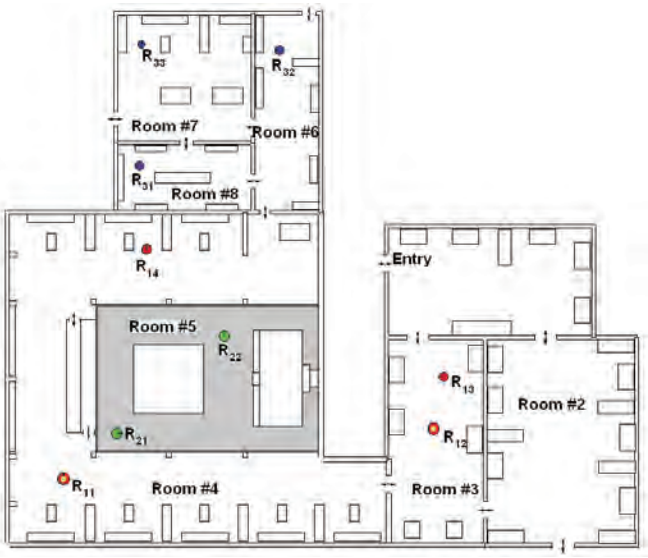
We allow for a maximum of 2^7-1 different *RobotID*'s, thus limiting to 127 the amount of robots in the environment, a number that appears sufficient for any practical purpose.

Unique IDs are also assigned to rooms and passages between rooms; moreover, as each visitor will be provided with a mote acting as a tracking device, they may also be uniquely identified through that device's ID; such ID's will be stored in a 16-bit long field to be carried in the payload of the message; the maximum amount of elements for those *IDList*'s is only limited by the message payload length, specified in the relative field. In the following, such ID's will be indicated respectively as *RoomID*, *PassageID*, and *VisitorID*.
As a group forms at the entrance, a list of *VisitorID*'s is created and is transmitted to the current RTL in a `SVC_SIG`; RTL will then broadcast a `SVC_REQ` message to all robots through the WiFi backbone. The message will include the room where service is required (i.e. the starting room for the tour). Each robot will reply with a `SVC_REPLY` message containing an estimate for its service time, as explained above. RTL will then assign the task to one of the robots whose reply has been received and that provided the best bid, and the corresponding `TASK_ASS` message contains the list of *VisitorID* and the list of *RoomID* (i.e. the description of the "visit"). The selected robot will finally signal its acceptance by sending a `TASK_ASS` message to RTL.
Finally, the `LDR_SEL` message is used during the leader election phase. This message will be broadcast by each robot periodically, either spontaneously or upon reception of an analogous message from one of its fellows. The robot with the highest ID will eventually be selected as the leader, and this information will be automatically shared by the whole team.
The current leader, after a fixed period of time, broadcasts a `LDR_SEL` message to all the other robots to signal its will to resign from its role in favor of a potential new leader, and will thus trigger a new election. We assume that each robot knows the IDs of the other participants to the team, so that after receiving all the answers it can infer whether it won the race to leadership or not. A timeout mechanism provides robustness: if a robot believes it is next elected leader, but has not received answers from some other robot within the timeout, it will simply check their status through a `RBT_PING` message, and will consider them "dead" for the current round. After a grace period, necessary to the old leader to end any previous assignments, the new elected leader will be effectively operating.
This mechanism is RTL-driven, so it could be prone to error as a consequence of a fault in the current RTL; for this reason, the protocol assumes that each robot periodically pings the leader to check if it is still alive and functioning. Upon detection of a fault, the robot that noticed the anomaly independently starts the leader election mechanism.
As previously mentioned, the current RTL is identified by the `000 0000` reserved address, which would "force" it to resign from its role, as a consequence of the "highest ID wins" rule. The "leader boost" bit preceding the Source ID field may be used to artificially modify the natural address in order to allow for some flexibility in the election mechanism, according to a policy that can also be defined at the application layer. An example of this will be given with reference to the application scenario described later on; for instance, the current RTL may force its address to `1000 0000` in order to signal its availability to maintain the leadership. Analogous behavior is implemented for non-leader robots.
The complete set of messages is shown in Table 1, together with a brief description.

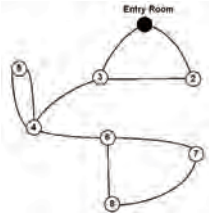
7. An Application Scenario

In the course of the years, the Robotics Lab of University of Palermo developed a robotic architecture that takes into account several suggestions from cognitive science.

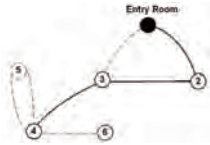
The architecture has been successfully tested in the CiceRobot project on tasks related to guided tours in the Archaeological Museum of Agrigento (Chella & Macaluso, 2006; Chella et al., 2007). The robot was able to modify the predefined paths through the museum by rearranging the sequence of findings to be visited depending on user queries.
For the purpose of the current research, it is important to point out that the robot planning system is based on a 3D Robot/Environment Simulator. The planning by simulation paradigm allows to easily and carefully perform those forms of planning that are more directly related to perceptual information. In fact, the preconditions of an action can be simply verified by geometric inspections in simulation, avoiding the verification by means of logical inferences on symbolic assertions; also the effects of an action are not described by adding or deleting symbolic assertions, but they can be easily described by the situation resulting from the expectations of the execution of the action itself in the simulator.
The proposed architecture has been deployed in the Archaeological Museum of Agrigento. As previously said, each robot is able to guide visitors both in a prearranged tour and in an interactive tour. Let us consider the multi-robot coordination in our experimental setup.



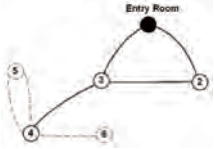
(a) Map of the environment.



(b) Graph representation of the whole environment.



(c) Graph representation of the environment for Robot R_{13}.



(d) Graph representation of the environment for Robot R_{12}.

Figure 3: The site described in the application scenario

Supposing that a group of visitors is waiting at the entrance; the RTL multi-casts a `SVC_REQ` message to those robots that are able to reach the entry (red and yellow/red robots in Figure 3). Each robot replies with its estimated performance relative to the current task, i.e. the time it takes to reach the entry. Through the 3D Robot/Environment simulator each robot can imagine itself going through the environment to reach the current target: the simulated interaction between the robot and the environment allows to easily compute an accurate estimate of the robot performance. In the example reported in Figure 3 both robots R_{11} and R_{14} are busy guiding visitors, while robots R_{12} and R_{13} are idle. Even if R_{13} is the nearest robot to the entry,

the RTL allocates the task to the robot R_{12} as its performance estimate is better. Actually R_{13} is not able to directly go from *Room* #3 to *Entry Room* due to a slope that is too steep for it, but not so for R_{12}. During the visit, as R_{12} is about to complete the sub-visit it has been assigned, it sends a SVC_SIG message to the RTL to request another task allocation for the group.
We also performed several tests on a simulated environment to evaluate the performances of the leadership alternation approach.
It should be pointed out that the leader is functionally identical to the other robots, except for the coordination burden it must deal with. Therefore, the leader too participates to every auction for tasks assignment; anyway to reduce its computational load, the bid proposed by the leader is incremented by an additional cost. This way we reduced the number of tasks the leader will self-assign. Moreover, in agreement with the need of reducing the leader computational load, the participation to the election phase can be restricted to the robots that are currently idle; they may reinforce their proposal for leadership by setting the "leader boost" bit in their address while sending the LDR_SEL message. No conflicts may arise among robots competing for the leadership, as ties will be automatically broken by the use of unique addresses.
In some cases, the combination of the two mentioned mechanisms resulted in the unwanted persistence of the same leader. In order to overcome this drawback, the additional cost related to the leader tender is implemented as a linear decreasing function of time.
Fault tolerance is achieved in this setting through the periodic check on the leader status performed by all robots, at the cost of a minimal traffic overhead introduced by the control messages.
At the moment of writing, the experiments were performed assuming no knowledge sharing was carried out by robots. More elaborated scenarios could be devised so that the information acquired so far by the current leader is passed onto its successor and merged in order to achieve a common updated and consistent knowledge representation.

7. Conclusion

This paper presented a framework for the coordination of a group of robots moving in a structured indoor environment in order to manage automatically guided museum tours.
The design of a hybrid wireless networking architecture, composed by WiFi devices inter-operating with wireless sensor nodes has been discussed, and it has been shown how it can operate as a whole in order both to provide a communication backbone for the robots, and to extract useful information from the environment.
The robustness of the communication protocol implemented in the proposed framework has been enforced through a fault-tolerant leader election mechanism, which allows for an easy extension process of the innate system knowledge with new information acquired during run-time.
Experiments have been carried on in the context of the RoboNet project conducted at the Archaeological Museum of Agrigento, Italy, and the proposed coordination mechanisms have been tested through simulations.

8. References

Akyildiz, I.; Su, W.; Sankarasubramaniam, Y. & Cayirci, E. (2002). A survey on sensor networks. *IEEE Communication Magazine*, vol. 40, n. 8, August 2002, pp. 102–114.

Alami, R.; Fleury, S.; Herrb, M.; Ingrand, F. & Robert, F. (1998). Multi-robot cooperation in the MARTHA project. *Robotics & Automation Magazine, IEEE*, vol. 5, no. 1, pp. 36–47.

Balch, T.; Arkin, R. C. Communication in reactive multiagent robotic systems. Autonomous Robots 1:1-25. (1994)

Balakrishnan, H.; Baliga, R.; Curtis, D.; Goraczko, M.; Miu, A.; Priyantha, N. B.; Smith, A.; Steele, K.; Teller, S. & Wang., K. (2003). Lessons from developing and deploying the cricket indoor location system. *Technical report, MIT Computer Science and AI Lab.*

Bergbreiter, S. & Pister, K. S. J. (2003) Cotsbots: An off-the-shelf platform for distributed robotics. *Proc. of the IEEE Int. Conf. on Robotics and Automation*, Las Vegas, NE, USA.

Chella, A.; Liotta, M. & Macaluso, I. (2007). Cicerobot - A cognitive robot for interactive museum tours. *Industrial Robot: An International Journal*, vol. 34, pp. 503-511 ISSN: 0143-991X.

Chella, A.; Lo Re, G.; Macaluso, I.; Ortolani, M. & Peri, D. (2007) Multi-robot Interacting Through Wireless Sensor Networks. *Lecture Notes in Computer Science*, vol. 4733, pp. 789-796, 2007. ISSN: 0302-9743.

Chella, A. & Macaluso, I. (2006) Sensations and Perceptions in "Cicerobot" a Museum Guide Robot. *Proc. of BICS 2006 Symposium on Brain Inspired Cognitive systems*, Greece.

Dudek, G.; Jenkin, M.; Milios, E. & Wilkes, D. (1996) A taxonomy for multi-agent robotics. *Autonomous Robots*, vol. , no. 4, pp 375–397.

Mataric, M. J. (1997) Behaviour-based control: examples from navigation, learning, and group behaviour. *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 9, no. 2, pp. 323–336.

Mataric, Maja J. (1998) Using Communication to Reduce Locality in Distributed Multi-Agent Learning, *Journal of Experimental and Theoretical Artificial Intelligence*, special issue on Learning in DAI Systems, Gerhard Weiss, ed., vol 10, no. 3, pp. 357–369.

McMickell, M. B.; Goodwine, B. & Montestruque, L. A. (2003) Micabot: A robotic platform for large-scale distributed robotics. *Proc. of the IEEE Int. Conf. on Robotics and Automation*, Taipei, Taiwan.

Moore, K. L.; Chen, Y. Q. & Song, Z. (2004) Diffusion based path planning in mobile actuator-sensor networks (mas-net) - some preliminary results. *Proc. of SPIE Conf. on Intelligent Computing: Theory and Applications II, part of SPIEs Defense and Security*, Orlando, FL, USA.

Parker, L. E. (2003) Current Research in Multirobot Systems, *Artificial Life and Robotics*, vol. 7, no. 2-3.

Priyantha, N. B.; Chakraborty, A. & Balakrishnan, H. (2000) The cricket location-support system. *Proc. of the Sixth Annual ACM International Conference on Mobile Computing and Networking (MOBICOM)*, Boston, MA, USA.

Santoro, N. (2006) Design and Analysis of Distributed Algorithms (Wiley Series on Parallel and Distributed Computing. ISBN: 978-0-471-71997-7.

Smith, A.; Balakrishnan, H.; Goraczko, M. & Priyantha, N. B. (2004) Tracking moving devices with the cricket location system. *Proc. of the 2nd ACM MobiSys*, pages 190–202, Boston, MA, USA.

Thrun, S.; Burgard, W. & Fox, D. (2005) Probabilistic robotics. MIT Press.

Winfield, A. (2000) Distributed sensing and data collection via broken ad hoc wireless connected networks of mobile robots. In: Parker L, Bekey G, Barhen J (eds) *Distributed autonomous robotic systems*, vol 4. Springer-Verlag, Tokyo, pp 273-282

2

An Empirical Study on Ecological Interface Design for Multiple Robot Operations: Feasibility, Efficacy, and Issues

Hiroshi Furukawa
University of Tsukuba
Japan

1. Introduction

Multiple robots systems are one of effective solutions to implement robust, flexible, and adaptable systems that can be used in various conditions (Lee et al., 2000). The systems are increasingly being used in environments that are inaccessible or dangerous to humans (Stoeter et al., 2002 ; Chaimowicz et al, 2005). Operations in these environments are numerous, and include reconnaissance, exploration, and surveillance. As an example, one of promising areas of the application is a network robots system, which is a distributed architecture for remote control of multiple robots systems (Wirz et al., 2006). Another example is swarms of insect robots (Lee et al., 2000). Large amount of small and simple robots are used to achieve tasks in a form of agent-based automation. Regardless of their purpose of use, there are several factors which pose a challenging problem in supervision and management of multiple robots.

The operator's task involves not only manipulation of each robot but also achievement of the top goal that has been assigned to the entire team of humans and robots. Clearly, support of the operator's skill-based behaviours is important. Equally, it is important to support human operators in their understanding of the overall state of a work-in-progress and the situation around it using a system-centred view. Although cognitive resources of humans are limited, operators are demanded to understand highly complex states and make appropriate decisions in dynamic environment. Furthermore, human-machine interfaces (HMIs) can display large amounts of complex information which risk overwhelming the operator at exactly the worst time, i.e., in an emergency situation (Sheridan, 2000). As a consequence, there has been increased interest in developing human-robot interfaces (HRIs) for human supervision of multiple robots (Goodrich et al., 2005).

The main goal of our research project is the development of an interface design concept based on ecological interface design (EID) for human supervision of a robot team. EID is a design paradigm based on visualization of constraints in work environment onto the interface to reduce the cognitive workload during state comprehension (Vicente, 1999; Vicente & Rasmussen, 1992; Vicente, 2002). EID provides information about states of functions that are necessary to achieve the top goal of a human-machine system.

Information on function is identified using the abstraction-decomposition space (ADS) (Rasmussen, 1986). An ADS is a framework for representing the functional structures of

work in a human-machine system that describes hierarchical relationships between the top goal and physical components with multiple viewpoints, such as abstraction and aggregation. Since the operator's comprehension of the functional states based on the ADS is an essential view for the work, supporting the view is crucial for operators to control a human-machine system, comprehend system states, make operational plans, and execute the plans appropriately under abnormal or unanticipated conditions (Miller & Vicente, 2001). EID is a design paradigm in which the ADS of a target system is represented in such a way as to allow operators to comprehend the states of the functions intuitively. The function-based HMI is designed to enable operators to develop a system-centred view even under high-workload conditions. This can be thought of as an externalization onto the HMI of the operator's mental model of the system (Rasmussen & Pejtersen, 1995). However several attempts have been made to apply the design paradigm to HRI, empirical evidence of the effectiveness of this approach, while necessary, is not sufficient.

In this study, the EID paradigm was used as the basic framework for implementing the information about a human-robot team work into an interface display. This chapter describes two experiments conducted to reveal the basic efficacy of EID in human-robot interactions, and the development of a design method using a multi-level representation of functions to improve human-robot collaboration.

The first experiment, Experiment 1, was conducted to reveal the feasibility of the proposed concept using an experimental test-bed simulation, as the first step of the project (Furukawa, 2006). The results indicated that the whole work can be modelled using ADS, and it is feasible to design useful functional indications based on the ADS. The results also show the need to consider participants' strategies developed for tasks to evaluate the effectiveness of HRI display. However, because the operation tasks were not complex, only a few strategies were used by the participants in the experiment.

The aim of the second experiment, Experiment 2, was to evaluate the basic efficacy of the human-robot interface design concept under the condition that the wide variety of strategies was used for operations (Furukawa, 2008). The conditions of the test-bed simulation had been changed to prompt the participants to develop various strategies. The results demonstrate that the designed interface design has basic efficacy to provide adequate and useful functional information for supporting human operators for supervision of a robot team.

2. Related Works

This section shows several proposed methods that used the EID paradigm for robot operations, and discuss the promising usages in designing of HRI for multi-robots systems.

Sawaragi and his colleagues applied the EID concept to HRI to support naturalistic collaboration between a human and a robot at the skill level via a 3D display (Sawaragi et al., 2000). Nielsen et al. proposed an ecological interface paradigm for teleoperation of robots, which combines video, map, and robot-position information into a 3-D mixed reality display (Nielsen et al., 2007). In both studies, the target level of operation is limited to skill-based control on a robot. The proposed methods can be used for direct operation of an individual robot in a multi-robots system. Jin and Rothrock propose a function-based interface display which indicates the state of communication between human operators and multiple Robots, in which the target function is limited to one (Jin & Rothrock, 2005). The indication can be used as a display design for representation of a function of a multi-robots system. On the other hand,

the proposed paradigm of the HRI design described in this chapter is a function-based display that indicates the *whole work* assigned to humans and *multiple robots*.
In a study conducted by Xu, an ADS is adopted as an analytical tool to identify problems and suggest opportunities for enhancing a complex system with agent-based automation. ADS models of human-machine systems indicate incomplete knowledge of human operators, lack of information displayed in interfaces, lack of procedures of operators, and so on. This analytical method can be used to evaluate appropriateness of HRI for multiple-robots systems.

3. Display Design based on Ecological Interface Design Paradigm

3.1 The RoboFlag Simulation Platform

This study uses the RoboFlag simulation, which is an experimental test-bed modelled on real robotic hardware (Campbell et al., 2003). The chief goal of an operator' job is to take flags using home robots and to return to the home zone faster than the opponents. One operator directs a team of robots to enter an opponent's territory, capture the flag, and return to their home zone without losing the flag. Defensive action takes the following form: while in the rival's territory, a rival robot can inactivate an intruding robot by bumping into it. Similarly, a rival robot will be inactivated if it is hit by a friendly robot while in friendly territory.
Fig. 1 shows the display used for an operator to monitor and control his or her own team of robots (Campbell et al., 2003). A circle around a robot indicates the detection range within which the robot can detect opponents and obstacles.

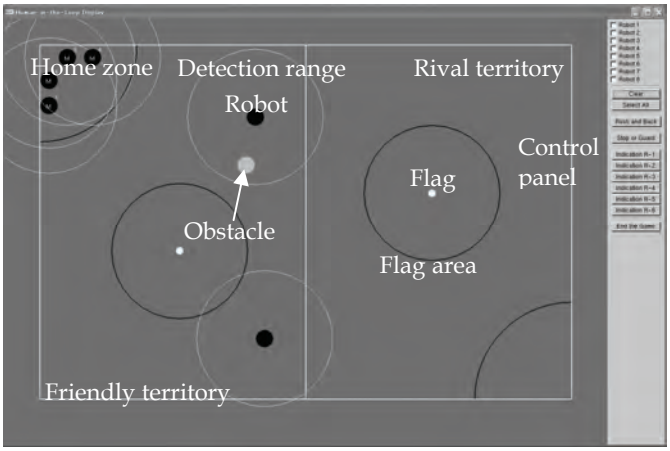


Figure 1. The original display for the RoboFlag simulation

The simulation provides two types of operations that operators can select according to their situation: manual controls and automatic controls. In manual control mode, an operator indicates a waypoint to a robot by clicking the point on a display. Two types of automatic controls were implemented in this study. When *Rush and Back* (R&B) mode is assigned, the robot tries to reach the flag and returns home after it captures the flag. The course selected is straightforward, in that the robot heads directly to the destination. In *Stop or Guard* (S/G) mode, the robot stays in the home position until it detects an opponent. If an opponent robot comes into detection range, the robot tries to inactivate the opponent. The robots are semi-

autonomous, that is to say, they have the ability to make fine adjustment to their own course to avoid rival robots or obstacles near the original course.
The basic tasks for achieving the chief goal are two: *Offence* and *Defence*. The former comprises two sub-tasks, which are *Capturing the flag* and *Taking it to one's own home zone*. *To keep the robot active* is also necessary for the sub-tasks. The sub-tasks of the latter are to prevent opponent robots from coming close to the flag and returning home with it.
Because time constraints are severe in this RoboFlag game, human operators need gain an understanding of the situation as rapidly as possible. Furthermore, it is necessary for operators to comprehend the state of entire area as well as the local area.

3.2 Design of Ecological Interface for Human-Robot Systems

The definition of human-robot systems in this chapter is illustrated in Fig. 2. The system consists of four agents: *Top Goals*, *Work Environment*, *Robots* and a *Human Operator*. Top goals are settled by the designer of the system. The robots, sometimes also environment, are designed to match these goals, and operators are trained to achieve these goals. Arrows with "Information" indicates flows of information to comprehend other agents, and "Control" indicates flows of signals to control other agents. "Negotiation" indicates settlement of interference in goals or means between robots and an operator. This framework is a general form, which just shows possibility of existence of each flow. There may be a situation that some of them are not included in real systems.
Top goals are functions that are defined on a human-machine system with quantified or qualified criteria. The goals are classified to two groups: positive goals and negative goals. The formers are reasons for existing of the system, and correspond to its beneficial influences to the outside world, e.g., exploration of moon surface. The latter correspond to a task to prevent bad influences to the outside world, e.g., collision avoidance.

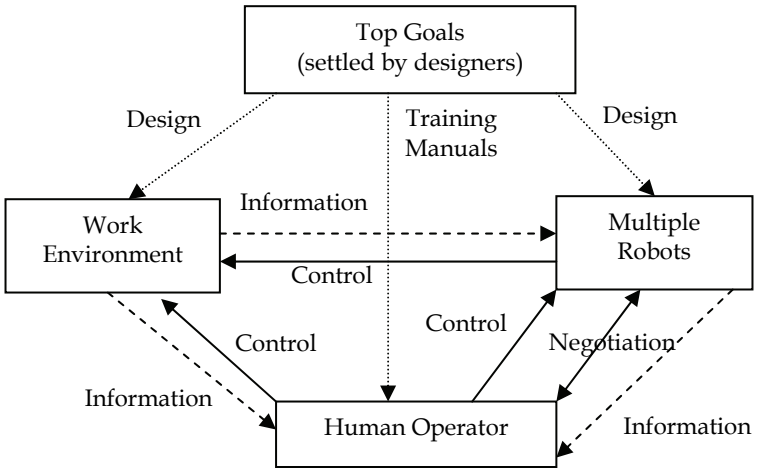


Figure 2. An overview of of multiple robots systems

Many researches have reported the effectiveness of means-ends models which functions are defined as primitive elements of controlled artefact (Vicente & Rasmussen, 1992; Burns & Hajdukiewicz, 2004). The ADS is the basic concept of the means-ends models, which is also known as the abstraction hierarchy. A design concept based on the ADS is EID (Vicente &

Rasmussen, 1992), in which the ADS of a target system, i.e., the means-end relations of the work, is represented to allow operators to comprehend the ADS intuitively. The HMI is also designed to support skill-, rule- and knowledge-based behaviours (Rasmussen, 1986).
In this study, an ADS was introduced as a basic framework for state representation of the multiple robots system, and the HRI was designed according to the EID paradigm to support operators to comprehend the states of functions dedicated to the controlled artefact.

3.3 Implementation of Functional Indications for Multi-Robot Operations

This section shows the human-robot interface for the RoboFlag simulation used in this experimental study and the design process of the functional indications based on the proposed design concept.
The following are descriptions of four function-based interface designs, in which one was designed to represent the state of a lower-level sub-function under an Offence function, and the second under a Defence function. The third and forth were designed for common sub-functions under the two functions. Previous studies using the RoboFlag simulation showed that human-robot interactions depend on various contributory factors (Parasuraman et al., 2005). These four functions were selected because results from the previous studies indicated that it was difficult to comprehend the states of the four functions during plays.
To specify each state of the function, expressions that graphically showed the state in the physical relations between each robot and the object was used. This has aimed to enable operators intuitive to understand the state of the functions and relationships between the functions.

3.3.1 A Functional Indication for Offensive Function

Fig. 3 shows the outline of the ADS whose top is the Offence function. Two functions, *Capture flag* and *Take flag home*, depicted below the function are the means of achieving the top function. To capture the flag, the robot needs to reach the flag (*Reach flag*). At the same time, the robot should be in active mode (*Stay active*); and avoid opponents: *Avoid opponents* is one of the necessary functions to achieve the goal.

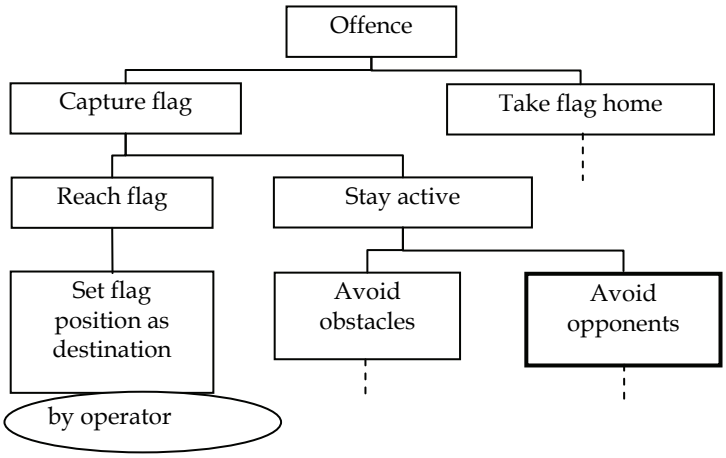


Figure 3. A hierarchical functional model for the function *Offence* identified using the abstraction-decomposition space

Fig. 4 depicts the ADS below the function *Avoid opponents*. One of the means of achieving the function *Avoid opponents* is *Set way-point such as not to encounter opponents*. To select an appropriate course to reach the flag, the situation along the course, especially the positions of opponents, should be understood by the decision-maker. The proposed indication was applied to the function *State comprehension near courses*, which is one of the key sub-functions included in the Offence function, and is allocated to the human operator.

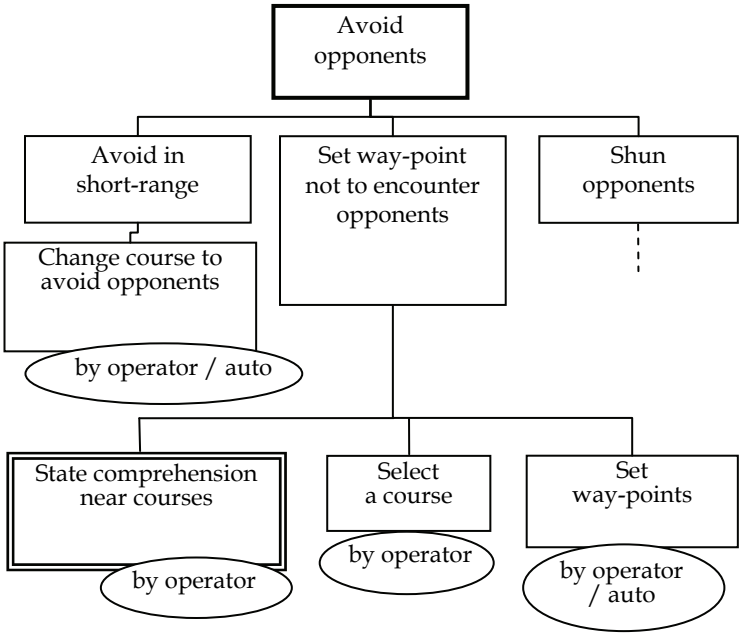


Figure 4. A hierarchical functional model for the function *Avoid opponents* identified using the abstraction-decomposition space

The indication is depicted in Fig. 5. A robot is shown as a black circle and the flag as a white circle. The two straight lines connecting the robot and the flag show the *trajectories* along which the robot is going to move. The two lines on the outside, which connect the detection range and the flag area, show the range in which detection becomes possible when the robot moves along the route. In other words, opponents in this area can tackle their own robots moving along the course. The display clearly indicates the *Field of play* of the target task. One of the operator's options is to send a robot as a scout to the field if there is an area where the situation is unknown.

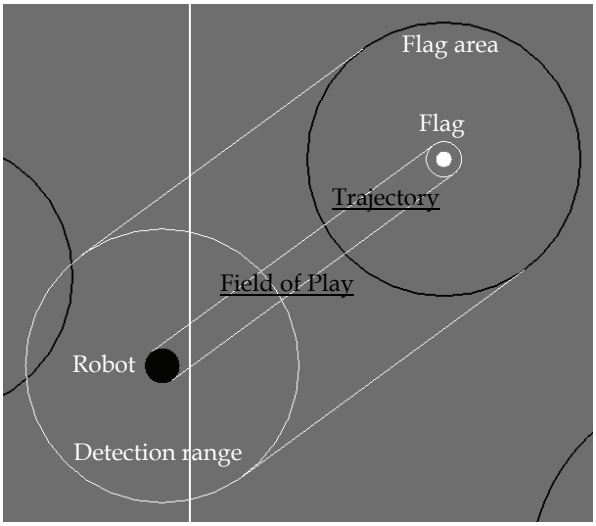


Figure 5. An interface design *Field of play* indicating the state of the function *State comprehension near courses*

3.3.2 A Functional Indication for Defensive Function

The *Intercept opponents* function is an indispensable sub-function for achieving the Defence function. Fig. 6 shows the ADS. *Cooperation between defensive robots* is a type of defensive function realized by a team of robots, and *Block opponents* is a defensive function possessed by individual robots.

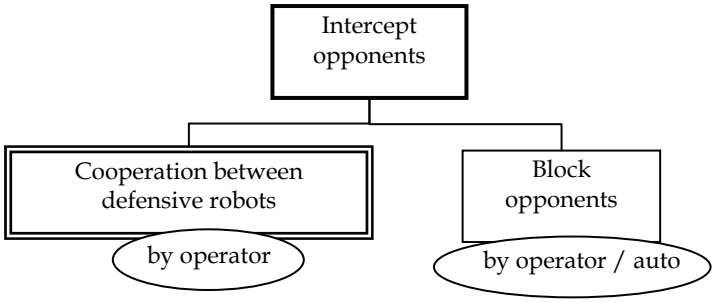


Figure 6. A hierarchical functional model for the function *Intercept opponents* identified using the abstraction-decomposition space

The proposed indication was applied to the function *Cooperation between defensive robots,* which allocated to a human operator. The picture illustrated in Fig. 7 is the functional indication designed for enabling an operator to be clearly aware of the state of the function. A circle around a robot indicates the detection range as described in the previous section. A fan-shaped sector, a *Defensive sector* is where a robot in S/G mode has a high ability to intercept opponent robots coming through. Outside the Defensive sector, the possibility of

catching opponents is lower than within the sector. An operator can use spaces between the sectors as an indication of the *defensive ability of the defensive robot team* in the position.

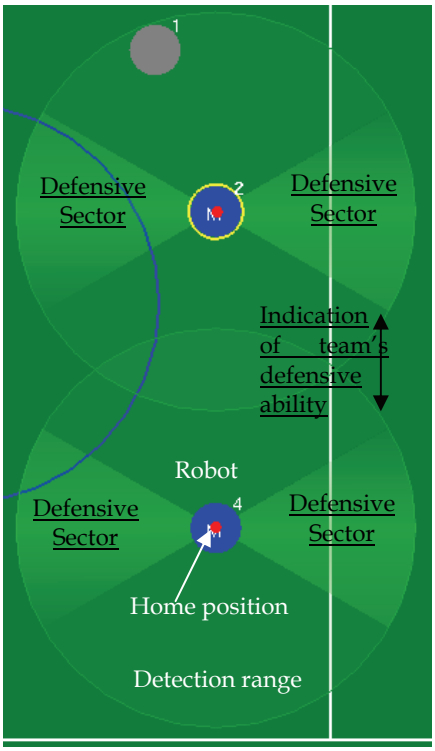
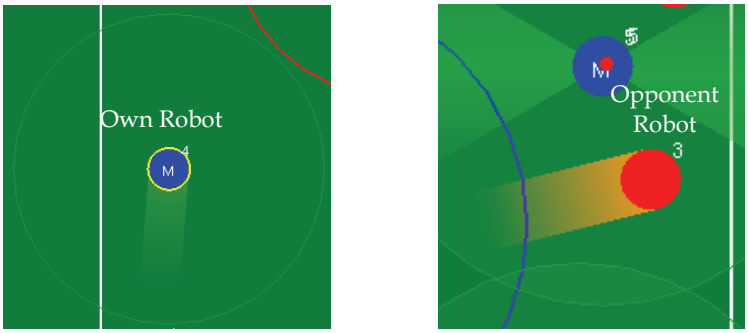


Figure 7. An interface design *Defensive sector* indicating the state of the function *Cooperation between defensive robots*

3.3.3 Functional Indications for Avoidance and Deterrence



(a) Ally movement (b) Opponent movement

Figure 8. An interface design *Ally movement* and *Opponent movement* indicating the state of the function *Shun opponents* and *Block opponents*

The third and fourth indications were designed for two functions of *Shun opponents* and *Block opponents*. The two functions are represented by indications of movements of own and opponent robots. *Ally movement* and *Opponent movement* are bar-like indications drawn on the robot, where they point to the direction of movement of the robot and the length of the bar is set in proportion to speed of the robot. The indications are shown in Fig. 8. The necessity of the two indications was recognized through the Experiment 1. Therefore, the indications were only used in Experiment 2.

4. Experiment 1

This section describes the first experiment conducted to reveal the basic efficacy of EID in human-robot interactions, where the material was first presented at (Furukawa, 2006). After explanation of a procedure of the experiment using the prototype system, we discuss the results to examine the usefulness of ADS for representing whole tasks allocated to humans and robots, the feasibility of designing indications for the functions in the ADS, and the efficacy of function-based interface design to improve human-robot collaboration.

4.1 Procedure

Twenty-two paid participants (undergraduate and graduate students) took part in the experiment. All participants reported that they had normal or fully corrected vision and hearing. The participants were randomly divided into two groups of eleven. One group (the original group; O1 – O11) used the original human-robot interface for the RoboFlag simulation, and the other group (the modified group; M1 – M11) used the modified interface display designed according to the proposed concept.

Offensive and defensive tasks of the rival robots were fully automated by using the two types of automatic controls implemented in this study.

The participants learned their tasks, rules of the game and the details of the assigned HRI, and mastered skills for controlling the team of robots through playing the game several times. They were asked to try it out until they found their own strategies to play the game. After they had decided on their strategies, they played the game five times as part of the main experiment. At the end of each game, they were asked to write the details of their strategies and usage of information represented on the display. The quantified data acquired in the main experiments of five games were then statistically analysed.

4.2 Results

4.2.1 Statistical Data Analysis

The number of flags captured was counted for every game. The averages and standard deviations of participants' captures in the original and modified conditions are M = 0.75, SD = 0.62, and M = 1.20, SD = 0.85. A repeated-measures ANOVA test indicates that the difference between two conditions is significant ($F(1, 20) = 6.164$, $p = 0.022$**). This result may suggest that the modified display is effective in supporting operators in their offensive task, regardless of their ability or the strategy used for the task.

Averages and standard deviations of win percentages under the original and modified conditions are M = 45.4%, SD = 34.7%, and M = 63.6%, SD = 26.6%. However, a t-test shows that the difference between two conditions is not significant ($t = 1.379$, $df = 18.712$, $p = 0.184$).

In addition, the results of the statistical analysis show that there are no significant differences between the original and modified conditions for the number of flags captured by opponent robots, the numbers of times that participants' and opponents' robots were tagged, total elapsed times, and time before the first capture by participants' and opponents' robots. However, at least, the results show no sign of any ill effects caused by using the modified interface.

4.2.2 Strategies Developed and Use of Functional Indications

This section illustrates the strategies developed by the eleven participants of the modified group and how they used the information on functions represented on the display. The participants played RoboFlag several times using the original display after they had completed the main experiments with the modified version. In an interview immediately afterwards, they were asked to explain the strategies they used during the main experiments, their usages of the indications of functions during the experiments, and the importance of the information in completing their missions. The typical strategies for offensive and defensive tasks are described in Tables 1 and 2 respectively with the usages of the functional indications and the participants who used the strategies.

For offensive operations, five participants mainly used the R&B automatic operation to capture the flag. Four of them tried to comprehend the state of the robots and situation around the course using the Field of play indication. For defensive operations, ten participants allocated two to four robots on a course that opponent robots followed to capture the flag. Eight of ten used the Defensive sector indication to decide appropriate spaces between the guarding robots at the training phase and/or the main experiments. Their usage and target functions exactly match with those expected in designing phase.

Strategies	Use of functional indication (Field of play)	
	Used	Did not use
used R&B	M1, M2, M7, M10	M11
manual operation	none	M3, M4, M5, M6, M8, M9

Table 1. Participants categorized by their strategies for offensive tasks and usage of the *Field of play*

Strategies	Use of functional indication (Defensive sector)	
	Used	Did not use
no operations	none	M3
defensive operations	M1, M2, M4, M5, M7, M9, M10, M11	M6, M8

Table 2. Participants, categorized by the strategies for defensive tasks and usage of the *Defensive sector*

The participants who chose manual controls for offensive actions fixed all the waypoints and timings of the orders in advance. The indication was not necessary for them during the main experiments. In spite of this, they mentioned that the indication was useful for developing their own strategies during the trial-and-error processes in the training.

One participant who used the manual-controlled strategy for offence decided not to take any defensive action. A swift attack was his only strategy. The Defensive sector display is not necessary for this strategy.

4.3 Discussion

The analysis on the operators' uses of the functional indications suggests that definition of functions specified in the ADS meets the participants' understanding of functions, and that the ADS includes all the functions to which participants directed their attention in the operations. It also demonstrates that the functional indications, which are designed for the functions, were useful for participants to comprehend states of the functions. Because only a few strategies were used by the participants, further experiments should be conducted under a condition that the wide variety of strategies was used for operations.

The results also indicate that the need for a functional display closely depends on the strategies actually used during operations. This result suggests that individual difference in strategies should be taken into account when designing suitable interface displays for supervising multiple robots.

As for this experiment, the functional indications added to the original display did not cause obvious harm to the participants even when the information was not necessary in their operations. It can be said that the ADS and the interface display based on the ADS were appropriately built, which do not cause any interference in participants' supervision.

These findings may lead to the conclusion that the proposed design concept can offer a proper framework for developing HRIs which provide effective human supervision of multiple robots.

5. Experiment 2

This section describes the second experiment conducted to discuss three research questions about the basic efficacy of the human-robot interface design concept under the condition that the wide variety of strategies was used for operations (Furukawa, 2008). The first question is "Do the indications provide adequate information about status of functions that operators want to comprehend?" The second is "Are the indications useful for state comprehension of the functions?" and the third "Are the indications effective in appropriate comprehension of states of the functions?" The conditions of the test-bed simulation had been changed to prompt the participants to develop various strategies. The results provided positive evidences for the first and second questions, and valuable suggestions for the next stage of this project.

5.1 Procedure

New settings were applied for the robots in this study. The number of defensive robots was changed from three to four for improvement of the defensive ability. The offensive robots had three different courses to enter the player's territory, and three types of the timings. A pair of settings was randomly selected from the alternatives. Because of the randomness, the participants had to operate their robots adaptively.

Twenty-one paid participants took part in the experiment. The participants were randomly divided into three groups of seven. One group (MO group) used the modified human-robot interface for the RoboFlag simulation at the first stage and the original HRI at the second

stage of the experiment. The second group (OM group) used the original interface at the first stage and the modified HRI at the second stage. The third group (OO group) used the original HRI at the first and second stages.
At the first stage, the participants learned their tasks, rules of the game, and the details of the assigned HRI. They also mastered skills for controlling the team of robots through playing the game. They were asked to try it out until they found their own strategies to play the game. The time limit of the training phase was set eighty minutes. After they had decided on their strategies, they played the game ten times as part of the main experiment. At the second stage, they tried to master the assigned HRI. The time limit was fifteen minutes. At the end of each game, they were asked to write the details of their strategies and usage of information represented on the display. The quantified data acquired in the main experiments were then statistically analyzed.

5.2 Results

5.2.1 Strategies Developed and Use of Functional Indications

This section illustrates the strategies developed by the participants of the MO and OM groups and how they used the information on functions represented on the modified display. The both groups of participants had experiences in playing RoboFlag with the modified display. In an interview immediately after the main experiments, they were asked to explain the strategies they used during the main experiments, their usages of the indications of functions during the experiments, and the importance of the information in completing their missions. In this study, we considered that a functional indication was used by a participant only when the participant mentioned actual usefulness or necessity of the indication.
1) Disposition of Defensive Robots: Table 3 described the typical strategies used for disposition of defensive robots and the usages of the Defensive area. The first number in a cell is the total number of participants who used the strategy or the indication. The two numbers in parentheses are the subtotal numbers for MO and OM groups, respectively.
All fourteen participants (MO and OM) assigned one or few robots defensive tasks. Three of them tried to set defensive robots adaptively during plays, and all used the Defensive area for the disposition. The other eleven had decided positions to set them up in their training phases. Understandably, the numbers of users are low.

Strategies	No. of participants		Percentage (b/a)
	Used the strategy (a)	Used Defensive area (b)	
fixed	11 (6, 5)	1 (1, 0)	9% (17%, 0%)
adaptively set	3 (1, 2)	3 (1, 2)	100% (100%, 100%)

Table 3. The typical strategies used for disposition of defensive robots with usages of the designed functional indication

2) Defensive Actions: The strategies used for defensive actions and the usages of the indications are shown in Table 4. Whether they selected the S/G automated control or manual control, all participants used one or two of the functional indications, which are Defensive area, Opponent movement, and Ally movement.

Strategies	No. of participants			
	Used the strategy	Used Defensive area	Used Opponent movement	Used Ally movement
automated (S/G)	13 (7, 6)	5 (4, 1)	9 (4, 5)	2 (0, 2)
manual	1 (0, 1)	0 (0, 0)	1 (0, 1)	0 (0, 0)

Table 4. The typical strategies used for defensive actions with usages of the designed functional indication

3) Selection of Offensive Routes: As shown in Table 5, four typical strategies were developed by participants for selecting routes to reach the flag and to go back home. During the training phase, some participants found a route in which own robots could travel without going inside of opponent's defensive areas. They used the fixed route in every trial. This strategy is called *fixed detour* in this study. *Adaptive routing* is a way that participants tried to find an opening in opponents' defence line during play and send robots through it. Because the setting depended on situation, every route might be different every time. In *feint strategy*, participants move own robots near opponent robots to let them follow the own robots. As a result, participants can use the opening of opponent's defensive line as a safe route. The fourth strategy is *swift attack*, where participants send robots into opponent's flag area as soon as the game stated. This strategy can be used only for routing from own home to opponent's flag.

Strategies	No. of participants		
	Used the strategy	Used Trajectory	Used Field of play
Task: route to reach the flag			
fixed detour	2 (1, 1)	0 (0, 0)	0 (0, 0)
adaptive routing	4 (4, 0)	2 (2, 0)	0 (0, 0)
feint	6 (2, 4)	2 (1, 1)	1 (1, 0)
swift	2 (0, 2)	0 (0, 0)	0 (0, 0)
Task: route to back home			
fixed detour	2 (1, 1)	0 (0, 0)	0 (0, 0)
adaptive routing	4 (2, 2)	2 (1, 1)	1 (0, 1)
feint	8 (4, 4)	2 (2, 0)	1 (1, 0)
swift	0 (0, 0)	0 (0, 0)	0 (0, 0)

Table 5. The typical strategies used for selection of offensive routes with usages of the designed functional indication

None of the participants who selected the fixed detour or swift strategies used either of two functional indications relative to routing, which are Trajectory and Field of play. It is understandable because state comprehension was not necessary in the operation with the strategies. On the other hand, state comprehension of the offensive robots was important

when they were using the other two strategies. Some of them, who used the functional indications, pointed out the usefulness to comprehend the states. The other who did not use the indications explained that they were trained well enough to comprehend situations without the information.

4) Offensive Actions: All participants selected manual control for offensive actions (Table 6). Only two participants in MO group used Opponent movement, and other five did not. On the other hand, six participants in OM group made use of the functional indication. The causes of the difference in the number between the two groups were not confirmed through the interviews.

Strategy	No. of participants	
	Used Opponent movement	Used Ally movement
manual	8 (2, 6)	5 (3, 2)

Table 6. The typical strategies used for offensive actions with usages of the designed functional indication

5.2.2 Subjective Evaluation of Adequacy of the Functional Information

In the interview, the participants were asked to answer additional information which was not displayed in the modified display but they wished to use in the operations. All participants answered that they did not feel inadequacy of functional information. And three pointed out that some information about own robot's intent might be useful to predict the robot's move. For example, indication that shows a robot is going to change the direction because there is an opponent robot right before it.

5.2.3 Quantitative Analysis of Effects on Performance

Several performance parameters were measured for every game, which are the number of flags captured by participants' and opponents' robots, the numbers of times that participants' and opponents' robots were tagged, total elapsed times, and time before the first capture by participants' and opponents' robots. In addition, win percentages were calculated for every condition.

Statistical analysis on the parameters could not show any significant differences between each pair of two conditions described below:

Analysis I: The first main experiment with the modified display (MO group), and the first main experiment with the original display (OM group).

Analysis II: The second main experiment with the modified display (OM group), and the second main experiment with the original display (OO group), where the original display was used for the first stage equally.

Analysis III: The second main experiment with the original display where the modified display was used for the first stage (MO group), and the second main experiment with the original display where the original display was used for the first stage (OO group).

However, at least, the results show no sign of any ill effects caused by using the modified interface.

5.3 Discussion

The variety of strategies was larger than the previous experiment where only a few strategies were developed. From this point of view, the experimental condition designed in this study was appropriate for the study on the functional indications based on the proposed design concept. Although the number of participants is limited, the experiment offered sufficient data for this initial stage of the project.

5.3.1 Adequacy of the Functional Information

The results in Section 5.2.2 indicate that the participants did not feel any need for indications for functions which were not considered during the design phase. In other words, the designed functional indications covered all the functions of which the participants needed informational assistance. This suggests that the proposed design method has basic efficacy to design appropriate HRI that provides sufficient information to operators for conducting their work.

As the participants pointed out, we have claimed the importance of providing information about intention of automated systems (Furukawa et al, 2004). In next stage of this study, we are planning to apply the idea to designs of HRI.

5.3.2 Usefulness of Designed Functional Indications

The use of functional indications can be recognized as results of participants' subjective evaluation on usefulness of the indications for their operations. Through trials in the training phase, he or she selected proper functional indications to use for conducting tasks with his or her own strategies. The reason for selection was that the participant recognized the indications were useful and worth to use. The results show that every functional indication was actually used in selected conditions. This may suggest that the designed indications in this study were useful to comprehend situations of the functions, and that the proposed design concept is practical to design HRIs adaptable to operations with variety of strategies.

5.3.3 Effectiveness of the Designed Functional Indications

The results of the statistical analysis indicate that the measured performances are distributed widely throughout the participants, suggesting that more factors should be considered in data analysis on their performances.

For some strategies, state comprehension is not a necessary task in conducting operations. It means that the necessity of a functional indication for a participant depends on a strategy developed by the participant. Furthermore, the necessity of a functional indication depends on participant's ability to comprehend situation. The necessity is low for who has high ability to understand dynamic states of own and opponent robots. The numbers of data in this study were not sufficient to conduct statistical analysis using additional factors. It is expected that additional experiments can quantitatively reveal the efficacy.

6. Conclusion

This chapter describes two experiments conducted to reveal basic ability of a human-robot interface design concept, in which the ecological interface display concept is used as the

basic framework for implementing the information about a human-robot team work into an interface display.
The results from the first experiment may suggest that the whole work can be modelled using ADS, and it is feasible to design useful functional indications based on the ADS. The results also show the need to consider two factors to design effective HRI displays: the one is participants' strategies developed for tasks, and the other is how they use the functional indications.
In the second experiment, the adequacy, usefulness and effective of the functional indications were evaluated under the condition that the wide variety of strategies was used for offensive and defensive operations. The results demonstrate that the designed interface display has basic efficacy to provide adequate and useful functional information for supporting human operators for supervision of a robot team. It is expected that additional experiments with a large number of participants can quantitatively reveal the ability where strategies developed and use of functional indications are taken into account.
This empirical study provides empirical evidence for the efficacy of the proposed approach to enable effective human supervision of multiple robots.
To elaborate the practical and effective design concept for HRIs, several techniques must be necessary to develop. Typical examples are a method for designing functional models for target tasks using an ADS as a framework, a method for selecting functions for which support of comprehension is necessary for operators, and a method for designing effective indications for easy understanding of states of the functions.

7. Acknowledgement

This work has been partially supported by Grants-Aid for Science Research 18500104 of the Japanese Ministry of Education, Science, Sports and Culture.

8. References

Burns, C. & Hajdukiewicz, J. (2004). Ecological Interface Design, Brunner-Routledge, 0-415-28374-4 , Florida

Campbell, M.; D'Andrea, R.; Schneider, D.; Chaudhry, A.; Waydo, S.; Sullivan, J.; Veverka, J. & Klochko, A. (2003). RoboFlag games using systems based hierarchical control, *Proceedings of the American Control Conference*, pp. 661–666, 0780378962, Denver, 2003, IEEE, Piscataway, N. J.

Chaimowicz, L.; Cowley, A.; Gomez-Ibanez, D.; Grocholsky, B.; Hsieh, M.; Hsu, H.; Keller, J.; Kumar, V.; Swaminathan, R. & Taylor, C. (2005). Deploying air-ground multi-robot team in urban environments, In: *Multi-Robot Systems. From Swarms to Intelligent Automata Vol. III*, Parker, L.; Schneider, F. & Schultz, A., (Eds.) , pp. 223–234, Springer, 1-4020-3388-5, Dordrecht, Netherlands

Furukawa, H. (2006). Toward ecological interface design for human super-vision of a robot team, *Proceedings of 3rd International Conference on Autonomous Robots and Agents - ICARA'2006*, pp. 569–574, 0-473-11566-2, Palmerston North, New Zealand, 2006, Massey University, Palmerston North, New Zealand

Furukawa, H. (2008). Functional display for human supervision of a multiple robot system: Adequacy for operations with a variety of strategies, *Proceedings of 2008 IEEE International Conference on Distributed Human-Machine Ssytems*, pp.39-44, Athens, Greece, March

Furukawa, H.; Nakatani, H. & Inagaki, T. (2004). Intention-represented ecological interface design for supporting collaboration with automation: situation awareness and control in inexperienced scenarios, P*roceedings of Human Performance, Situation Awareness and Automation II*, pp. 49–55, 0-8058-5341-3, Daytona Beach, Month, Lawrence Erlbaum Associates, Mahwah, NJ

Goodrich, M.; Quigley, M. & Cosenzo, K. (2005). Task switching and multi-robot teams, In: *Multi-Robot Systems: From Swarms to Intelligent Automata Vol. III*, Parker, L.; Schneider, F. & Schultz, A., (Eds.), pp. 185–195, Springer, 1-4020-3388-5, Netherlands

Jin, J. & Rothrock, L. (2005). A visualization framework for bounding physical activities: toward a quantification of gibsonian-based fields, *Proceedings of the Human Factors and Ergonomics Society 49th Annual Meeting*, pp. 397–401,, Orlando, 2005

Lee, J.; Thomas, G. & Pollack, E. (2000). Ecological interface design (EID) and the management of large numbers of intelligent agents, In: *Human Error and System Design and Management*, Elzer, P.; Kluwe, R. & Boussoffara, B. (Eds.), pp. 137-151, Springer-Verlag, 1-85233-234-4, London

Miller, C. & Vicente, K. (2001). Comparison of display requirements generated via hierarchical task and abstraction-decomposition space analysis techniques, *International Journal of Cognitive Ergonomics*, Vol. 5, No. 3, pp. 335–355, 1088-6362

Nielsen, C.; Goodrich, M. & Ricks, R. (2007). Ecological interfaces for improving mobile robot teleoperation, *IEEE Transactions on Robotics*, Vol. 23, No. 5, Oct., pp. 927-941, 1552-3098

Parasuraman, R.; Galster, S.; Squire, P.; Furukawa, H. & Miller, C. (2005). A flexible delegation- type interface enhances system performance in human supervision of multiple robots: empirical studies with RoboFlag, *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, Vol. 35, No. 4, Jul., pp. 481–493, 1083-4427

Rasmussen, J. (1986). *Information Processing and Human-machine Interaction*, Elsevier Science Publishing, 0444009876, New York

Rasmussen, J. & Pejtersen, A. (1995). Virtual ecology of work, In: *Global Perspectives on the Ecology of Human-Machine Systems*, Flach, J.; Hancock, P.; Caird, J. & Vicente, K., (Eds.), pp. 121-156, Lawrence Erlbaum Associates, 0805813810, UK

Sawaragi, T.; Shiose, T. & Akashi, G. (2000). Foundations for designing an ecological interface for mobile robot teleoperation, *Robotics and Autonomous Systems*, Vol. 31, No. 3, May, pp. 193–207, 0921-8890

Sheridan, T. (2000) HCI in supervisory control: twelve dilemmas, In: *Human Error and System Design and Management*, Elzer, P.; Kluwe, R. & Boussoffara, B. (Eds.), pp. 1-12, Springer-Verlag, 1-85233-234-4, London

Stoeter, S.; Rybski, P.; Stubbs, K.; McMillen, C.; Gini, M.; Hougen, D. & Papanikolopoulos, N. (2002). A robot team for surveillance tasks: design and architecture, *Robotics and Autonomous Systems*, Vol. 40, No. 2-3, pp. 173–183, 0921-8890

Vicente, K. (1999). *Cognitive Work Analysis: Toward Safe, Productive, and Healthy Computer-Based Work*, Lawrence Erlbaum Associates, 0805823964, NJ

Vicente, K. (2002). Ecological interface design: progress and challenges, *Human Factors*, Vol. 44, No. 1, pp. 62–78, 0018-7208

Vicente, K. & Rasmussen, J. (1992). Ecological interface design: theoretical foundations, *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 22, No. 4, Jul/Aug., pp. 589–606, 0018-9472

Wirz, R.; Marin, R. & Sanz, P. (2006). Remote programming over multiple heterogenous robots: a case study on distributed multirobot architecture, *Industrial Robot – An International Journal*, Vol. 33, No. 6, pp. 431–442, 0143-991X

Xu, W. (2007). Identifying problems and generating recommendations for enhancing complex systems : Applying the abstraction hierarchy framework as an analyticaltool, *Human Factors*, Vol. 49, No. 6, Dec., pp. 975-994, 0018-7208

3

Force Field Simulation Based Laser Scan Alignment

Rolf Lakaemper and Nagesh Adluru
Temple University Philadelphia
USA

1. Introduction

Alignment of sensor data, typically acquired from cameras, laser range scanners, or sonar sensors, is the basis for all robot mapping tasks. Recent advances in the development of laser range devices make research on laser range alignment a focus of robot mapping research. In contrast to cameras, laser range scanners offer relatively precise depth information, yet the feature density is relatively sparse. Since alignment algorithms are based on feature correspondence, a lack of features naturally causes problems. One way to approach that problem is to cover the area with a high number of scans, such that subsequent scans have a low relative displacement only. This guarantees sufficient scan overlap and a reliable detection of feature correspondences. Though this approach is feasible for many mapping applications, it can not be assumed for an important field of robotics, namely Urban Search and Rescue Robotics ("rescue robots"), in there especially the setting of multi robot mapping. In multi robot mapping, a number of robots scan the environment independently, without reliable knowledge of their relative position. Additional sensors, like GPS, can not be assumed due to the nature of the environment. Non autonomous rescue robots were e.g. deployed after the 9/11 attack to assist in the search for victims in the collapsed towers. In such an environment, GPS is not available because of the massive concrete walls surrounding the robots. The task of multi robot mapping in rescue environments imposes especially challenging constraints:

- no precise or reliable odometry can be assumed, which means especially that the robots' relative poses are unknown
- due to the nature of catastrophe scenarios no distinct landmarks are given
- the overlap between pairs of the robots' scans is minimal

Figure 1 shows 12 out of 60 single scans from multiple robots, taken in a disaster test area at NIST, Gaithers-burg, MD. Even for humans it is hard to detect overlapping features.
Our approach to alignment of such a data set is to first give a rough estimate of the robots' poses, called the pre-alignment, and then to improve the achieved map. This article deals with the second step, the improvement, see figure 2.
This article introduces of a new process, called 'Force Field Simulation' (FFS), which is tailored to align maps under the aforementioned constraints. It is motivated by simulation of dynamics of rigid bodies in gravitational fields, but replaces laws of physics with constraints derived from human perception. It is an approach of the family of gradient

descent algorithms, applied to find an optimal transformation of local maps (in particular, laser range scans) to build a global map based on feature correspondences between the local maps. Figure 3 shows the basic principle: forces (red arrows) are computed between 4 single scans (the 4 corners). The scans are iteratively transformed by translation and rotation until a stable configuration is gained. The single scans are not merged, but kept separated. As they are moved according to the laws of the motion of rigid bodies in a force field, single scans are not deformed. FFS has the following properties:

1. Low level correspondences (data point correspondences) are not made by a hard decision (an integral of forces between pairs of points defines the force field in place of hard 'nearest neighbor'correspondences)
2. FFS is a gradient approach, it does not commit to an optimal solution in each iteration step
3. The iteration step towards an optimal solution is steered by a 'cooling process', that allows to jump the system out of local minima
4. FFS transforms all scans simultaneously
5. FFS easily incorporates structural similarity modeling human perception to emphasize/strengthen the correspondences

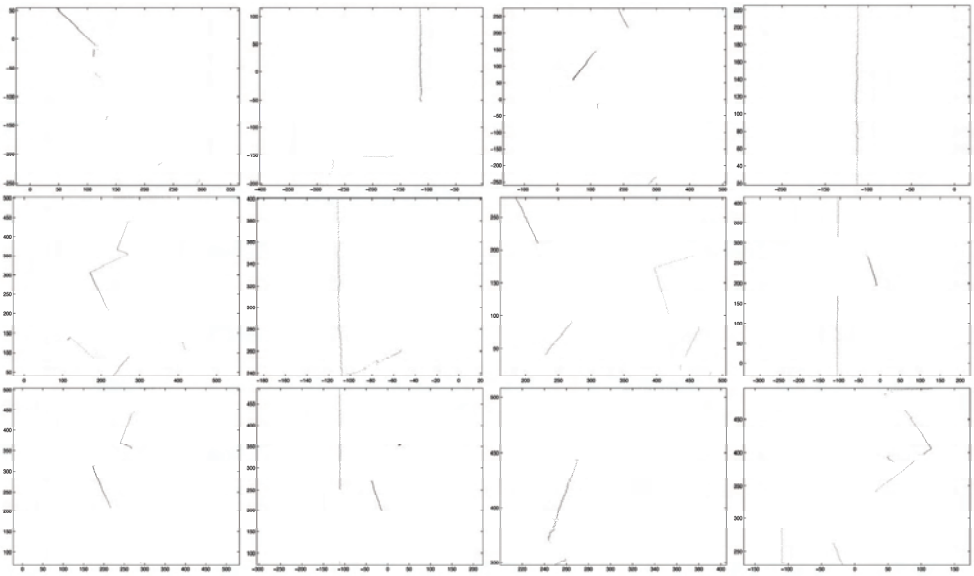


Figure 1. 12 out of 60 single scans from the NIST disaster test area. Even for humans it is hard to detect overlapping features

2. Related work

The problem of aligning *n* scans has been treated as estimating sets of poses (Lu and Milios, 1997). Since sets of poses and the associated structures (maps) are conditionally independent, this estimation is Simultaneous Localization and Mapping. The conditional independence is e.g. the key for Rao-Blackwellization (factoring the posterior of maps) of particle filters for SLAM (Montemerlo et al., 2002).

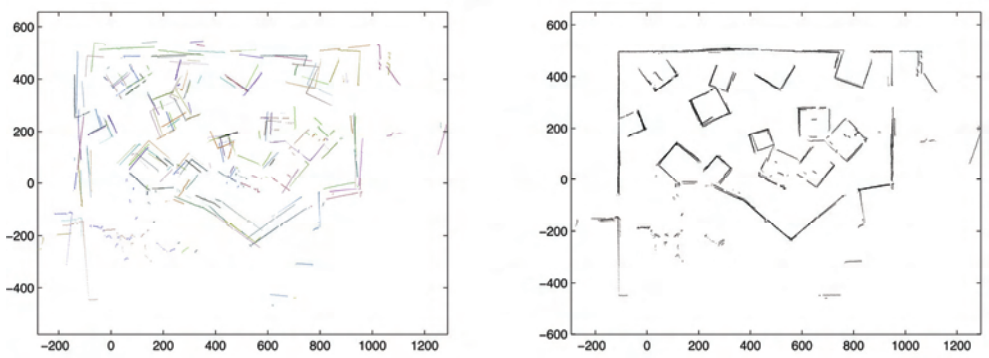


Figure 2. Left: Pre-alignment of the 60 scans of the NIST disaster test area. Right: improved alignment. Force Field Simulation is used to achieve the improved alignment, given an estimated pre-alignment

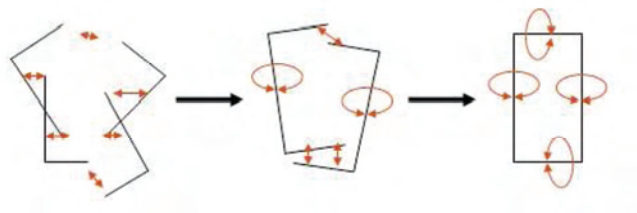


Figure 3. Basic principle of FFS. Forces are computed between 4 single scans. Red arrows illustrate the principle of forces. The scans are iteratively (here: 2 iterations) transformed by translation and rotation until a stable configuration is achieved

There have been several algorithms to estimate the sets (Olson et al., 2006; Frese, 2006; Frese et al., 2005; Thrun et al., 2002; Minguez et al., 2006; Konolige, 2003). The underlying framework for all such techniques is to optimize a constraint-graph, in which nodes are features, poses and edges are constraints built using various observations and measurements like odometry scan-matching of range scans. These techniques differ in

- how they represent graphs - e.g. (Frese, 2006) uses a sophisticated data structure called Tree-map,
  (Thrun et al., 2002) represents using sparse extended information filters (SEIF).
- how they build constraints - e.g. (Lu and Milios, 1997) uses linearized constraints obtained from scan-matching and odometry, (Olson et al., 2006) works with non-linear constraints.
- how they optimize the graphs - e.g. (Olson et al., 2006) uses stochastic gradient descent for approximate optima, borrowing the ideas from learning theory, (Lu and Milios, 1997) solves for exact optima using brute-force, (Frese et al., 2005) use Gauss-Seidel relaxation again for approximate optima.

All these approaches have performed well in many practical cases but they have one drawback that is they are sensitive to behavior of error models of sensors because of several assumptions and approximations which might not hold with sparse sensing.

(Lu and Milios, 1997) linearizes constraints by linearizing pose-relations, solving a linear equation of the form $AX = B$ to estimate X, the set of poses. This requires that A is invertible, so they conjecture that A is invertible if the constraint-graph is fully connected and the errors of the observations behave in a gaussian/normal way. (Borrmann et al., 2007) extends the same technique for 3D scans.
(Olson et al., 2006) presents an approximate optimization of non-linear constraints and demonstrate that their approach of approximating the optimization process in non-linear state space yields superior results compared to finding exact optima by approximating a non-linear state space (SLAM) to a linear state space.
Another strategy of attacking the problem is to treat the problem of SLAM from a perspective of aligning n scans simultaneously. The algorithms exploiting this perspective build from image registration techniques, the most famous being Iterative Closest Point (TCP) (Besl and McKay, 1992; Chen and Medioni, 1992) and it's numerous variants to improve speed and converge basins (Rusinkiewicz and Levoy, 2001) and (Lu and Milios, 1994; Birk and Carpin, 2006). Basically all these techniques do search in transformation space trying to find the set of pair-wise transformations of scans by optimizing some function defined on transformation space. The techniques vary in defining the optimization functions that range from being error metrics like "sum of least square distances" to quality metrics like "image distance" as in (Birk, 1996). Their optimization process itself can be gradient descent or hill climbing or using genetic programming strategy as in (Robertson and Fisher, 2002). All of these techniques have one major limitation, which is they search in *pair-wise* transformation space. Though in some variants of ICP the error from all pair-wise transformations is spread across all transformations to simultaneously align all scans, the procedure can be highly sensitive to outliers (Rusinkiewicz et al., 2005).
FFS also adapted the perspective of aligning n scans, it treats the alignment problem as an optimization problem. Rather than using a least squares solution to compute intermediate motions, FFS uses an iterative gradient technique to solve for (local) optima. Here FFS is similar to the approach proposed by (Eggert et al., 1998), which simulates a dynamic spring system to register multiple range scans simulatenously. They describe the advantages of such a gradient descent system as follows: '*The reason [not to use a least square solution] is that the effects of any significantly incorrect correspondences are compounded when the best alignment is computed (...) With a dynamic system it is possible to move in the direction of an intermediate solution without being totally committed to it*'. (Eggert et al., 1998) differ in the choice of the registration function, which in contrast to FFS is based on one to one correspondences between points, as well as in the optimization technique.
FFS uses a gradient method with decreasing step width Δ_t. The registration function (target function) of FFS is based on Gaussian fields, similar to (Boughorbel et al., 2004). In contrast to (Boughorbel et al., 2004), FFS uses a variable, decreasing σ for each iteration step t. Additionally, (Boughorbel et al., 2004) solve the optimum of the registration using a quasi-Newton method, hence they do not steer the system with a step width parameter.
Since we keep the single scans separated, our search space is high dimensional, in the *2D* case it is $3n$-dimensional *(3D: $6n$*-dimensional), with n being the number of scans. For example, our experiment described in section 4.2 uses 60 scans, our search space is therefore 180-dimensional. Birk and Carpin use a random walk technique to reach the optimal solution. Since random walk techniques tend to become critical in high dimensions, we do not utilize this technique in our approach but decide in favor of a guided (gradient) walk.

This search in high dimensional space at first sight seems very complicated, demanding computation of high dimensional gradient but fortunately using potential field simulation for various computer vision tasks like contour detection, segmentation, registration has been empirically successful (Yang et al., 2006; Jalba et al., 2004; Xu and Prince, 1998), (Ayyagari et al., 2005), (N.Paragios et al., 2003), (Veltkamp and Hagedoorn, 1999). Since mapping is closely related to registration, the approaches whose motivations are closely related to our approach are (Biber and Strasser, 2006; Ayyagari et al., 2005; Eggert et al., 1998). In (Eggert et al., 1998) they align range scans by moving them simultaneously. The movements are not just based on the minimizing error of transformation computed using correspondences but on the simulated fields generated by imaginary springs attached to the corresponding points. Our technique differs from (Eggert et al., 1998) in that the force field is generated not just by closest point correspondences but using perceptual principles and gaussian fields similar to (Boughorbel et al., 2004). (Biber and Strasser, 2006) also performs search in 3n dimensional space. For each configuration they compute energy as the sum of the Normal Distribution Transforms (NDT) (Biber and Strasser, 2003) of all the scans in the configuration and update the configuration using Newton's optimization algorithm that involves the first and second derivatives of the energy. Their approach is very closely related to ours but does not use perceptual features and rigid body dynamics and hence in principle can be more sensitive to outliers.

3. Force field based mapping

The following motivates and describes the FFS algorithm. A pseudo code representation can be found at the end of this section.

3.1 Basic Principle

To draw the analogy to Newtonian Physics, each scan s_i can be seen as a rigid body of masses: the scan points represent the masses, rigidly connected by massless rods. A global map $\boldsymbol{g}$ defines the transformation of all scans, it therewith defines the distribution of all masses (the union of all scan points). In the framework of Newtonian Physics the gravitational forces between these masses forms a gradient field. The FFS algorithm is motivated by simulation of the movement of bodies in a gradient field. In contrast to pure physics it replaces physical principles of masses and forces by principles that correspond to human visual perception, i.e. gravitation is replaced by 'strength of correspondence'. Also, to achieve convergence to a stable state of minimal total energy, the kinetic energy is not taken into account, i.e. the velocity of each rigid body after each iteration step is set to 0. Also FFS uses a 'cooling' strategy in its step width parameter that initially adds energy to the system to allow for escape from local minima, see section 3.2.

Let $S=s_1,\ldots,s_n$ be a set of n scans gained from laser range scan devices. A scan $s_i=(p^i_1,\ldots p^i_j)$ consists of j data points. Data points are the coordinates of reflection points of the laser range scanner in a local coordinate frame defined by a single robot pose. We also assign a scalar value, a *mass* m^i,- to each data point, which can be interpreted as the perceptual importance. For the purpose of multi robot mapping we assume that each scan is possibly gained from a different robot, while the robots' relative poses are unknown or poorly estimated. The task of the algorithm is an optimization over the set of the robots' poses; hence the goal is to find transformations for all n scans $s_{i=1..n}$ to registrate the scans, such that

similar features in different scans match 'perceptually consistently' when they are superimposed on top of each other.
Observe that the order of local maps is irrelevant in our framework since we transform the scans simultaneously, which is an important property of the algorithm to be applicable for multi robot mapping. For single robot mapping, FFS is canonically extandable to online FFS, see section 3.5.
The transformations performed are rotation θ_i and translation x_i, y_i of each scan s_i. Superimposing the transformed maps builds a *global map* g as shown in Fig. 4.
During the global map building process single maps are not merged but kept separated. Therefore a global map is defined by the vector of the transformation parameters of all scans: a *global map* $g = (t_1,..., t_n)$ is a $3n$-dimensional vector of transformation parameters $t_i = (x_i, y_i, \theta_i)$ of all n scans $s_{i=1..n}$. The space of all global maps is denoted by $\mathcal{G}$. To registrate the scans,we define *a fitness measure* P_g to evaluate the 'perceptual consistency' of a global map g. Finding the global map g_k that minimizes P_g is clearly an optimization problem. FFS solves this optimization problem with a gradient technique that iteratively transforms all scans simultaneously until a stable configuration (local minimum) is reached. The following section will motivate and define the fitness measure P_g as well as the implementation of the gradient approach.

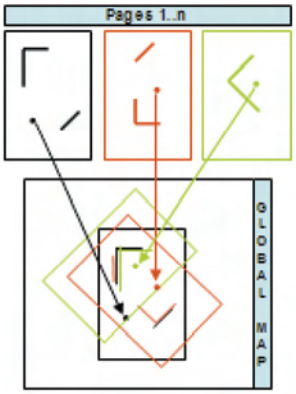


Figure 4. Single scans are transformed to build a global map

3.1.1 Correspondence function

As in (Boughorbel et al., 2004), the basic idea of our registration method is to use a Gaussian field to define a strength of correspondence between data points, i.e. a measure for both spatial proximity and visual similarity of two points belonging to different scans.
A *correspondence* between data point p_1 and a data point p_2 is defined as a vector

$$V(p_1, p_2) = C(p_1, p_2) \frac{p_2 - p_1}{\|p_2 - p_1\|} \tag{1}$$

Its magnitude $\|V(p_1, p_2)\| = C(p_1, p_2)$ describes the strength of correspondence, defined as:

$$C(p_1,p_2)=\frac{1}{\sigma_t\sqrt{2\pi}}e^{\left(-\frac{\|p_2-p_1\|^2}{2\sigma_t^2}\right)}m_1m_2\cos(\angle(p_1,p_2)) \tag{2}$$

with m_i being the mass assigned to p_i, and the angle $\angle(p_1,p_2)$ being the angle between the *directions of points* p_1,p_2, which will be defined in section 3.3. Intuitively, the direction of a point is the direction of an underlying model of a linear structure (a line segment).

A major difference to the pure physics simulation is that the mass values assigned to the data points are not assumed to be constant. The mass m_i for a point p_i is used to compute the force as in eq. 3, yet it can be reassigned a different value for the computation of movement of the scan (we are not modeling physics but perception, hence freedom from Newton's law is given). Steering the mass enables the algorithm to react better to perceptual properties: there is no perceptual reason for an 'important point', e.g. a corner point, assigned a high mass for force computation, to be less mobile than other points during movement computation (caused by its high mass). This observation suggests using different masses during the computation of forces than during the computation of the movement. In the current system, the mass assignment is used to regulate influences of different scan densities, i.e. high density points (due to a high number of scans in a single area) are assigned lower mass values, while low density points (above a certain threshold) are slightly overemphasized using higher mass values.

Expression 2 can be interpreted as a force field whose sources are located in the data points. It has the following properties:

1. The strength of correspondence decays with Euclidean distance, the influence of distance is controlled by the parameter σ_t
2. The strength of correspondence is weighted by the mass of each data point and depends on the angle between point directions, i.e. it is 0 for orthogonal directions, 1 for parallel directions.

We propose this model for the following reasons:

1. Distance likelihood and parallelism follow the basic principles of Gestalt Psychology (Wertheimer, 1958), modeling low level cognition.
2. The scale parameter a_t gives additional freedom to adjust the process (see section 3.2), it enables the correspondence process to work on different visual scales.
3. Assigning a mass to a data point can be seen as assigning a visual importance to it. Data points in higher regions of interest can be stronger emphasized.

In terms of the physical framework, the correspondence $V(p_1,p_2)$ (eq. 1) describes a force on p_1 towards p_2 with strength $C(p_1,p_2)$. Embedding the scans s_i into $\mathbb{R}^2$ using the transformations defined by a global map g, we can define a vector field $F:\mathbb{R}^2\supset\mathcal{P}\rightarrow\mathbb{R}^2$, the *Force Field* on the set of all points $\mathcal{P}=\{p|p\in\bigcup_{i=1..n}s_i\}$ by summing the correspondences.

$$F(p_i)=\sum_{p_j\in\mathcal{P}\setminus p_i}V(p_i,p_j) \tag{3}$$

By definition of the strength of correspondence F is radial and hence a gradient field. With

$$A=m_1m_2\cos(\angle(p_1,p_2))$$

the overlying potential is defined by

$$P(p_i) = \frac{1}{2} \sum_{p_j \in \mathcal{P} \setminus p_i} \int_{\infty}^{r} \frac{A}{\sigma_t \sqrt{2\pi}} e^{\left(-\frac{z^2}{2\sigma_t^2}\right)} dz \tag{4}$$

with $r = \sqrt{(X-x)^2 + (Y-y)^2}, p_i = (X, Y), p_j = (x, y) \in \mathcal{P}$.
Note: $P(p_i)$ is the potential over F since:

$$F(p_i) = -\nabla P$$
$$= \begin{bmatrix} -\frac{\partial \mathcal{P}}{\partial x} \\ -\frac{\partial \mathcal{P}}{\partial y} \end{bmatrix} = \begin{bmatrix} \sum \frac{A}{\sigma_t \sqrt{2\pi}} e^{\frac{-r^2}{2\sigma_t^2}} . \frac{X-x}{r} \\ \sum \frac{A}{\sigma_t \sqrt{2\pi}} e^{\frac{-r^2}{2\sigma_t^2}} . \frac{Y-y}{r} \end{bmatrix} = \sum \frac{A}{\sigma_t \sqrt{2\pi}} e^{\frac{-r^2}{2\sigma_t^2}} . \vec{u} = \sum V(p_i, p_j)$$

where $\vec{u} = \frac{p1-p2}{\|p1-p2\|}$ and all sums are defined over $p_j \in \mathcal{P} \setminus p_i$.
Finally, we define *the fitness measure* or *potential of a global map* $g \in \mathcal{G}$ as the sum of potentials of all data points$p \in (P)$:

$$P(g) = \sum_{p_i \in \mathcal{P}} P(p_i) \tag{5}$$

In this framework, the potential $P(g)$ can be interpreted as the weighted average distance of all point pairs of different scans, the weight being the strength of correspondence. This potential can be seen as the quality of registration achieved by the transformations defined by g. To minimize the potential we apply an iterative gradient descent approach, the gradients in each data point given by $F(p_i)$, eq. 3. Computing the correspondences explicitly gives us these gradients , hence in the implementation of the algorithm there's no need to explicitly compute and derive the potential $P(g)$ (eq. 5) for the actual gradient descent.
In FFS, the computation of the transformation of each scan is determined assuming movement of rigid bodies in the given gradient field, i.e. all data points $p_i^j \subset \mathcal{P}$ of a single scan s_i share the same transformation, consisting of rotation and translation. However, eq. 3 does assume a non-rigid, independent movement of the data points, also the points' potential $P(p_i)$ (eq. 4) is defined over the space of all single (not rigidly connected) point configurations, which is a $2m$ dimensional space with $m = |\mathcal{P}|$. This means, the gradient $F(p_i)$ is defined under the assumption of unrestricted freedom of movement in $\mathbb{R}^{2}$. To implement rigid body movement, we have to impose a restriction on the movement. The restriction is defined by the possible point configurations that are allowed by the transformations $g \in \mathcal{G}$. The laws of rigid body dynamics define these constraints: computing the gradient $F(p_i)$ in each data point as in equation 3 results in a $2m$-dimensional gradient vector, the laws of rigid body dynamics map this vector to a $3n$ dimensional vector ∇g- ∇g describes the transformation of all n scans such that each data point moves in the direction of maximum descent of $P(g)$ in eq. 5 , i.e. under the rigid body constraints. We therefore achieve a movement in the direction of the gradient gained in the $2m$ dimensional single point space projected onto the restricted $3n$ dimensional rigid movement subspace.

The basic laws of dynamics of rigid bodies in force fields accumulate the translation of all masses of a single scan into a single translation and a single rotation around a defined center. For each scan s_{ir} the translational and rotational acceleration has to be determined. The translational acceleration $\alpha_T(s_i)$ of a scan s_i is defined by:

$$a_T(s_i) = \frac{\sum_{p \in s_i} F(p)}{\sum_{p \in s_i} m_p} \tag{6}$$

The rotational acceleration α_R is computed by torque and moment of inertia. Torque and inertia play the role of force and mass respectively, but take into account the distance to the rotational center c_R.

$$inertia = \sum_{p \in s_i} m_i \|p_i - c_R\|^2$$

$$torque = \sum_{p \in s_i} \|p_i - c_R\| \times F(p)$$

α_R is defined as

$$a_R = \frac{torque}{inertia} \tag{7}$$

The rotational center c_R is either defined as the robot's position, or by the center of mass. Experiments show, that in the first iteration steps it is useful to set the rotational center to the center of mass, while in later steps the robot's position is preferable. The first choice enables easier rotation, the second is modeling the actual scan setting more precisely. Hence, the closer the global map is to the solution, the more preferable is the robot's position as rotational center.

With α_T and α_R the transformation $t_k = (x_k, y_k, \theta_k)$ for scan s_k is defined by:

$$(x_k, y_k) = \frac{1}{2} a_T \Delta_t^2 \tag{8}$$

$$\theta_k = \frac{1}{2} a_R \Delta_t^2 \tag{9}$$

Δ_t being the step width of the gradient descent, as described in section 3.2.

With these constraints, the gradient in each iteration is computed by the following steps:

1. for each pair of points $p_i, p_j \in \mathcal{P}$ compute $V(p_i, p_j)$
2. for each point $p_i \in \mathcal{P}$ compute $F(p_i)$
3. for each scan $s_i \in S$ compute the transformation $t_k = (x_k, y_k, \theta_k)$ using the points $p^k_i \in s_k$. This step results in a $3n$ dimensional gradient vector ∇g.

Computing all correspondences V in step 1 is an $O(n^2)$ process, section 3.4.1 will deal with the necessary reduction of computational complexity.

Figure 5 shows 2 iteration steps of FFS using 2 simple scans, consisting of 3 and 5 data points. In the left figure, the forces $F(p_i)$ in each data point p_i are plotted as green dotted lines.The 2 scans are transformed until they are superimposed, i.e. a stable configuration (local minimum of $P(g)$) is reached.

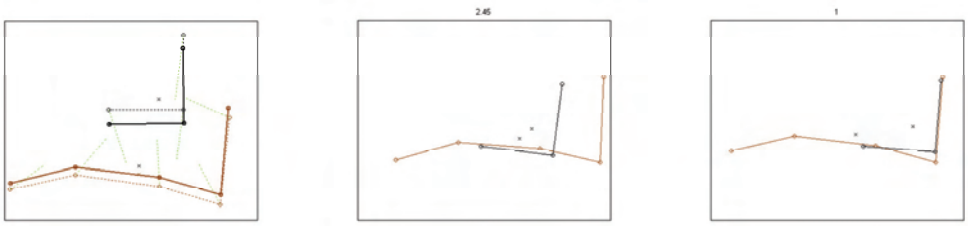


Figure 5. Left: Two scans (black/brown) superimposed. Dotted lines: scans at time *t,* green lines: forces on the data points at time *t.* Solid lines: scans after one iteration, time $t + \Delta t$. Both scans are translated/rotated according to the forces. Center: after iteration 5. Right: iteration 10

As in all gradient descent methods, the determination of the step width Δ_t is crucial. Also, gradient methods imply the danger of being trapped in local minima. We tackle both problems with the determination of step with Δ_t and σ_t as described in the following section.

3.2 Cooling down the motion: time stepping Δ_t and parameter σ_t

The determination of step width parameter Δ_t in any gradient descent approach is a well known problem. At chosen too small results in inapplicably slow convergence behavior and is not robust too noise, Δ_t chosen too big might miss the optimum. In FFS, the step width Δ_t is used as a steering parameter of the algorithm in connection with the parameter σ_t which determines the influence of distance in the correspondence function. We designed Δ_t as exponentially decreasing, σ_t linearly decreasing.

A large Δ_t allows the scans to be massively relocated (shuffled), they overshoot their correct position in the direction of the correspondence gradient. Naturally, a small Δ_t moves the scans less (the amount of replacement is directly proportional to $\Delta^2 t$, as defined by the laws of movement). We chose the strategy of decreasing Δ_t and σ_t experimentally, having analogies of the cooling behavior of algorithms like simulated annealing in mind. The imprecise, non optimal large Δ_t at the beginning allows the system to possibly escape from local minima. Observe that in contrast to a technique like simulated annealing we cool down a gradient guided process, not a random state change or a random walk technique that would not be applicable in our high dimensional search space. We therefore avoid the problems with a high computational load (high number of iteration steps) that tend to appear in simulated annealing due to unguided selection of the next state.

The parameter σ_t in equation 2 steers the influence of distance in the computation of point correspondences . A large σ_t enhances the relative influence of data correspondences with greater distances, and, since it equalizes this spatial proximity property, favors the influence of visual similarity. A small σ_t emphasizes local proximity, which is useful if the global map is already close to an optimum.

The effect of cooling is shown in figure 11 showing our experimental results on the 'apartment data set'. Observe that the potential function (the fitness measure) 12 is not monotonically decreasing in the first iteration steps. This shows an 'overshooting' of the system due to a large Δ_t, in the data registration this can indicate an escape from a local minimum.

It is important to mention that after each iteration the system resets the velocity of each scan to zero. This guarantees that the system converges to a stable state (assuming $\Delta_t \rightarrow 0$).

3.3 Point direction and optional resampling

The 'point direction' is used in the correspondence equation 2 to assign to points the direction of an underlying linear structure. It is derived by modeling the point set with line segments using the extended EM algorithm described in (Latecki and Lakaemper, 2006). Utilizing a segment split and merge approach, the extended EM algorithm automatically adjusts number and location of the line segments in a way such that linear structures are represented (by a single, possibly long line segment) as well as round structures (by multiple short segments). Hence, even scenarios not being rich in linear structures are robustly represented. The algorithm was already successfully applied to model indoor and outdoor rescue scenarios. A 3*D* version of this algorithm for approximation of scan points with planar patches is described in (Lakaemper and Latecki, 2006).

The data of each scan s_i is approximated by a set of line segments L_i. The direction of a point p in s_i is the angular direction (in the scan's coordinate frame) of the closest line segment in L_i. The approximation of the data set with line segments results in a very stable and intuitive estimation of point directions. Figure 6 shows the influence of point directions for the correspondences.

The closest line segment l_j to the point p is called the *supporting line segment* if its distance to p is below a certain threshold. Points without supporting line segments are removed from the data set. Due to the nature of the extended EM algorithm, these removed points are points in areas with low point density. Low point density results from objects which are hit less than others. This is the result of either erroneous scanning, non static objects, or low scanning density, which by itself results from either long distance to an object or simply the fact that a certain location was only scanned a few times. All of these topics include uncertainty about the existence of the object, hence we disregard such points. The behavior of the extended EM segment fitting guarantees that safe, distinct objects are not removed. The removal of uncertain data increases the stability of the FES algorithm.

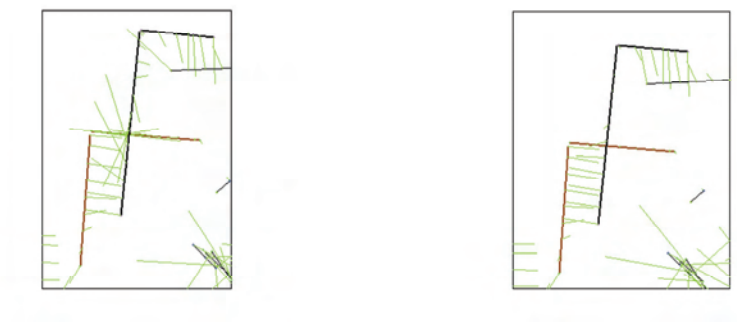


Figure 6. Left: forces between two scans (red lines belong to first scan, black lines to second scan) computed without direction information. Right: forces computed using L and K as described in example 2 (using equal masses). Correspondences between non parallel structures are weakened

Having the segments, the data can optionally be resampled along the supported line segments with an equal sampling distance. Such a point set has a more homogeneous distribution of points, which tends to be advantageous: experiments showed that homogeneous distributions are helpful to avoid local minima if the configuration of scans is

still far from the optimal solution, since over represented areas (e.g. features with unusual high scanning density due to multiple scans in a single location) are equalized. Additionally, the optional resampling can significantly speed up the computation if the number of data points reduces drastically due to the chosen sampling resolution, see section 3.4.1. If the data is resampled, only the line segments (two endpoints) are stored, also resulting in a significant data compression (typically about 1:100).

3.4 Computational complexity

3.4.1 Time complexity

The definition of C (equation 2) on pairs of data points leads to an algorithm with $O(n^2)$ time complexity where n is the number of points. This is certainly prohibitive for real applications. Different techniques can be used to reduce the complexity by taking advantage of two main properties of equation 2:

1. For each data point only its local neighborhood must be examined, since the forces between points rapidly decrease with distance. Hence some techniques successfully built into ICP implementations (which suffers from the same complexity problem) can be used to reduce the complexity, e.g. KD-trees. In the current implementation, we take advantage of the line segment representation of the data; we use a bounding box intersection approach on axis aligned bounding boxes around the line segments: bounding boxes around all line segments are computed and extended by $2\ \sigma_t$ in each direction. The force between two data points is computed only if the two corresponding lines' bounding boxes overlap. Hence we first reach a computational complexity based on the number $m << n$ *of* line segments, which is significantly lower than the number n of data points. Secondly, though bounding box intersection is an $O(m \log m)$ computation (note again: m=number of segments), update techniques as reported in (Cohen et al., 1995), reduce the expected complexity to $O(m)$. This linear complexity is reported under the constraint of 'relatively small' movements of objects, such that the $O(m \log m)$ sorting in the sweep and prune step reduces to $O(m)$ on a nearly presorted list. The constraint of small movements is met for most of the iteration steps in FFS. To give an idea of the order of magnitude of reduction that is achieved some numbers for the NIST data set (see section 4.2) should be mentioned:
   - 60 scans contain a total of 21420 points, represented by 332 line segments (on average: 65 points per segment)
   - average number of colliding pairs of segments per iteration 1500, hence we have 65 x 65 x1500(= 6, 337,500) computations, compared to 21420^2(460,000,000).
2. The data points have to be evaluated with a certain accuracy only. By approximating the evaluation of force field we can achieve computational reduction in the following 2 manners:
   - The current FFS implementation sub samples each segment equally with some sampling distance. For the NIST data set (sampling distance 10cm), we achieve in average 7 data points per segment, the force computation is therefore reduced to 7 x 7 x 1500(= 73, 500) computations.
   - Greengard and Strain introduced Fast Gauss Transform (FGT) (Greengard and Strain, 1991) which is in turn based on Fast Multipole Methods introduced for high speed simulation of particle dynamics in potential fields (Greengard and Rokhlin, 1987). The main advantage with FGT is that the force field can be computed in

linear time with a constant factor depending on the precision required in computation of the field. Details can be found in (Greengard and Strain, 1991). The main idea is to compute the force field using a divide and conquer strategy and exploiting Hermite and Taylor expansions. FGT has been first introduced in (Elgammal et al., 2003; Ayyagari et al., 2005).

3.4.2 Space complexity

Since we approximate the scans by segments , it is not necessary to keep the original data. For each scan, only the segments' endpoints have to be stored. Experiments with the extended EM algorithm (Latecki and Lakaemper, 2006) on 2D Laser data sets show an average compression rate of 1:100 (200 data points per segment).

3.5 Online FFS

The described algorithm easily can be extended to online SLAM, i.e. scans are recorded and processed subsequently, as they arrive from the laser device. The extension is canonical: each additional scan is pre aligned, then FFS runs on the previously aligned data set plus the new scan.The current FFS system targets the application of multi robot mapping, hence the sequential processing is not implemented yet.

Algorithm 1 Force field based mapping

l: Compute supporting line segments (section 3.3)
2: Resample data set(section 3.3)
3: $S \leftarrow$ initial state of transformations
4: initialize step width Δ_t and σ_t (section 3.2)
5: **repeat**
6: Assign masses to data points
7: Compute forces using σ_t (eq. 3)
8: Assign constant mass values to data points
9: Compute rotational and translational acceleration (eq.7 and eq. 6)
10: Compute transformations ∇g
11: $g \leftarrow g + \nabla g$
12: Set velocity of all points to zero
13: update σ_t and Δ_t (section 3.2)
14: **until** average of relocations > Threshold

3.6 Pre Alignment

The pre alignment does not make use of odometric sensor data, but is based on shape similarity. It finds distinct shape features in single scans and tries to find an optimal overlap based on the shape similarity of these features. For further details see (Adluru et al., 2006).

4. Experimental results

4.1 Performance Comparison to classical ICP

Fig. 7 shows the difference between the results of aligning a hypothetical set of 3 simple scans using classical ICP and our approach. Due to the hard constraints of using the nearest point correspondence only ICP (top row) ends in a non perceptually optimal configuration.

FFS takes into account the correspondences between all points first, a decreasing σ_t finally guarantees the correct positioning of the scans, decreasing the influence of points being too far away. The bottom right image shows the result of FFS after 12 iteration steps.

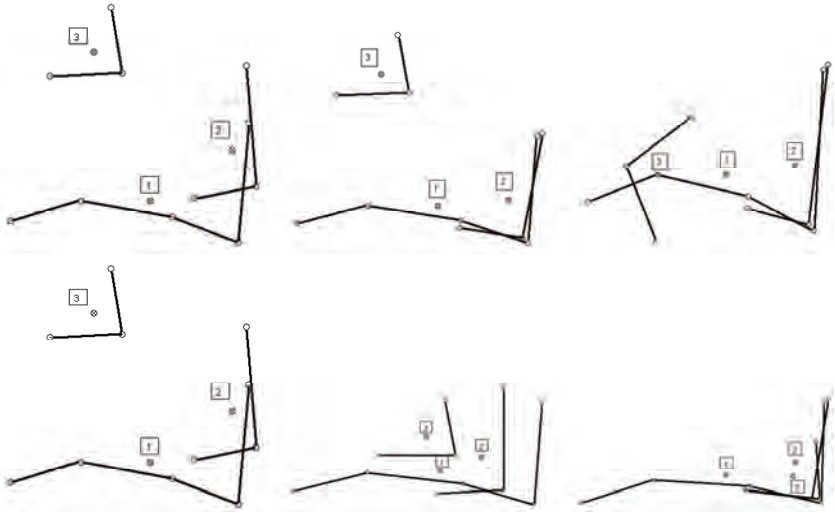


Figure 7. The top row shows 3 steps of the alignment of 3 scans (each scan consists of a single corner only) by classical ICP, the bottom-row shows the results of the proposed approach. The alignment progress can be seen from left to right in both cases. The square boxes show the robot poses of the the scans

4.2 NIST disaster area

The NIST data set used in this experiment simulates a typical data set of multi robot mapping in rescue scenarios. It is especially complicated, as it matches the complicated constraints imposed by these settings, which contain only imprecise odometry, no landmarks and very little overlap.

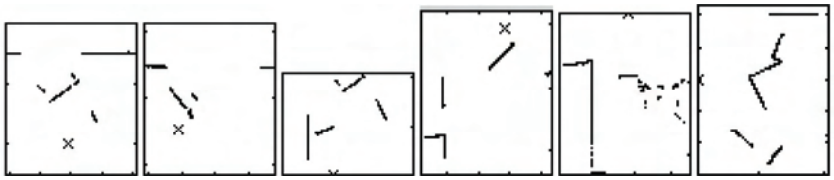


Figure 8. 6 out of 60 scans of the NIST rescue scenario data set. The scans in this data set are very sparse and have minimal overlap

The data set consists of 60 scans taken from 15 different positions in directions E,N,W,S with an overlap of 5° (i.e. overlap between E and N, N and W etc.). The area has a size of approx. 10 x 15m, the 15 locations differ approx. 2*m* from each other (see Figure 9). The distance between the positions of the 4 scans taken from an assumed single position differs up to 30cm, with an angular error of up to 20° to the assumed direction. This data set can be seen as a multi robot mapping scenario using 15 robots, with 4 scans gained from each robot.

Although the pre-alignment assumes this setting, FFS actually treats the 60 scans as independent scans without help of any further information, e.g. constraints on the groups of 4. The single scans have very little pair wise overlap. Figure 9, (l)-(6) shows 6 example scans, all located on the left side of the global map; the overlapping pairs among these scans are (1,2), (1,3),(1,4), (2,3), (3,4), (3,6), (4,5).

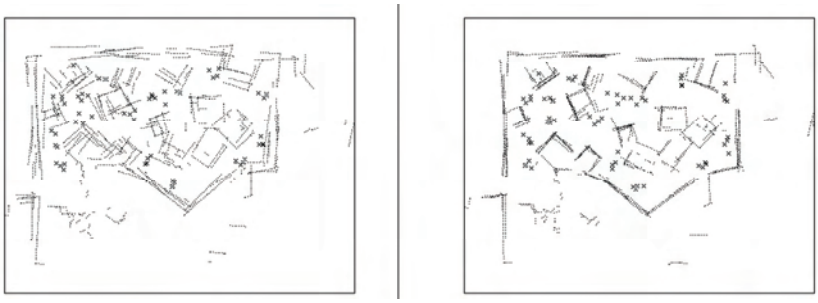


Figure 9. left: 60 scans superimposed building a global map using a rough initial transformation estimation, right: after 20 iterations of FFS. The crosses show the robots' positions, (the reader might try to find the 6 single scans of fig. 8 in the global map. 1,2,3 and 6 are part of the upper left corner, 4 and 5 are located in the lower left corner). The final result of FFS is shown in fig. 10, left

The test performed on this complicated data set demonstrates the robustness of the FFS system. The initial, pre-alignment map is gained by the shape-based algorithm described in Section 3.6. Fig. 9 shows the initial global map as well as iteration step 20, fig. 10 the final global map, after 50 iterations. The data set was re-sampled as described in section 3.3. The radius of the ROI was set to 5 cm, the parameters for the motion cooling were set as:

- Δ_t decreases from 5 to 1 with step factor of 0.96
- σ_t decreases from 15 to 4 with step size of 0.25

Although the data is poorly pre aligned and the overlap between the single scans is minimal, FFS successfully reconstructs the global map, which proves its applicability for this multi robot setting. The mean translation/rotation of the scans (translation/rotation between initial and final global map) is 16cm/4°, the maximum translation/rotation is 50cm/10.5°.

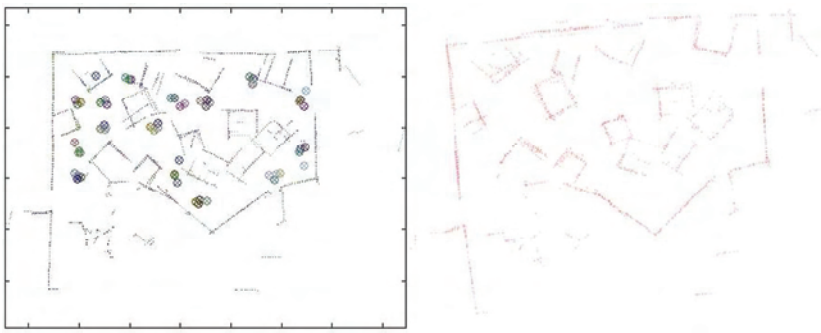


Figure 10. Left, final map (after 50 iterations) of intialization fig. 9 with FFS. Right, the final map obtained by the Lu & Milios technique as reported in (Borrmann et al., 2007). The systems lead to results of comparable quality

The alignment can also be seen as a movie at: http://knight. cis .temple .edu/~lakaemper/FFS/ FFSTheMovie.wmv
The movie especially makes clear the effect of the motion cooling.
Figure 10 shows the result after 50 iteration steps. The computational time for each iteration is 1 second on a l.SGHz desktop computer in the current MATLAB implementation, using the bounding box approach as described in section 3.4.1, and resampling with a distance of 10cm.
In order to compare the proposed FFS approach to the state of the art of existing robot mapping approaches, we applied three influential approaches to the NIST data set illustrated in Figure 9. We applied the particle filter based DP-SLAM (Eliazar and Parr, 2004), the ICP based VASCO robot mapping module of CARMEN, and improved grid-based SLAM with Rao-Blackwellized Particle Filters (Grisetti et al., 2005). All three approaches failed to produce any reasonable results, since they are based on sequential processing of data (online SLAM), which can not be applied on this data set due to the extremely minimal overlap of consecutive scans (even if the order is known).
However, we compared to a recent implementation of Lu/Milios type SLAM (Borrmann et al., 2007). The results are shown in figure 10. Both algorithms show an overall comparable performance, although local differences can be seen: the Lu/Milios type SLAM reconstructs the top right corner better, while FFS performs better on the left side.
Figure 12, left, shows the potential $P(g)$ vs. iteration curve for this data set. The potential is monotonically decreasing, hence in this case FFS steers directly towards a (local) minimum, which is reasonable due to the initialization. The next experiment will show a different case.

4.3 Apartment

This experiment demonstrates the benefits and applicability of FFS in data sets which are incorrectly pre aligned e.g. due to effects of wrong loop closing. We used the IROS 2006 test data set taken from http: //staff.science.uva.nl/ ~ zivkovic/FS2HSC/dataset. html. The data set consists of about 2000 scans from which we select every 10^{th} scan. Thus, our test data set consists of 200 scans taken from a single robot in an apartment of size about 16 x 8m. As shown in Fig. 11(1), the pre-alignment gained shows a huge error, additionally the alignment is very imprecise (blurred features). The experiment shows the power of FFS to escape local minima: starting with a large stepping parameter Δ_t, the first transformation blurs the data set and therefore weakens the wrong correspondences, giving space for new connections, Fig. 11(2). Transforming all scans in parallel eventually results in a version of the map, which not only shows the misaligned parallel walls correctly contracted but also corrected the huge error as shown in Figure 11(4). The values of the parameters are equal to the experiment in Section 4.2. Figure 11 also shows a limit of the algorithm: one single initially strongly misaligned scan does not find consistent correspondences and therefore can not be correctly re positioned by the algorithm. It stays in its incorrect position. We assume that no algorithm working only on low level perceptual features is able to handle such a strong error correctly; mid level cognitive correspondences are needed. However, mid level perceptual features can easily be integrated into the system using correspondence functions modeling these perceptual forces, which will be part of the future work on the system.

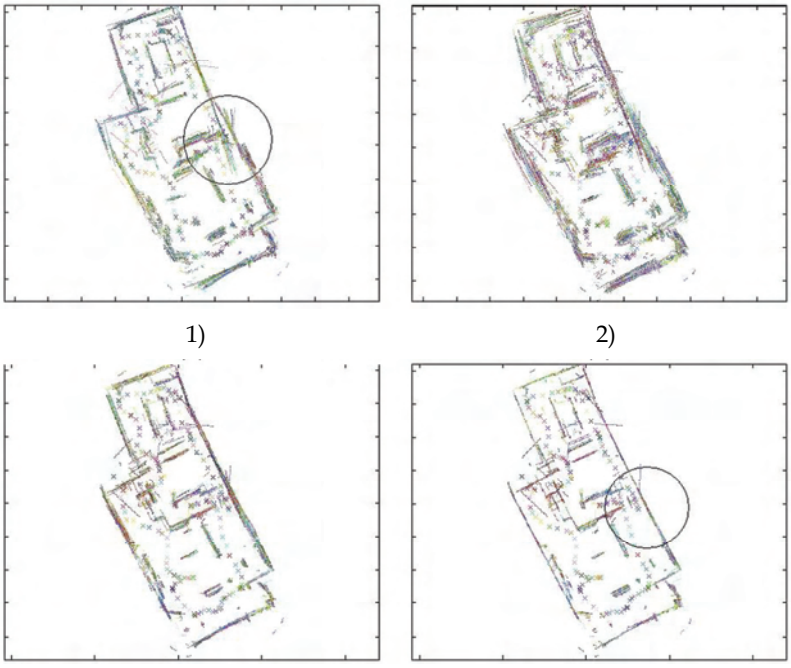


Figure 11. Apartment data set. 1) initial configuration. The circled area shows an error due to incorrect loop closing. 2) A large step parameter dT blurs the map in the first iteration step to escape from the local minimum 3) iteration 50: 4) iteration 150: FFS has not only contracted the edges given in 1), but also has realigned the entire global map to fix the error (circled area)

4.4 NIST maze

This data set consists of 16 scans with similar structures, a typical indoor environment, yet again scanned with minimal overlap. See figure 13 and 14 for this experiment.

5. Conclusion and future work

We presented a new approach to the problem of multi robot mapping under the constraints given in rescue scenarios. It does not rely on odometry i.e. the relative pose between the robots is unknown. It also can deal with the problem of extremely minimal overlap. Experiments conducted on a real data set of a disaster area from NIST shows the performance of the FFS approach under these complicated constraints and proved its applicability to the problem of multi robot mapping, they also proved the excellent performance of the algorithm correcting effects from wrong pre alignment. The future work will mainly focus on detection of higher level features: the modeling of the correspondence function with respect to the masses opens different ways to interface to mid level modules. The approach is easily extendable to 3*D*, a report about the performance of an implementation of the 3*D* FFS is the topic of a forthcoming paper.

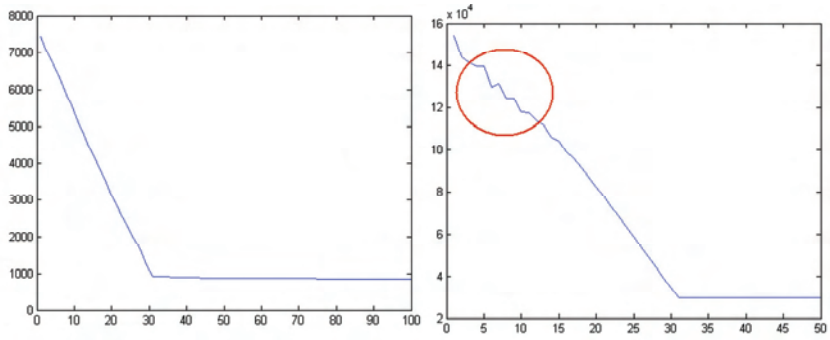


Figure 12: Left, potential vs. iterations of FFS for disaster data. Right, potential for Apartment data set. The potential (encircled) of the apartment data is not monotonically decreasing, indicating a possible escape from a local minimum

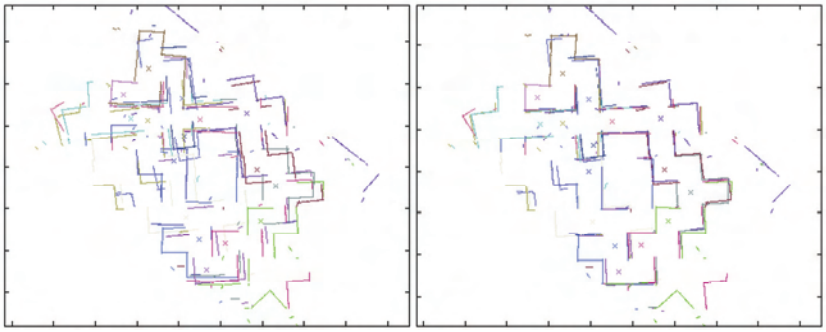


Figure 13. Left, initial configuration of NIST's maze data (16 scans). Right, after 5 iterations of FFS

6. Acknowledgment

We thank Andreas Nuechter from University of Osnabrueck, Germany, for his helpful comments and the code for the experiments on 3D-Lu/Milios-SLAM.

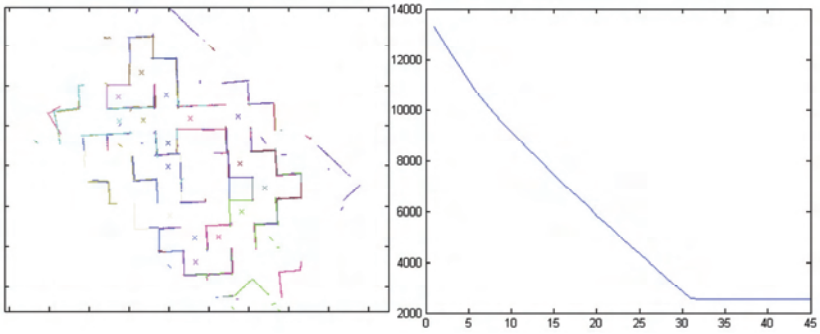


Figure 14. Left, final map obtained with FFS. Right, the potential vs. iterations plot

7. References

Adluru, N., Latecki, L. J., Lakaemper, R., and Madhavan, R. (2006). Robot mapping for rescue robots. In *Proc. of the IEEE Int. Workshop on Safety, Security and Rescue Robotics (SSRR),* Gaithersburg, Maryland, USA.

Ayyagari, V. R., Boughorbel, F., Koschan, A., and Abidi, M. A. (2005). A new method for automatic 3d face registration. In CVPR *'05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Workshops,* page 119, Washington, DC, USA. IEEE Computer Society.

Besl, P. and McKay, N. (1992). A method for registration of 3-d shapes. IEEE *Transactions on Pattern Analysis and Machine Intelligence,* 14(2):239-256.

Biber, P. and Strasser, W. (2003). The normal distributions transform: A new approach to laser scan matching. In *In Proceedings of the IEEE/RS] International Conference on Intelligent Robots and Systems.*

Biber, P. and Strasser, W. (2006). nScan-matching: simultaneous matching of multiple scans and application to slam. In *Robotics and Automation. ICRA Proceedings IEEE International Conference on,* pages 2270- 2276.

Birk, A. (1996). Learning geometric concepts with an evolutionary algorithm. In *Proc. of The Fifth Annual Conference on Evolutionary Programming.* The MIT Press, Cambridge.

Birk, A. and Carpin, S. (2006). Merging occupancy grid maps from multiple robots. In *Proceedings of the IEEE,* volume 94, pages 1384 -1397.

Borrmann, D., Elseberg, J., Lingemann, K., Nuchter, A., and Hertzberg, J. (2007). The Extension of Lu and Milios Style SLAM to 6 Degree of Freedom. In *IROS 2007, (submitted).*

Boughorbel, R, Koschan, A., Abidi, B., and Abidi, M. (2004). Gaussian fields: a new criterion for 3d rigid registration. *Pattern Recognition,* 37:1567-1571.

Chen, Y. and Medioni, G. (1992). Object modelling by registration of multiple range images. *Image Vision Comput.,* 10(3):145-155.

Cohen, J. D., Lin, M. C., Manocha, D., and Ponamgi, M. K. (1995). I-COLLIDE: An interactive and exact collision detection system for large-scale environments. In *Symposium on Interactive 3D Graphics,* pages 189-196,218.

Eggert, D. W., Fitzgibbon, A. W., and Fisher, R. B. (1998). Simultaneous registration of multiple range views for use in reverse engineering of CAD models. *Computer Vision and Image Understanding: CVIU,* 69(3):253-272.

Elgammal, A., Duraiswami, R., and Davis, L. S. (2003). Efficient kernel density estimation using the fast gauss transform with applications to color modeling and tracking. IEEE *Trans. Pattern Anal. Much. Intel!.,* 25(11):1499-1504.

Eliazar, A. and Parr, R. (2004). DP-SLAM 2.0. In *Proc. of IEEE International Conference on Robotics and Automation.*

Frese, U. (2006). Treemap: An o(log n) algorithm for indoor simultaneous localization and mapping. *Auton. Robots, 2l(2):W3-l22.*

Frese, U., Larsson, P., and Duckett, T. (2005). A multilevel relaxation algorithm for simultaneous localization and mapping. *Robotics, IEEE Transactions on Robotics and Automation,* 21:196-207.

Greengard, L. and Rokhlin, V. (1987). A Fast Algorithm for Particle Simulations. *Journal of Computational Physics,* 73:325-348.

Greengard, L. and Strain, J. (1991). The fast gauss transform. *SIAMJ. Sci. Stat. Comput.,* 12(l):79-94.

Grisetti, G., Stachniss, C, and Burgard, W. (2005). Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling. In *In Proceedings of the IEEE Int. Conference on Robotics & Automation (ICRA),* pages 2443-2448.

Jalba, A. C., Wilkinson, M. H., and Roerdink, J. B. (2004). CPM: A deformable model for shape recovery and segmentation based on charged particles. IEEE *Transactions on Pattern Analysis and Machine Intelligence,* 26(10):1320-1335.

Konolige, K. (2003). Map merging for distributed robot navigation. In *Proceedings of the 2003 IEEE International Conference on Intelligent Robots and Systems,* pages 212-217, Las Vegas, NV.

Lakaemper, R. and Latecki, L. J. (2006). Decomposition of 3D laser range data using planar patches. In IEEE *Int.* Con/, *on Robotics and Automation (ICRA).*

Latecki, L. J. and Lakaemper, R. (2006). Polygonal approximation of laser range data based on perceptual grouping and EM. In IEEE *Int.* Con/, *on Robotics and Automation (ICRA).*

Lu, F. and Milios, E. (1994). Robot pose estimation in unknown environments by matching 2D range scans. In *CVPR94,* pages 935-938.

Lu, F. and Milios, E. (1997). Globally consistent range scan alignment for environment mapping. *Auton.* Robots, 4(4) :333-349.

Minguez, J., Montesano, L., and Lamiraux, F. (2006). Metric-based iterative closest point scan matching for sensor displacement estimation. *Robotics, IEEE Transactions on Robotics,* pages 1047-1054.

Montemerlo, M., Thrun, S., Roller, D., and Wegbreit, B. (2002). FastSLAM: A factored solution to the simultaneous localization and mapping problem. In *Eighteenth national conference on Artificial intelligence,* pages 593-598, Menlo Park, CA, USA. American Association for Artificial Intelligence.

N.Paragios, M.Rousson, and V.Ramesh (2003). Non-rigid registration using distance functions. In *Computer Vision and Image Understanding,* volume 89, pages 142-165.

Olson, E., Leonard, J., and Teller, S. (2006). Fast iterative optimization of pose graphs with poor initial estimates. In ICRA

Robertson, C. and Fisher, R. (2002). Parallel evolutionary registration of range data. In *Computer Vision and Image Understanding,* volume 87, pages 39-50.

Rusinkiewicz, S., Brown, B., and Kazhdan, M. (2005). 3D scan matching and registration, *ICCV Short Course.*

Rusinkiewicz, S. and Levoy, M. (2001). Efficient variants of the ICP algorithm. In *Proceedings of the Third Intl. Conf. on 3D Digital Imaging and Modeling,* pages 145-152.

Thrun, S., Koller, D., Ghahramani, Z., Durrant-Whyte, H., and Ng, A. (2002). Simultaneous mapping and localization with sparse extended information filters. In Boissonnat, J.-D., Burdick, J., Goldberg, K., and Hutchinson, S., editors, *Proceedings of the Fifth International Workshop on Algorithmic Foundations of Robotics,* Nice, France. Forthcoming.

Veltkamp, R. and Hagedoorn, M. (1999). State-of-the-art in shape matching. *Technical Report* UU-CS-1999-27, Utrecht University, The Netherlands.

Wertheimer, M. (1958). Principles of perceptual organization, *Readings in Perception(D. Beardske and M. Wertheimer, Eds.). Princeton, N]: Princeton University Press.*

Xu, C. and Prince, J. (1998). Snakes, shapes, and gradient vector flow. IEEE *Transactions on Image Processing,* pages 359-369.

Yang, R., Mirmehdi, M., and Xie, X. (2006). A charged active contour based on electrostatics. In *Advanced Concepts for Intelligent Vision Systems, ACIVS,* pages 173-184. Springer-Verlag LNCS 4179.

4

Flocking Controls for Swarms of Mobile Robots Inspired by Fish Schools

Geunho Lee and Nak Young Chong
School of Information Science, Japan Advanced Institute of Science and Technology
Japan

1. Introduction

Self-organizing and adaptive behaviors can be easily seen in flocks of birds or schools of fish. It is surprising that each individual member follows a small number of simple behavioral rules, resulting in sophisticated group behaviors (Wilson, 2000). For instance, when a school of fish is faced with an obstacle, they can avoid collision by being split into a plurality of smaller groups that can be merged after they pass around the obstacle. Based on the observation of such habits of schooling fishes, we propose collective navigation behavior rules that enable a large swarm of autonomous mobile robots to flock toward a stationary or moving goal in an unknown environment. Recently, robot swarms are expected to be deployed in a wide variety of applications such as odor localization, mobile sensor networking, medical operations, surveillance, and search-and-rescue (Sahin, 2005). In order to perform those tasks successfully, the behaviors of individual robots need to be controlled in a simple manner to support coordinated group behavior.

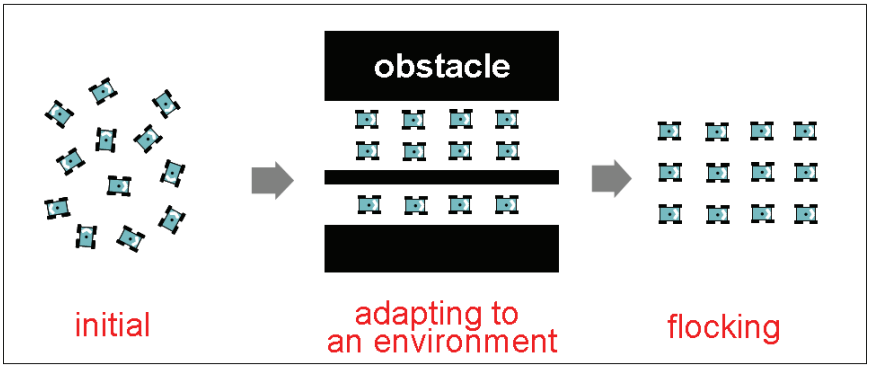
Reynolds presented a distributed behavioral model of coordinated animal motion based on fish schools and bird flocks (Reynolds, 1987). His work demonstrated that navigation is an example of emergent behavior arising from simple rules. Many navigation strategies reported in the field of swarm robotics can be classified into centralized and decentralized strategies. Centralized strategies (Egerstedt & Hu, 2001) (Burgard *et al*, 2005) employ a central unit that organizes the behaviors of the whole swarm. This strategy usually lacks scalability and becomes technically unfeasible when a large swarm is considered. On the other hand, decentralized strategies are based on interactions between individual robots mostly inspired by evidence from biological systems or natural phenomena. Decentralized strategies can be further divided into biological emergence (Baldassarre *et al*, 2007) (Shimizu *et al*, 2006) (Folino & Spezzano, 2002), behavior-based (Ogren & Leonard, 2005) (Balch & Hybinette, 2000), and virtual physics-based (Spears et al, 2006) (Esposito & Dunbar, 2006) (Zarzhitsky et al, 2005) approaches. Specifically, the behavior-based and virtual physics-based approaches are related to the use of such physical phenomena as crystallization (Balch & Hybinette, 2000) gravitational forces (Spears *et al*, 2005) (Zarzhitsky *et al*, 2005) (Spears *et al*, 2004) and potential fields (Esposito & Dunbar, 2006). Those works mostly use a force balance between inter-individual interactions exerting an attractive or repulsive force within the influence range, which might over-constrain the swarm and frequently lead to deadlocks. Moreover, the computations of relative velocities or accelerations between robots

are needed to obtain the magnitude of the force. Regarding the aspect of calculating the movement position of each robot, accuracy and computational efficiency issues will arise.
In this paper, from the observation of the habits of schooling fishes, a geometrical motion planning framework locally interacting with two neighbor robots in close proximity is proposed, enabling three neighboring robots to form an equilateral triangle lattice. Based on the local interaction, we develop an adaptive navigation approach that enables a large swarm of autonomous mobile robots to flock through an unknown environment. The proposed approach allows a swarm of robots to split into multiple groups or merge with other groups according to the environmental conditions. Specifically, it is assumed that individual robots are not allowed to have any unique identifier, a pre-determined leader, a common coordinate system, any memory for past decisions and actions, and a direct communication with each other. Given these underlying assumptions, all robots execute the same algorithm and act independently and asynchronously of each other. In spite of such minimal conditions, the above-mentioned potential applications often require a large-scale swarm of robots to navigate toward a certain direction from arbitrary initial positions of the robots in an environment populated with obstacles. For instance, in exploration and search-and-rescue operations, robot swarms need to be dispersed into an unknown area of interest in a uniform spatial density and search for targets. Consequently, the proposed approach provides an efficient yet robust way for robot swarms to self-adjust their shape and size according to the environment conditions. This approach can also be considered as an *ad hoc* mobile networking model whose connectivity must be maintained in a cluttered environment.
The rest of this paper is organized as follows. Section 2 presents the robot model and the statement of the swarm flocking problem. Section 3 describes the basic motion planning of each individual robot locally interacting with neighboring robots. Section 4 presents a collective solution to the swarm flocking problem. Section 5 illustrates how to extend the solution algorithms to the swarm tracking problem. Section 6 provides the results of simulations and discussion. Section 7 draws conclusions.

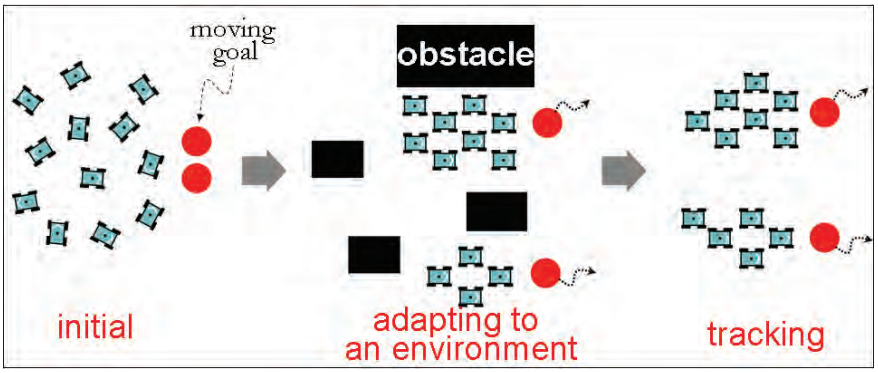
2. Problem Statement

We consider a swarm of n autonomous mobile robots, where individual robots are denoted respectively by $r_1, r_2, \cdots, r_n$. Each robot is modeled as a point, which freely moves on a two-dimensional plane. It is assumed that the initial distribution of robots is arbitrary and distinct. The robots have no leader and no unique identification numbers. They do not share any common coordinate system, and do not retain any memory of past actions that gives inherently self-stabilizing property[1] (Suzuki & Yamashita 1999). They can detect the positions of other robots within their limited ranges of sensing, but do not have any explicit direct means of communication to each other. Each of the robots executes the same algorithm, but acts independently and asynchronously from other robots. They repeat an endless activation cycle of observation, computation, and motion.

[1] Self-stabilization is the property of a system which, started in an arbitrary state, always converges toward a desired behavior (Dolev, 2000) (Schneider, 1993).



(a) adaptive flocking



(b) adaptive tracking

Figure 1. Illustration of two flocking control problems

Denote the distance between any two robots r_i and r_j, located respectively at p_i and p_j, as $dist(p_i, p_j)$. Also denote a constant distance as d_u that is finite and greater than zero. Each robot has a limited sensing boundary *SB*. Then r_i detects the positions of other robots, $\{p_1, p_2, \cdots\}$, located within its *SB*, and makes a set of the observed positions O_i obtained with respect to its local coordinate system. From O_i, r_i can select two specific robots r_{s1} and r_{s2}, respectively. We call r_{s1} and r_{s2} the *neighbor* of r_i, and define their positions $\{p_{s1}, p_{s2}\}$ as the *neighbor set* N_i. Given p_i and N_i, *Triangular Configuration* is defined as a set of three distinct positions $\{p_i\, p_{s1}, p_{s2}\}$ denoted by T_i. Next, we can define *Equilateral Configuration* E_i if and only if all the possible distance permutations $dist(p_{\pi(i)}, p_{\pi(j)})$ in T_i are equal to d_u.

In this paper, each robot attempts to follow a certain rule to generate E_i from an arbitrary T_i. We formally define each individual robot's behavior as *Local Interaction*, which allows the position of r_i to be maintained to be d_u with N_i at each time toward forming E_i. Now, we can address the following problem of *Flocking Controls* for a swarm of robots based on local interactions (see Fig. 1):

- (***Flocking Controls***) Given $r_1,\cdots,r_n$ located at arbitrarily distinct positions in a two dimensional plane, how to enable the robots to move toward a stationary or moving goals while adapting to an environment populated with obstacles.

3. Local Interaction

ALGORITHM - 1 LOCAL INTERACTION (code executed by each robot r_i)

constant d_u := a uniform distance

Function $\varphi_{interaction}(O_i, p_i)$

1 $(p_{ct,x}, p_{ct,y})$:= centroid(p_i, $\{p_{s1}, p_{s2}\}$)
2 ϕ := angle between $p_{s1}p_{s2}$ and $r_i's$ local horizontal axis
3 $p_{ti,x} := p_{ct,x} + d_u \cos(\phi + \pi/2)/\sqrt{3}$
4 $p_{ti,y} := p_{ct,y} + d_u \cos(\phi + \pi/2)/\sqrt{3}$
5 $p_{ti} := (p_{ti,x}, p_{ti,y})$

Local geometric shapes of a school of tuna are known to form a diamond shape (Stocker, 1999), whereby tunas exhibit the following schooling behaviors: maintenance, partition, and unification. Similarly, local interaction for a swarm of robots in this paper is to generate an equilateral triangular lattice. This section explains how the local interaction is established among three neighboring robots.

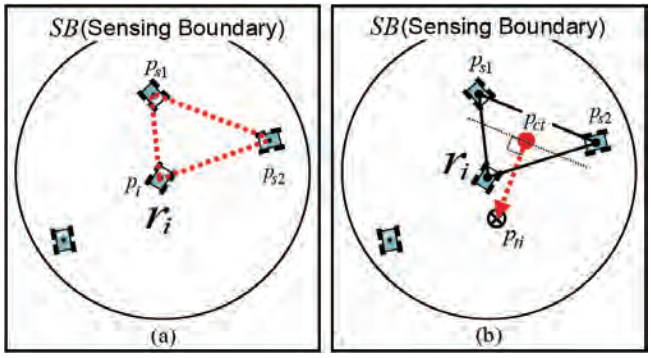


Figure 2. Illustration of local interaction ((a) triangular configuration, (b) target computation))

As presented in ALGORITHM-1, the algorithm consists of a function $\varphi_{interaction}$ whose arguments are p_i and N_i at each activation step. Consider any robot r_i and its two neighbors r_{s1} and r_{s2} located within its *SB*. As shown in Fig. 2-(a), three robots are configured into T_i whose vertices are p_i, p_{s1} and p_{s2}, respectively. First, r_i finds the centroid of the triangle $\Delta p_i p_{s1} p_{s2}$, denoted by p_{ct}, with respect to its local coordinates, and measures the angle ϕ between the line connecting the two neighbors and r_i's horizontal

horizontal axis. Using p_{ct} and ϕ, r_i calculates the target point p_{ti} as illustrated in Fig. 2-(b). Each robot computes the target point by their current observation of neighboring robots. Intuitively, under ALGORITHM-1, r_i may maintain d_u with its two neighbors at each time. In other words, each robot attempts to form an isosceles triangle for N_i at each time, and by repeatedly doing this, three robots configure themselves into E_i.

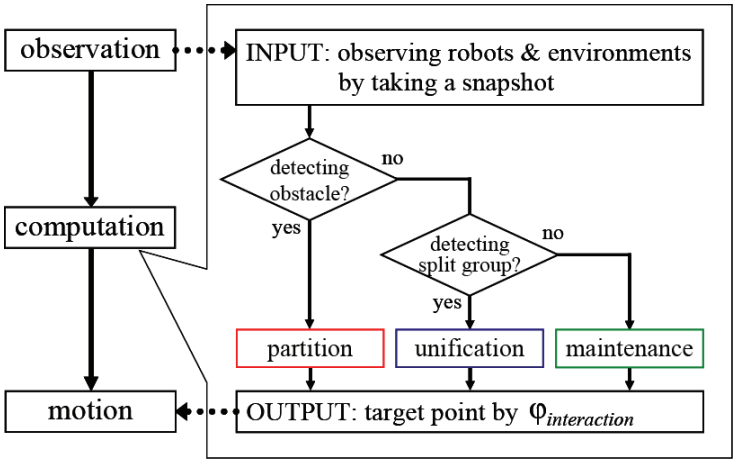


Figure 3. Adaptive flocking flowchart

4. Adaptive Flocking Algorithm

4.1 Architecture of Adaptive Flocking

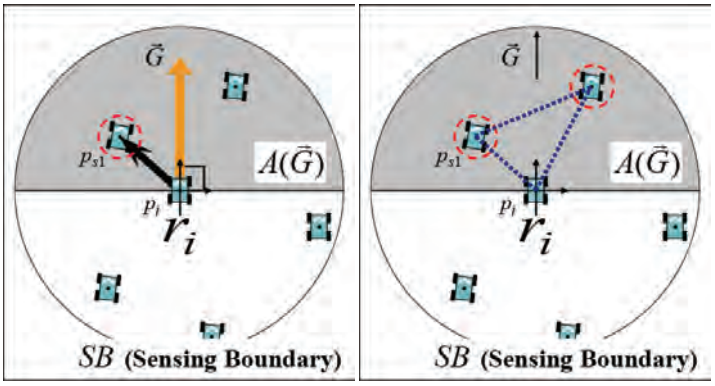
The adaptive flocking problem addressed in Section 2 can be decomposed into three sub-problems as illustrated in Fig. 3, each of which is solved based on the same local interaction (see Section 3).

- *Maintenance*: Given that robots are located at arbitrarily distinct positions, how to enable the robots to flock in a single swarm.
- *Partition*: Given that an environmental constraint is detected, how to enable a swarm to split into multiple smaller swarms adapting to the environment.
- *Unification*: Given that multiple swarms exist in close proximity, how to enable them to merge into a single swarm.

As illustrated in Fig. 3, the input of the algorithm for each time instant is O_i and the environment information with respect to the local coordinate system of each robot. The output is p_{ti} computed by $\varphi_{interaction}$. At each time, r_i can either be idle or execute their algorithm, repeating recursive activation at each cycle. At each cycle, each robot computes their movement positions (computation), based on the positions of other robots (observation), and moves toward the computed positions (motion). Through this activation cycle, when the robot finds any geographical constraint within its *SB*, the robot executes the *partition algorithm* to adapt its position to the constraint. On the other hand, when the robot finds no geographical constraint, but observes any robot around the outside of its group, the

robot executes the *unification algorithm*. Otherwise, the robot basically executes the *maintenance algorithm* while navigating toward a goal.

4.2 Team Maintenance



(a) 1st neighbor selection (b) 2nd neighbor selection

Figure 4. Illustration of team maintenance

The first problem is how to maintain a uniform interval among individual robots while navigating. This enables the robots to form a multitude of equilateral triangle lattices. Each robot adjusts $\vec{G}$, termed the goal direction, with respect to its local coordinates and computes O_i at the time *t*. As illustrated in Fig. 4-(a), let $A(\vec{G})$ denote the area of goal direction defined within the robot's *SB*. Next, each robot checks whether there exists a neighbor in $A(\vec{G})$. If multiple neighbors exist, r_i selects the first neighbor r_{s1} located the shortest distance away from p_i and defines its position as p_{s1}. Otherwise, r_i spots a virtual point p_v located an adequate distance d_v away from p_i along $\vec{G}$, defined as p_{s1}. As shown in Fig. 4-(b), the second neighbor r_{s2} is selected such that the total distance from p_{s1} to p_i passing through p_{s2} is minimized. As a result, p_{ti} can be obtained by $\varphi_{interaction}$ in ALGORITHM-1.

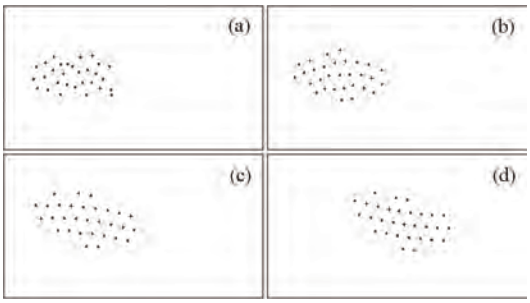
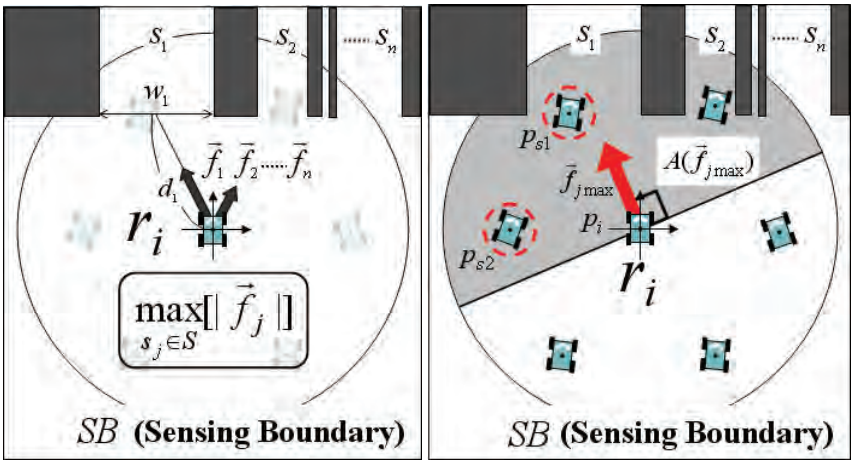


Figure 5. Simulation for maintenance algorithm ((a) initial distribution, (b) 2 sec. (c) 4 sec. (d) 11 sec.)

Fig. 5 shows the simulation results of maintenance algorithm with 30 robots under no environmental constraints. Initially, robots are arbitrarily located on the two-dimensional plane. As shown in Figs. 5-(b) and (c), each robot generates its geometric configuration with their neighbors while moving toward a goal. Fig. 5-(d) illustrates that robots maintain a single swarm while navigating. Once the target is detected by any of the robots closest to the goal, the swarm could navigate toward the goal through individual local interactions.

4.3 Team Partition



(a) favorite vector (b) neighbor selection

Figure 6. Illustration of team partition

When a swarm of robots detects an obstacle in its path, each robot is required to determine its direction toward the goal avoiding the obstacle. In this paper, each robot determines their direction by using the relative degree of attraction of the passageway (Halliday *et al.*, 2007), termed the favorite vector $\vec{f}$, whose magnitude is given by

$$|\vec{f}_j| = |w_j / d_j^2| . \tag{1}$$

In Fig. 6-(a), s_j denotes the passageway with width w_j, and d_j denotes the distance between the center of w_j and p_i. Note that if r_i can not exactly measure w_j beyond its *SB*, w_j is set to the maximum value of *SB*. Now the passageways can be represented by a set of favorite vectors $\{\vec{f}_j \,|\, 1 \le j \le n\}$ and then r_i selects the maximum magnitude of $\vec{f}_j$ denoted as $|\vec{f}_j|_{\max}$. As shown in Fig. 6-(b), r_i defines a maximum favorite area $A(\vec{f}_{j\max})$ based on the direction of $|\vec{f}_j|_{\max}$ within its *SB*. Next, r_i checks whether there exists a neighbor in $A(\vec{f}_{j\max})$. If neighbors are found, r_i selects r_{s1} located the shortest distance away from itself to define p_{s1}. Otherwise, r_i spots a virtual point p_v located at an adequate distance d_v in the direction of $|\vec{f}_j|_{\max}$ to define p_{s1}. Finally r_{s2} is selected such that the total distance from p_{s1} to p_i passing through p_{s2} is minimized. As a result, p_{ti} can be obtained by $\varphi_{interaction}$ in ALGORITHM-1.

In Fig. 7, there existed three passageways in the environment. Based on the proposed algorithm, robots could be split into three smaller groups while maintaining the local geometric configuration. Through the local interactions, the rest of the robots could naturally adapt to an environment by just following their neighbors moving ahead toward the goal.

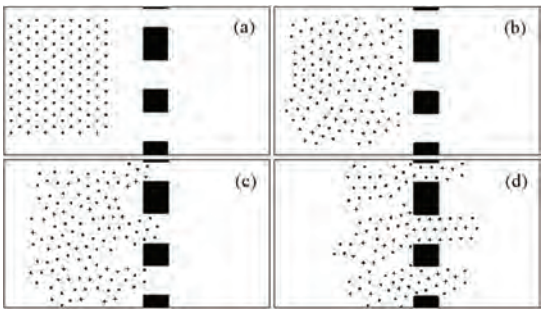
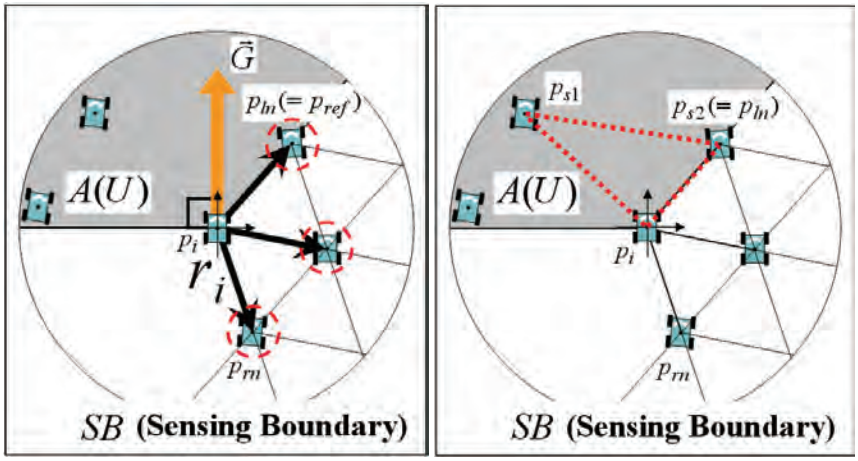


Figure 7. Simulation for partition algorithm ((a) initial distribution, (b) 5 sec. (c) 9 sec. (d) 18 sec.)

4.4 Team Unification



(a) unification area (b) neighbor selection

Figure 8. Illustration of team unification

In order to enable the multiple swarms in close proximity to merge into a single swarm, r_i adjusts $\vec{G}$ with respect to its local coordinates and defines the position set of robots D_u located within the range of d_u. Let $ang(\vec{m},\vec{n})$ be an angle between two arbitrary vectors $\vec{m}$ and $\vec{n}$. As shown in Fig. 8-(a), r_i computes $ang(\vec{G}, p_i p_{uk})$, where $p_i p_{uk}$ is the vectors starting from p_i to p_{uk} of D_u, and defines the neighbor position p_{ref} that gives the minimum $ang(\vec{G}, p_i p_{uk})$ between $\vec{G}$ and $p_i p_{uk}$. Starting from $p_i p_{ref}$, r_i checks whether there exists the neighbor position p_{ul} which belongs to D_u within the area obtained by

rotating $\overrightarrow{p_i p_{ref}}$ 60 degrees clockwise. If there exists p_{ul}, r_i finds another neighbor position p_{um} using the same method starting from $\overrightarrow{p_i p_{um}}$. Unless p_{ul} exists, r_i defines p_{ref} as p_{rn}. Similarly, r_i can decide the neighbor position p_{ln} while rotating 60 degrees counter clockwise from $\overrightarrow{p_i p_{ref}}$. The two positions, denoted as p_{rn} and p_{ln}, are located farthest in the right-hand or left-hand direction of $\overrightarrow{p_i p_{ref}}$, respectively. As illustrated in Fig. 8-(b), a unification area $A(U)$ is defined as the common area between $A(\vec{G})$ in SB and the rest of the area in SB, where no element of D_u exists. Then, r_i defines a set of robots in $A(U)$ and selects the first neighbor r_{s1} located the shortest distance away from p_i in $A(U)$. The second neighbor position is defined such that the total distance from p_{s1} to p_i can be minimized through either p_{rn} or p_{ln}. As a result, p_{ti} can be obtained by $\varphi_{interaction}$ in ALGORITHM-1. Fig. 9 demonstrates how two separate groups of 120 robots merge into one while maintaining the local geometrical configuration.

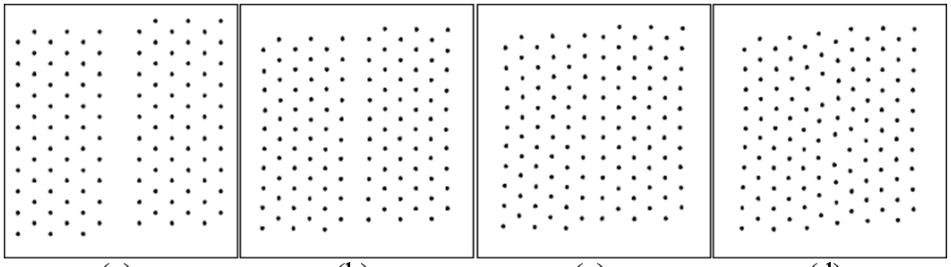


Figure 9. Simulation for unification algorithm ((a) initial distribution, (b) 5 sec. (c) 14 sec. (d) 20 sec.)

5. Adaptive Tracking Algorithm

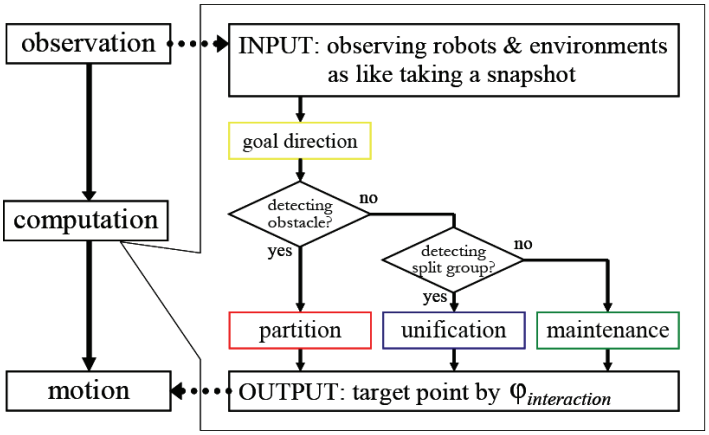
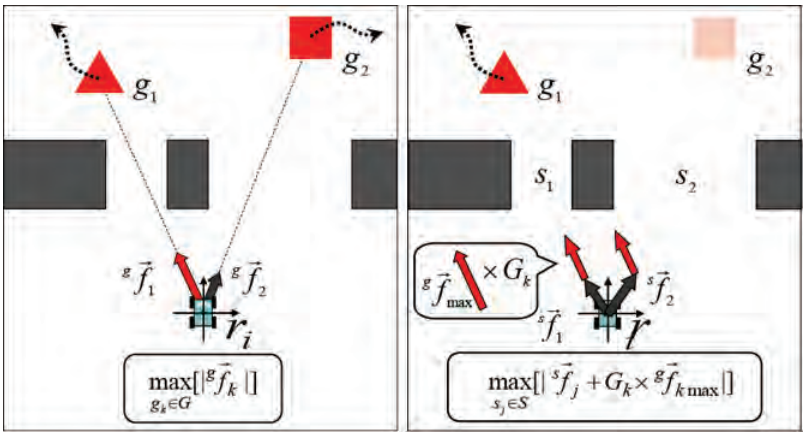


Figure 10. Adaptive tracking flowchart

This section introduces a straightforward extension of adaptive flocking to a more sophisticated example of swarm behavior that enables groups of robots to follow multiple

moving goals while adaptively navigating through an environment populated with obstacles. Fig. 10 shows the flowchart of this adaptive tracking application. Under the same activation cycle as described in Section 4, each robot first identifies the goal(s) in its *SB* and selects a single goal to track. After adjusting the *goal direction*, when the robot finds the geographical constraint within its *SB*, the robot executes the *partition algorithm* to adapt its position to the constraint. If the robot finds no constraint, but observes any robot around the outside of its group, the robot executes the *unification algorithm*. Otherwise, the robot basically executes the *maintenance algorithm* while navigating toward the selected goal. Notice that the adaptive tracking differs from the adaptive flocking in computation of the goal direction detailed below. Specifically, the partition in the tracking is to enable a single swarm to be divided into smaller groups according to an environmental constraint and/or selected goal.



(a) computation of goal favorite vectors (b) compuation of navigation direction

Figure 11. Illustrating direction selection in adaptive tracking

In Fig. 11, similar to Eq. (1), the favorite vector for the passageway is defined as $^s\vec{f}_j$. Likewise, the tracking goal is defined as $^g\vec{f}_k$. Assuming that one of the goals g_k is located some distance d_k away from p_i, the magnitude of the favorite vector $^g\vec{f}_k$ for the goal is given by

$$|^g\vec{f}_k| = |1/d_k^2| . \tag{2}$$

Here, it is assumed that the set of multiple moving goals *GS*, $\{^g\vec{f}_k | 1 \le k \le n\}$, has the same priority across the respective goals. From *GS*, r_i selects a favorite vector with the maximum magnitude denoted as $|^g\vec{f}_k|_{\max}$. As described in the Subsection 4.3 (see Fig. 6-(b)), r_i defines the maximum favorite area $A(^g\vec{f}_{k\max})$ and selects the neighbors within $A(^g\vec{f}_{k\max})$.

Next, let us consider the case that r_i observes both the goals and passageways. As shown in Fig. 11-(a), r_i first defines the favorite vectors of the observed goals $^g\vec{f}_k$, and then selects g_k with $|^g\vec{f}_k|_{\max}$. With respect to the selected goal, as seen in Fig. 11-(b), r_i selects s_j based on the following measure

$$\max_{s_j \in S} \left[\left| {}^{s}\vec{f}_j + G_k \times {}^{g}\vec{f}_{k\max} \right| \right] \tag{3}$$

where G_k indicates a weighting coefficient in order to upset the balance between ${}^{s}\vec{f}_j$ and ${}^{g}\vec{f}_k$. Similar to the previous approach, r_i defines $A({}^{s}\vec{f}_{k\max})$ where the first neighbor is selected.

6. Simulation Results and Discussion

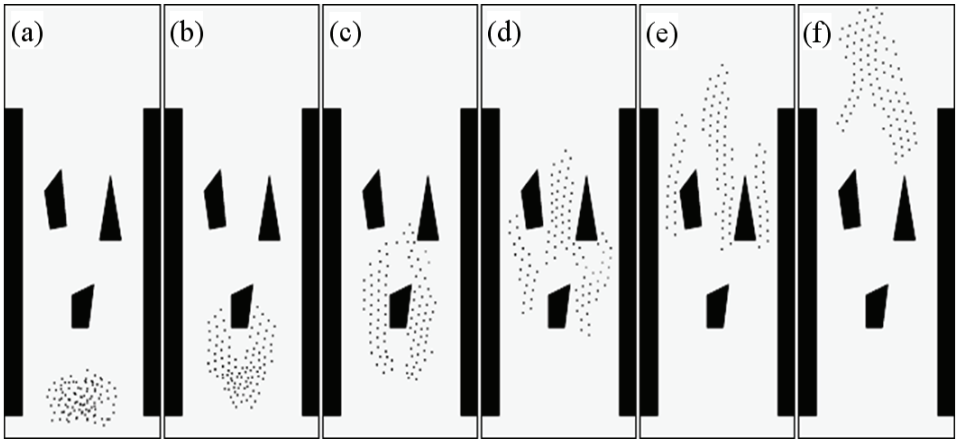


Figure 12. Simulation results of adaptive flocking toward a stationary goal

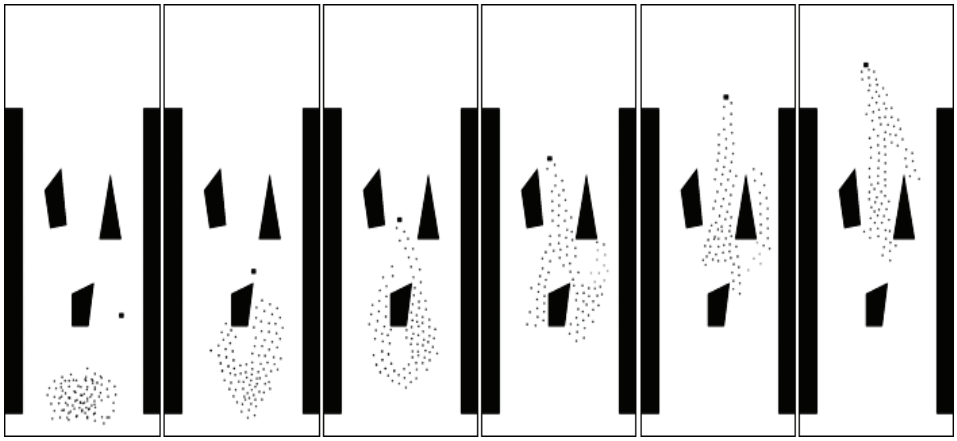


Figure 13. Simulation results of adaptive tracking toward a moving goal

To verify the proposed flocking and tracking algorithms, simulations are performed with a swarm of 100 robots. We set the distance d_v between p_v and p_i to 1.2 times longer than d_u and the range of SB to 3.5 times longer than d_u. Moreover, in the tracking simulations, G_k

was set to 10. The first simulation demonstrates how a swarm of robots adaptively flocks in an unknown environment populated with obstacles. In Fig. 12, the swarm navigates toward a stationary goal located at the upper center point. On the way to the goal, some of the robots detect an obstacle that forces the swarm split into two groups in Fig. 12-(b). The rest of the robots can just follow their neighbors moving ahead toward the goal. After being split into two groups, each group maintains the geometric configuration while navigating in Fig. 12-(c). Note that the robots that could not identify the obstacle just follow the moving direction of preceding robots. Figs. 12-(d) and (e) show that two groups are merged and/or split again into smaller groups due to the next obstacles. In Fig. 12-(f), the robots successfully pass through the environment.

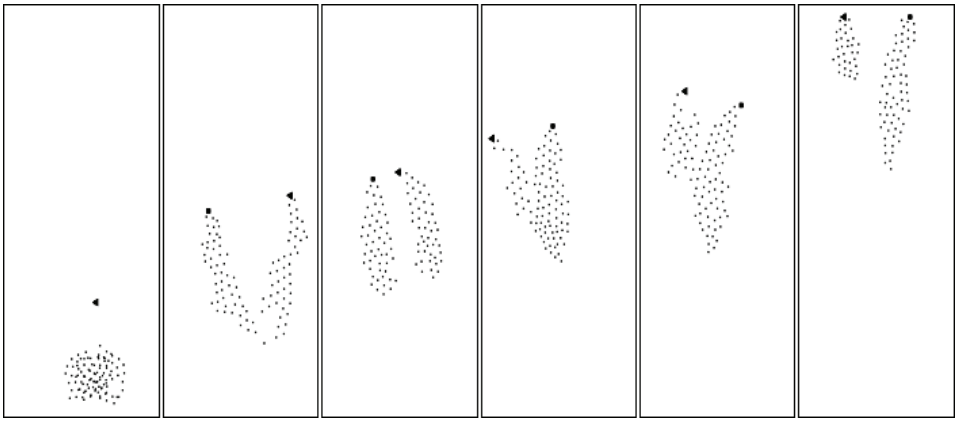


Figure 14. Simulation results of tracking two moving goals in free space

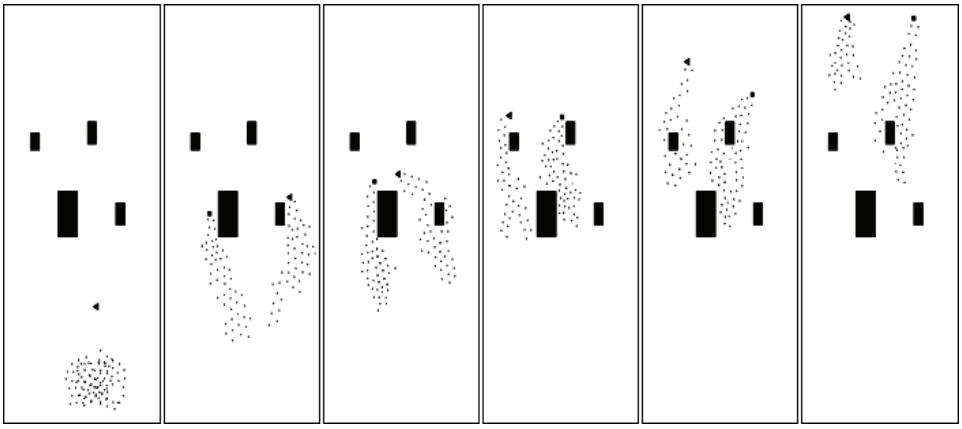


Figure 15. Simulation results of tracking two moving goals in a geographically-constrained environmental constraint

The next simulation results seen in Fig. 13 present the snapshots for tracking of a moving goal represented by the square. As the goal moves, the swarm starts to move. It can be observed that the snapshots of Fig. 13 differ from those of Fig. 12, since ${}^{s}\vec{f}_j$ varies in accordance with $\vec{G}$ detected at each time.

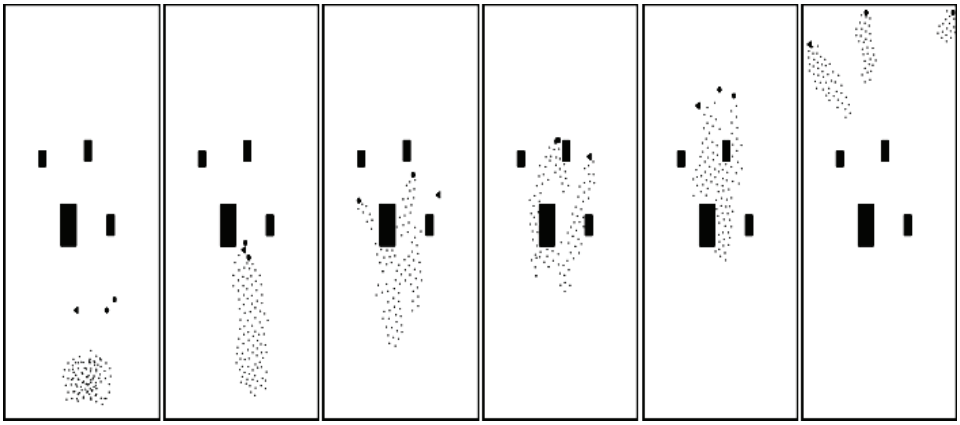


Figure 16. Simulation results of tracking three moving goals in a geographically-constrained environmental constraint

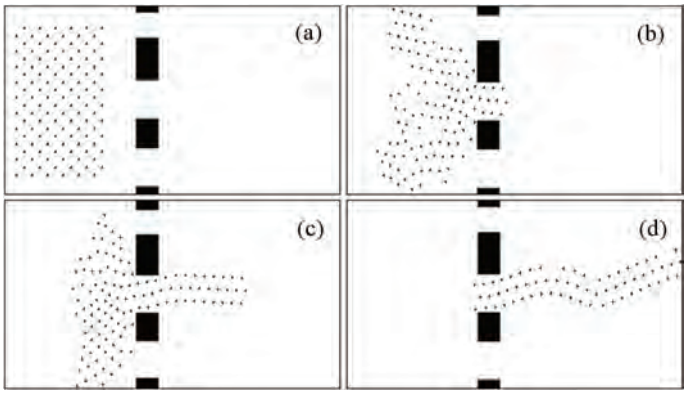


Figure 17. Simulation for flocking without partition capability ((a) initial distribution, (b) 13 sec. (c) 52 sec. (d) 148 sec.)

Figs. 14 and 15 present the snapshots that the same swarm tracks two moving goals having different velocities represented by the square and the triangle, respectively. The simulation conditions are the same, but Fig. 15 is carried out in the environment populated with obstacles. In addition, Fig. 16 shows how the swarm tracks three moving goals in the same environment. It can be observed that the swarm behavior of each case differs as expected.
In Fig. 17, we investigate the swarm behavior when the partition capability is not available. It took about 150 seconds to pass through the passageway. In the simulation result of Fig. 7,

it took about 50 seconds with the same velocity and d_u. From this, it is evident that the partition provides a swarm with an efficient navigation capability in an obstacle-cluttered environment. Likewise, unless the robots have the unification capability, they may separately perform a common task after being divided as presented in Fig. 18. The capability of unification can be used to make performing a certain task easier, which may not be completed by an insufficient number of robots.

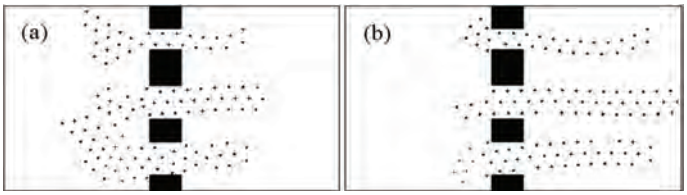


Figure 18. Simulation for flocking without unification capability ((a) 28 sec., (b) 40 sec.)

We believe that our algorithms work well under real world conditions, but several issues remain to be addressed. It would be interesting to verify (1) if the performance of the algorithms is sensitive to measurement errors caused by unreliable sensors, or (2) if the algorithms can be extended to three dimensional space. The algorithms rely on the fact that robots can identify other robots and distinguish them from various objects using, for instance, sonar reading (Lee & Chong, 2006) or infrared sensor reading (Spears *et al*, 2004). This important engineering issue is left for future work. Regarding using explicit direct communications, it also suffers from limited bandwidth, range, and interferences. Moreover, it is necessary for robots to use *a priori* knowledge such as identifiers or global coordinates (Lam & Liu, 2006) (Nembrini *et al*, 2002). We are currently studying the relation between the robot model (or capabilities) and different communication (or interaction) models.

7. Conclusion

In this paper, we presented a decentralized algorithm of adaptive flocking and tracking, enabling a swarm of autonomous mobile robots to navigate toward achieving a mission while adapting to an unknown environment. Through local interactions by observing the position of the neighboring robots, the swarm could maintain a uniform distance between individual robots, and adapt its direction of heading and geometric shape. We verified the effectiveness of the proposed strategy using our in-house simulator. The simulation results clearly demonstrated that the proposed flocking and tracking are a simple and efficient approach to autonomous navigation for robot swarms in a cluttered environment by repeating the process of splitting and merging of groups passing through multiple narrow passageways. In practice, this approach is expected to be used in applications such as odor localization, search-and-rescue, and *ad hoc* mobile networking.

Finally, we emphasize several points that highlight unique features of our approach. First, an equilateral triangle lattice is built with a partially connected mesh topology. Among all the possible types of regular polygons, the equilateral triangle lattices can reduce the computational burden and become less influenced by other robots, due to the limited number of neighbors, and be highly scalable. Secondly, the proposed local interaction is computationally efficient, since each robot utilizes only position information of other robots.

Thirdly, our approach eliminates such major assumptions as robot identifiers, common coordinates, global orientation, and direct communication. More specifically, robots compute the target position without requiring memories of past actions or states, helping cope with transient errors.

8. References

Reynolds, C. W. (1987). Flocks, herds, and schools: a distributed behavioral model, *Computer Graphics*, Vol. 21, No. 4, 25-34

Egerstedt, M. & Hu, X. (2001). Formation constrained multi-agent control, *IEEE Transactions on Robotics and Automation*, Vol.17, No.6, 947-951

Burgard, W.; Moors, M. & Stachniss, C. & Schneider, F. (2005). Coordinated multi-robot exploration, *IEEE Transactions on Robotics and Automation*, Vol.20, No.3, 120-145

Baldassarre, G.; Trianni, V. & Bonani, M. & Mondada, F. & Dorigo, M. & Nolfi S. (2007). Self-organized coordinated motion in groups of physically connected robots, *IEEE Transactions on Systems, Man, and Cybernetics - Part B*. Vol.37, No.1, 224-239

Shimizu, M.; Mori, T. & Ishiguro, A. (2006). A development of a modular robot that enables adaptive reconfiguration, *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, 174-179, IEEE

Folino, G. & Spezzano, G. (2002). An adaptive flocking algorithm for spatial clustering, In: *Parallel Problem Solving from Nature* (*LNCS)*, Vol.2439, 924-933, Springer Berlin, 978-3-540-44139-7

Ogren, P. & Leonard, N. E. (2005). A convergent dynamic window approach to obstacle avoidance, *IEEE Transactions on Robotics and Automation*, Vol.21, No.2, 188-195

Balch, T. & Hybinette, M. (2000). Social potentials for scalable multi-robot formations, *Proc. IEEE International Conference on Robotics and Automation*, 73-80, IEEE

Lee, G. & Chong, N. Y. (2006). Decentralized formation control for a team of anonymous mobile robots, Proc. 6th Asian Control Conference, 971-976, 979-15017-0

Spears, W.; Spears, D. & Hamann, J. & Heil, R. (2004). Distributed, physics-based control of swarms of vehicles, *Autonomous Robots*, Vol.17, No.2-3, 137-162

Spears, D.; Kerr, W. & Spears, W. (2006). Physics-based robot swarms for coverage problems, *International Journal on Intelligent Control and Systems*, Vol.11, No.3, 24-140

Esposito, J. M. & Dunbar, T. W. (2006). Maintaining wireless connectivity constraints for swarms in the presence of obstacles, *Proc. IEEE International Conference on Robotics and Automation*, 946-951, IEEE

Zarzhitsky, D.; Spears, D. & Spears, W. (2005). Distributed robotics approach to chemical plume tracing, *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1-6, IEEE

Sahin, E. (2005). Swarm robotics: from sources of inspiration to domains of application, In: *Swarm Robotics* (*LNCS)*, Vol.3342, 10-20, Springer Berlin, 978-3-540-24296-3

Lam, M. & Liu, Y. (2006). ISOGIRD: an efficient algorithm for coverage enhancement in mobile sensor networks, *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1458-1463, IEEE

Nembrini, J.; Winfield, A. & Melhuish, C. (2002). Minimalist coherent swarming of wireless networked autonomous mobile robots, Proc. 7th International Conference on Simulation of Adaptive Behavior, 373-382, IEEE

Suzuki, I. & Yamashita, M. (1999). Distributed anonymous mobile robot: formation of geometric patterns, *SIAM Journal of Computing*, Vol.28, No.4, 1347-1363

Dolev, S. (2000). Self-Stabilization, *MIT Press*, 978-0262041782

Schneider, M. (1993). Self-stabilization, *ACM Computing Survey*, Vol.25, No.1, 45-67

Wilson, E. O. (2000). *Sociobiology: the new synthesis*, Belknap Press, 978-067400235-7

Stocker, S. (1999). Models for tuna school formation, *Mathematical Biosciences*, Vol.156, 167-190

Halliday, D.; Resnick, R. & Walker, J. (2007). *Fundamentals of physics*, John Wiley & Sons Inc, 8th ed., 978-0471758013

5

Advances in Sea Coverage Methods Using Autonomous Underwater Vehicles (AUVs)

Yeun-Soo Jung, Kong-Woo Lee and Beom-Hee Lee
Seoul National University
Korea

1. Introduction

Due to rapid changes in modern technological development over the last several years, researches on small Autonomous Underwater Vehicles (AUVs) and Unmanned Underwater Vehicles (UUVs) appeared as important issues for various possibilities of application in the ocean. Military applications together with commercial need require practical details which are robust and cheap for realization as a product. In addition, works on coverage have been motivated by a wide range of real world applications that include non-humanitarian demining, deep-sea development sweeping and robotic spray-painting. Currently, many coverage applications utilize on-line coverage algorithms (Gabriely et al., 2003), where robots do not rely on a priori knowledge of a work-area, and thus must construct their motion trajectories step-by-step, addressing discovered obstacles as they move. This is the main contrast from the conventional off-line coverage algorithms, where robots are given a map of a work-area, and can therefore plan their paths ahead of deployment (Ge et al., 2005). In this paper, we focus on on-line coverage for underwater environment by multiple AUVs. Relevant works have shown that one of advantages of adopting multiple robots for a coverage task is the potential ability for more efficient coverage (Hazon et al, 2005). However, another advantage is that they usually offer greater robustness. Unfortunately,this important capability has been neglected in previous works done on on-line algorithms.

As far as the details of the on-line coverage methods, there are many kinds of covering motions that can be realized either by back-and-forth motions in vast cells, or by following a general Voronoi diagram in narrow cells (Acar et al., 2001). Exact cellular decomposition can also be achieved through the use of the boustrophedon decomposition (Choset, 2001) as well as through Morse functions (Acar, 2002). The boustrophedon approach has been extended in (Choset, 2001) to the multi-robot domain. Such form of coverage requires the coverage to be executed in formations, which may be accomplished in a variety of ways (Ge et al., 2005). Spanning Tree Coverage algorithms (Gabriely, 2003) have also been proposed for online coverage of an unknown and gridded environment by a robot.

For coverage with multi-robot teams, a frontier-based exploration technique has been used to guide robots towards the region between known and unknown areas on the map (Yamauchi et al., 1998). The Mark and Cover (MAC) algorithm has been proposed in (Wagner et al., 2000) with the proved convergence and the bounds on cover time of the algorithm. In (Yang & Luo, 2004), a neural network model has been used for exploration of a

Cartesian workspace by a robot, while avoiding local minima problems. The algorithm presented in (Wagner et al., 1999) allows robots to travel along the edge of the area to cover it, and then the robot clears only selected points visited to maintain the connectedness of the area. A market economy approach has been proposed in (Zlot et al., 2006), where coverage completeness has not been guaranteed.

While most of papers focus on the completeness and coverage time, the completeness with no missing area has not been properly addressed in undersea coverage. The completeness depends heavily on the sea current disturbance, and thus generates enormous impact on the performance of coverage algorithm. In this regard, the main structure of our approach is proposed as follows: First, we propose a new and efficient decentralized coverage method using single AUV. Second, we investigated the bound on the coverage time and the upper bound on the total traveling path length for complete coverage. Third, we propose an algorithm considering dynamic environment, i.e., hydrodynamics of AUV and modeling of sea current disturbances. This provides a practically potential possibility in the naval warfare such as Q-route survey, mine hunting, and minesweeping.

This paper is organized as follows: Section 2 briefly reviews the previous works closely related to our approach. In Section 3, we describe the problem in detail with definitions and solution approaches. Furthermore, we illustrate a modeling for AUV with hydrodynamics with sea current disturbances. Section 4 propose an efficient single AUV coverage method compared to the planar-based coverage algorithms. Section 5 proposes a new Multi-AUV coverage method with the mathematical analysis to compare the total traveling path length and coverage time for AUVs. Simulation results are presented in Section 7 for cases of a single AUV and Multi-AUVs, respectively. We finally conclude in Section 8.

2. Existing coverage technique

As well known, coverage technique is considered as an important central issue in utilizing the AUV for exploration in underwater terrain. Recently, several coverage algorithms have been suggested for on-line applications. On-line algorithms are usually needed due to the lack of a priori knowledge of the work-area, while the AUV must construct motion trajectories step–by-step, addressing detected obstacles as they move.

We will briefly introduce relevant works that are most closely related to our topic, i.e., Multi-AUV coverage technique for sweeping and surveillance in the dynamic underwater environment. Fundamentally, our concern is the coverage problem, which is one of canonical problems in robot motion planning. From a coverage algorithm's point of view, the main objective of this research is to investigate efficient methods for covering the littoral regions including mine reconnaissance, mapping, surveillance, and clearance. Among many various and different approaches to solving the coverage problem by robots, the centralized approach is known not to be robust enough especially when communication is limited between an operator and individual robots, and failures frequently occur. Thus, we rather focus on distributed on-line coverage methods in this work, and will review literature about this aspect. A large number of methods for solving the basic motion-planning problem have been reported (Latomb, 1991). The methods are based on a few different general approaches: road map, cell decomposition, and potential field (Latomb, 1991). The global path planning approach addressed in this paper is based on the cell decomposition method and particularly on the semi-approximate method (Hert et al., 1996). As stated in (Choset,2001),a cell decomposition breaks down the target region into cells such that coverage in each cell is

"simple" enough to make it easy to compute a path between any two robot configurations in the cell. Complete coverage is attained by ensuring the robot visits each cell in the decomposition. In this paper, we will look at three types of the decompositions: exact, approximate, and semi-approximate. The following surveys have been classified as core studies in understanding major approaches to coverage technique in robotics including Multi-AUVs in oceanography.

2.1 Exact cell

For a one-body mobile robot moving in the plane,where the environment is composed of polygons, an exact cell decomposition that results in a connectivity graph is a well-known planning approach (Latomb,1991). The free space of the robot is exactly partitioned into cells that are stored in a graph. The on-board global planner computes a path for the robot from this graph. Choset (Choset, 2001) is stated as Multi-AUV coverage technique using a exact cell decomposition. Kurabayashi et al.(Kurabayashi et al., 1996) suggest an off-line multi-robot coverage strategy using a Voronoi diagram-like and boustrophedon approach. They define a cost function to pseudo-optimize the collective coverage task. Rekleitis et al.(Rekleitis et al.,1997) use a visibility graph like decomposition of space to enable coverage with multiple robots. Here, the goal is to use the robots as beacons for each other to eliminate dead-reckoning error.

Butler et al.(Butler et al., 1999) develop one of a cooperative sensor-based coverage algorithm.The distributed coverage of retilinear environment (*DC*R) operates independently on each robot in a team. It applies to rectangular robots that use only contact sensing to detect obstacles and operate in a shared, connected rectilinear environment. The basic concept of *DC*R is that cooperation and coverage are algorithmically decoupled. This means that a coverage algorithm for a single robot can be used in a cooperative setting, and the proof of completeness is much easier to obtain.

*DC*R (Butler et al., 1999) is based on a complete single-robot coverage algorithm,i.e., contact sensor-based coverage of rectilinear environments (*CC*RM) which incrementally constructs a cellular decomposition of the environment (*C*). To produce cooperative coverage, *CC*RM is enhanced with two additional components. Of these, the overseer is the more important. Its job is to take incoming data from other robots and integrate it into *C*, which it must do in such a manner that *C* remains admissible to *CC*RM. The cell is first shrunk to avoid overlap with existing cells, then added to *C*. Incomplete cells in *C* are reduced to avoid overlap with the new cell, and all connections between cells are updated to reflect this addition. It can be shown that the overseer of *DC*R indeed performs this operation in such a way that coverage can continue under the direction of *CC*RM without *CC*RM even knowing that cooperation occurred.

2.2 Approximate cell

In this approach, the environment is divided into a fine grid where each cell in the grid contains a flag identifying it as free space or not.The resolution of the grid must be very high in order to capture every important detail, resulting in a graph with very many nodes.Thus path planning is not very efficient and the decomposition does not give rise to a natural way of describing the environment, but it is easy to implement. As stated in (Choset, 2001), Wagner et al.(Wagner et al., 1996) investigate the ability of multiple robots, that communicate by leaving traces, to perform the task of cleaning the floor of an unmapped

building. More specifically, they consider robots that leave chemical odor traces that evaporate with time, and evaluate the strength of smell at every point they reach, with some measurement error. Wagner et al.(Wagner et al., 1997) analyze the problem of many simple robots cooperating to clean the dirty floor of a nonconvex region represented by an approximate cellular decomposition, i.e., a grid, using the dirt on the floor as the main means of inter-robot communication. Their algorithm guarantees task completion by k robots and they prove an upper bound on the time complexity of this algorithm. Again, robots communicate only through traces left on the common ground.
Borrowing ideas from computer graphics Wagner et al.(Wagner et al.,1996) preserve the connectivity of the dirty region by allowing an agent to clean only a so called noncritical point, that is: a point that does not disconnect the graph of dirty grid points. This guarantees that the robots will only stop upon completing their mission. An important advantage of this approach, in addition to the simplicity of the agents, is its fault-tolerance: even if almost all the agents cease to work before completion, the remaining ones will eventually complete the coverage.

2.3 Semi-approximate cell

Hert et al. (Hert et al., 1996) present an on-line terrain covering algorithm that relies on a partial unknown of space where the width of cells are fixed, but the top and bottom can have any shape. Their planar terrain-covering algorithm can be applied to both simply and non-simply connected environment. The simplicity of their algorithms lies in its recursive nature. An AUV utilizing this algorithm may start at an arbitrary point in the environment and will zigzag along parallel straight lines to cover given area. Portions of the area that either would not be covered or would be covered twice using the zigzag procedures are detected by the AUV and covered using the same procedure. These smaller areas are covered as soon as they are detected and inlets within inlets are treated in the same way. The recursive zigzagging procedure causes the inlets to be covered in a depth-first order. The algorithm requires that the AUV remember the points at which it enters and exits every inlet it covers. This assures that each inlet is covered only once. The algorithm has been designed with an emphasis on efficiency. That is, it guarantees complete coverage for the entire area without duplication of the AUV path and coverage area in the process of exploration. In subsequent discussions, we will present a new coverage method for both single and multi-AUV to efficiently cover the dynamic shallow water.

3. Problem statement

In this section, we describe topic, Multi-AUV coverage method for unknown environment considering hydrodynamics of AUV and outside disturbances. The purpose of this study is to develop an efficient method of complete 3-dimensional coverage of unknown oceanic environment without missing areas for AUVs. Missing areas in AUV exploration usually occur when AUV leaves off the reference path affected by complex internal and external factors. Internal factors include noise from the controller, sensors, and the cycle of control whereas external factors include ocean current, waves, wind and the temperature and the density of seawater. The white segments in the Fig.1 represent missing areas from the exploration by an AUV.

This section describes how this study was organized and performed. Specifically, this study uses AUV models that are actually in use instead of concentrating on theoretic research and, therefore, practicality is more emphasized in this study as it is based on the elements of reality such as AUV hydrodynamics and external factors such as sea current disturbances.

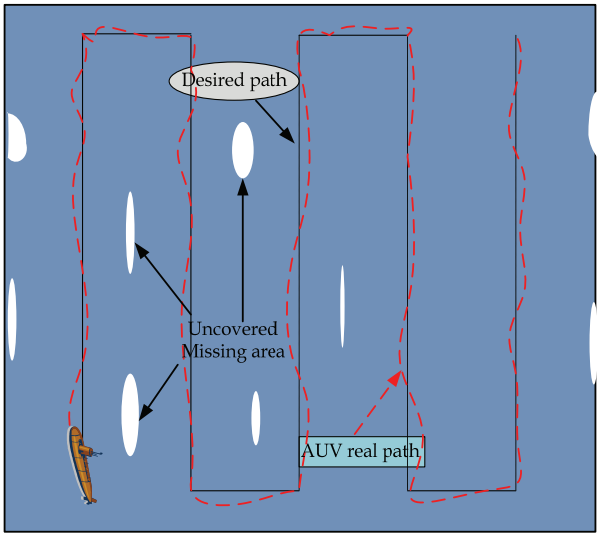


Figure 1. AUV missing areas caused by complex disturbances

3.1 AUV hydrodynamics

As AUV can navigate freely in 3-dimension space of underwater, its 6-degree of freedom motion has continuity and the equation of its motion reveals expansion on a major non-linear term, which should be considered according to the mode of motion. In this paper, the torpedo-type AUV was selected and the standard equation of motion for AUV (Fossen, 1994) was used. REMUS developed by WHOI (Prestero, 2001) was assumed the AUV model selected as an object of control for this study. The system of AUV movements could be simplified by breaking up its motion into vertical and horizontal factors. Fig. 2 shows coordinate frames that can be used to represent AUV motions.

$$\begin{aligned} M(v)\dot{v} + C_D(v)v + g(\eta) + d &= \tau \\ \dot{\eta} &= J(\eta)v \end{aligned} \tag{1}$$

From Equation (1), the position and the orientation of the AUV are represented in earth-fixed frame by $\eta = [x, y, z, \phi, \theta, \psi]^T$ and the velocity of pitch, heave and surge and the angular velocity are represented in body-fixed frame by $v = [u, v, w, p, q, r]^T$. The transformation between the two coordinate frames can be done using Jacobian matrix. In the Equation (1), M is an inertial mass matrix, C_D is the matrix of Coriolis and centripetal caused by the rotation of the earth, $g(\eta)$ is the dynamic stability vector, which is the sum of buoyancy and gravity, d is the external disturbances, τ is the vector for the propulsive force of the AUV and $J(\eta)$ is the conversion matrix. The 6-degree of freedom equations can be deduced from the

above Equation (1). The reference to the detail process (Fossen, 1994) is made from the Equation (1). However, it is necessary to simplify the equation that signifies only the movements in the directions of *u-axis* and *v-axis* and the rotation on *w-axis*, as the underwater exploration by AUV is performed at constant depth. When developing AUV controls, the motion in the horizon plane is separated from that in the vertical plane. Although control designs only deal with two-dimensional planes, three–dimensional vehicle control can be achieved simply by running the horizontal and vertical control algorithms simultaneously.

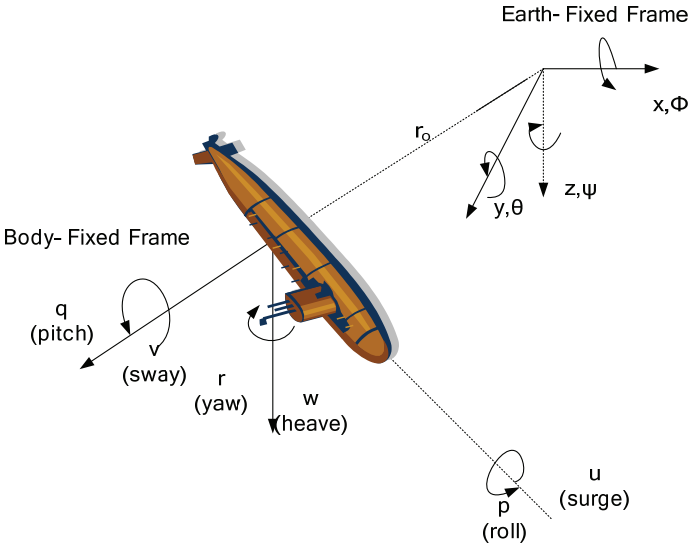


Figure 2. Earth-fixed frame and body-fixed frame for AUV

Assuming that the velocity of AUV motion in u direction results in constant forward surge at U_0 and neutralized buoyancy, the equation of horizontal motion in earth-fixed frame can be expressed as in Equation(2). The REMUS hydrodynamic coefficients for equations of motion in the horizontal plane obtained from research found in the (Prestero, 2001).

$$\begin{aligned} \dot{x} &= U_0 \cos\psi + v \sin\psi \\ \dot{y} &= U_0 \sin\psi - v \cos\psi \\ \dot{\psi} &= r \end{aligned} \tag{2}$$

3.2 AUV controller model

An AUV move with speed in the dynamic environment it need efficient internal control system. As stated in (Yan & Robot, 2007), AUV guidance systems as follows: waypoint guidance by line of sight (LOS), vision-based guidance, and Lyapunov-based guidance. Among these guidance laws, LOS is one of the most widely used guidance strategies for AUVs due to its ease of implementation (Naeem et al., 2003). It has been applied to some single AUV navigation missions (Belkhouche et al., 2006) .The LOS guidance is based on the geometry of the interception scenario. The system can be described in a relative system of coordinates as shown in Fig.3, which shows the AUV and the target. The positions for target

and AUV in the reference frame of coordinates are given by the vectors (x_t,y_t) and (x_K,y_K). The notation in Fig .3 show that LOS_{GK} is the LOS of AUV-Target and Θ_{GK} is the line of sight angle. In dynamic, the LOS angel to the target is defined as follow :

$$\theta_{GK} = \arctan\left(\frac{y_t - y_k}{x_t - x_k}\right) \tag{3}$$

The steering angle factor β is equal to Θ_{GK}:

$$\beta = \theta_{GK} \tag{4}$$

Therefore, LOS controller was used to steer the AUV along the reference path. The choice of LOS controller was because it maintains stability well by using the geometric relation between current position of the AUV and its reference path of travel when a large directional error occurs (Healey et al., 1993). Also, PD Controller (Prestero, 2001) was used to control the AUV as it allows easy observation of movements.

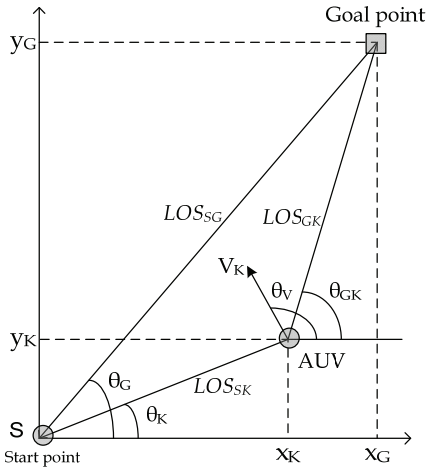


Figure 3. AUV LOS guidance controller in a static environment (without disturbances)

Performing a control input to Equation (2), which is the linearization of the horizontal element of the equation of AUV motion, the movements of AUV could be expressed by an equation of state, where the state parameters are velocity on y-axis (v), angular velocity on z-axis (r) and the angle of z-axis (ψ) in earth-fixed frame and the angle of horizontal steering wheel with control input $\delta(R)$.

3.3 AUV sensor model

AUV sensor model is based on Hert's algorithm (Hert et al., 1996). The AUV is a point moving in three dimensions. A fixed orthogonal coordinate system X =(x,y,z) is chosen with its origin at the AUV's starting point S and z axis passing through the earth's center. The AUV is equipped with sensors. The sensor allows the AUV to determine its own

coordinates relatives to X and those of any point detected in its sensing region. The sensing region is a rectangular polyhedron of dimension l x w x h (*length* x *width* x *height*), with the AUV at its center. These dimensions are chosen such that, from a vertical distance of $h/2$ from a horizontal plane, the AUV's sensor will take a sensing of a rectangle on the plane of dimensions l x w. The value $h/2$ is determined by the focal length of the AUV's sensors. The sensor also allows the AUV to determine the slope of the floor within its sensing region, which enables it to maintain the vertical distance of $h/2$ from the floor.

The sensed area is the portion of the ocean floor that the AUV can sense from a particular position R. This area is, in general, different from the intersection of the ocean floor with the sensing region. Rather, it consists of all points p on the floor that are in the sensing region and for which a line segment $\overline{pR}$ does not intersect the surface at any other point as shown in Fig.4.

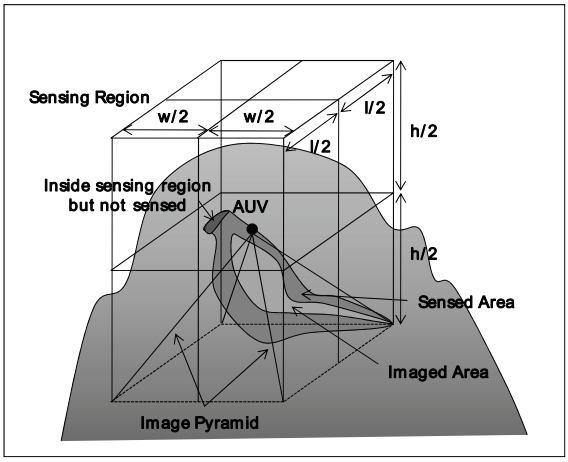
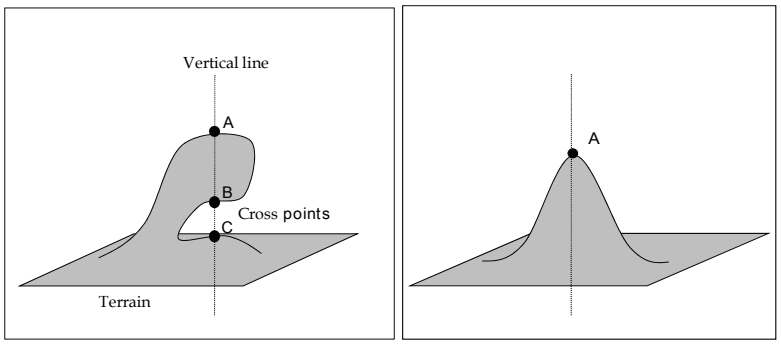


Figure 4. Illustrated AUV sensing model in underwater environment

The *imaged area* is the portion of the ocean floor of which the AUV takes a sensing from a particular position R. This will be some subset of the sensed area. In particular, it is the portion of the sensed area that lies within the *image pyramid*. This pyramid has its top at R in the center of the sensing region and base equal to the bottom of the sensing region (a rectangle of dimension l x w).

3.4 Underwater environment model

Although the topography of the actual ocean floor in exploration areas consists of complex structures, this paper assumed existence of vertically projectable surfaces only as shown in Fig. 5(b). The vertically projectable surface means the topographic surface on which only one cross point by a vertical line exist at any position. The structure in Fig. 5(a), for instance, is the typical example of vertically non-projectable complex surface on which a vertical line can have 3 cross points.



(a) (b)

Figure 5. Side view of environment model with cross points.

The reason for assuming the entire topography to be a vertically projectable surface was that the AUVs are to perform exploration maintaining the constant depth. As all the surface structures should be identified by viewing from the same height, the assumption that the surface topography is vertically projectable was necessary, which also allows 2-dimensional, instead of 3-dimensional, as shown in Fig.6, approach to the mapping of complex ocean geography.

We assume that area A in Fig.6 is bounded between two threshold surfaces, as shown in Fig.6(a), $z = z_{min}$ and $z = z_{max}$. Portions of the floor below z_{min} or above z_{max} are not to be covered as shown in Fig.6(a). The area is also bounded by a threshold slope, μ. Proportional to the image width w and is chosen to assure adequate overlap of the images. The range of possible values for μ is limited by the AUV's sensor parameters. In particular, if the focal range of AUV's sensor is $[a,b]$, then μ must be chosen to satisfy the relation:

$$\mu < \min\left(\frac{h}{3w}, \frac{2(b-a)}{w}\right) \tag{5}$$

This assures that images of adjacent portions of the surface will overlap to some extent and every part of the surface that is to be imaged lies within the sensor's focal range as shown in Fig. 6(b). The boundary B of the area A consists of a number of simple, nonintersecting closed curves. These are intersections of the floor with the threshold surfaces and the curves along the floor where the slope is equal to μ as shown in Fig.6(b). The *planar area* A_p and *planar boundary* B_p are defined as the projections of the area A and boundary B on the *xy*-plane (i.e, plane z = o). The outer boundary is the boundary curve that contains all other boundary curves in its interior as shown in Fig.6(c).

In order for AUV to be able to recognize the topography of ocean floor viewing it as a vertically projectable surface, the oceanic topography are divided into 4 types, Cape, Bay, Inlet and Island as shown in Fig. 7. Their definitions are as follows:

- **Cape**: Cape is a convex structure as the area between point C1 and C2.
- **Bay:** Bay is a concave structure as the area between point B1 and B2.
- **Inlet**: Inlet is an area that has a cape point as its entrance and bay point as its end as shown in Fig. 7(b)
- **Island**: Island is an area isolated from the surrounding topographic structure.

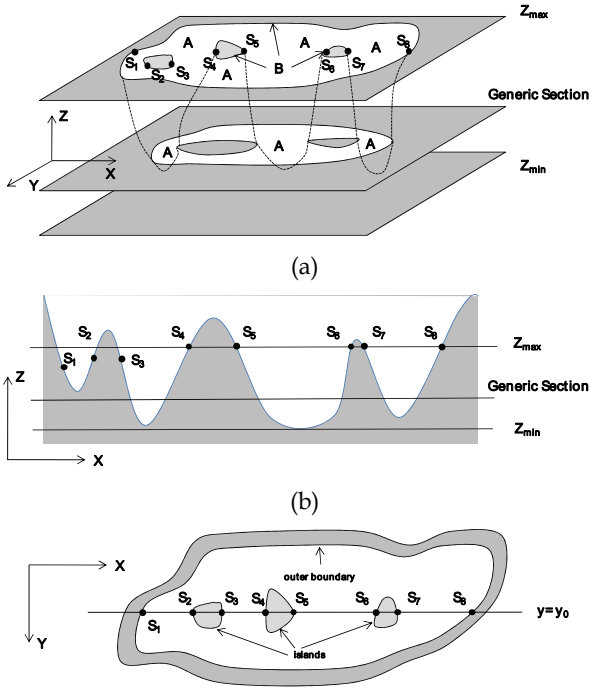


Figure 6. The procedure for converting 3-D underwater environment to 2-D terrain

Cape and Bay are considered here as the characteristic points. The classification used in this paper is based on Hert's method of classifying the topography. Fig. 7(a) shows the characteristic points of Cape and Bay. The vertical dashed lines in Fig. 7 represents the travel path of zigzagging AUV and the points B1, B2, C1 and C2 that meet with the dashed lines the characteristic points which allow recognition of certain oceanic topography when the AUV comes across them.

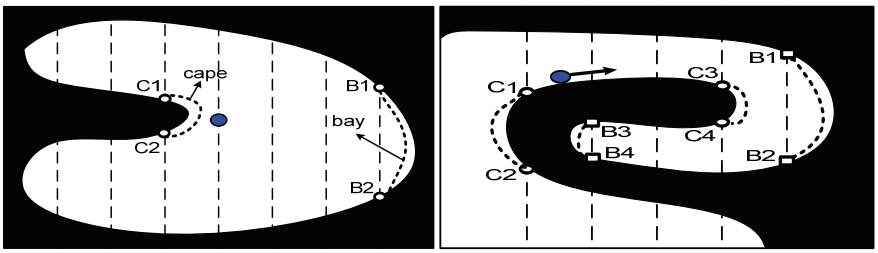


Figure 7. Types of sea environment

When the AUV encounters the cape point C1, it knows that there is a space ahead for further exploration and when it crosses the bay points B1 and B2, it recognizes that the exploration of the space entered from the point C1 has been completed. However, Fig. 7(b) shows the

recursive structure of inlets where another inlet can be formed inside the inlet from the point C1. That is, an inlet can have another inlet in it. Such recursive characteristic of inlets provides answers to the question as to how to approach to the exploration of inlets. These answers will be discussed in the next section. The AUV that enters the inlet passing the point C1 keeps on moving until it reaches another inlet as it passes the points C3 and C4. It will know that the exploration of the inlet beginning from the point C1 has been finally completed when it encounters the point B1 and B2. The AUV exploring an island will not be able to correctly identify the island structure until it explores the area repeatedly as it enters the unknown environment when it enters an inlet. Therefore, the efficiency of topographic exploration of ocean environment lies very much in how the exploration of islands is dealt with. The Section 4 will discuss about the method of efficient coverage in relation to this issue.

3.5 External disturbance model

This paper attempted to reflect the effects of underwater external sea current disturbances, which is considered as the external factors of cross track error by using the model of underwater external disturbances. The shallow coastal water was chosen as the subject area. The major effects of oceanic current in coastal areas come from tidal current and stoke's drift. Considering these elements as well as the minimum and maximum currents and their fluctuations, a model was designed to determine the type of changes in the velocity of oceanic current using the First Gauss-Markov process (Fossen, 1994).

$$\dot{V}_C(t) + \mu_0 V_C(t) = w(t) \tag{6}$$

In the above Equation (6), $w(t)$ represents white noise with average value of '0' and μ_0 is a non-negative constant. As the AUV movements are horizontal, the velocity of oceanic current can be described 2-dimentionally in the body-fixed frame by the direction (β_C) and the velocity (V_C) as follows:

$$\begin{aligned} u_C &= V_C \cos\beta_C \\ v_C &= V_C \sin\beta_C \end{aligned} \tag{7}$$

where u_c is the velocity of oceanic current in x direction and v_c is the velocity of oceanic current in y direction in body-fixed frame coordinate. In order to apply this oceanic current model to the equation of AUV motion represented in body-fixed frame, the effect of the body posture is reflected also in the body-fixed frame as shown in the following Equation (8).

$$\begin{aligned} u_C &= V_C \cos(\beta_C - \psi) \\ v_C &= V_C \sin(\beta_C - \psi) \end{aligned} \tag{8}$$

The actual values of the changes in the velocity of ocean current and its direction would be applied to the implementation environment of the test in reality. The oceanic current represented in the body-fixed frame will have the effect on the movement of AUV as it is interpreted as a relative velocity and the final model will be complete reflecting the movement of AUV and the oceanic current as an external disturbance.

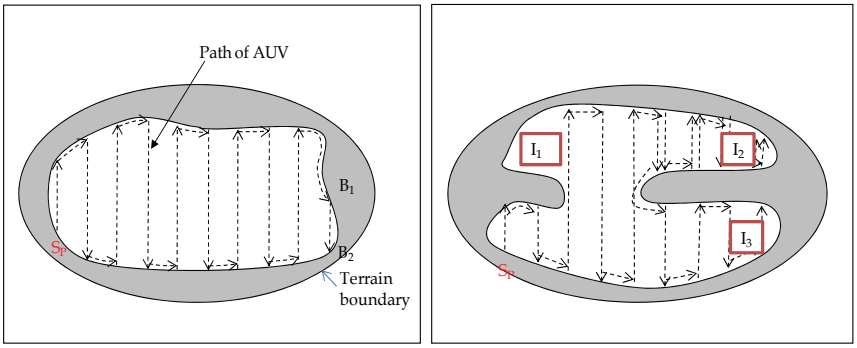
$$\begin{bmatrix} \dot{v} \\ \dot{r} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & 0 \\ a_{21} & a_{22} & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} v \\ r \\ \psi \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ 0 \end{bmatrix} \delta + \begin{bmatrix} -a_{11}\sin(\beta_C - \psi) \\ -a_{21}\sin(\beta_C - \psi) \\ 0 \end{bmatrix} V_C \tag{9}$$

The above Equation (9) is the matrix version of the same equation expressed by arranging in terms of magnitude of change in velocity $\dot{v}, \dot{r},$ and $\ddot{\psi}$, respectively. The δ is the value of the angle of the rudder obtained from the PD controller and LOS controller and b_1 and b_2 are the control constants for the rudder. The values a_{11} through a_{22} are the constants of the matrix obtained as the result of rearranging the equation in terms of $\dot{v}, \dot{r},$ and $\ddot{\psi}$, respectively.

4. An efficient coverage method with unknown underwater terrain

In this section, coverage methods of exploring unknown shallow water floor are investigated. First, we researched one of existing efficient covering Hert's algorithm briefly and examined how proposed coverage method is improved compared with Hert's. Then, by applying proposed coverage method, Multi-AUV coordinated coverage with *Role Changing*, which is effective method for unknown terrain, is proposed.

As stated in (Hert & Lumelsky, 1996), key to on-line terrain-covering algorithm is a simple zigzagging pattern of motion in which the AUV moves back and forth along successive grid lines, sweeping across an area either from left to right or from right to left. Fig. 8(a) shows the path of a AUV moving in this fashion in a simply connected area, the boundary of which contains only two bays and no capes. By starting at one of the bay points on the boundary S_p and zigzagging until it encounters the other bay, the AUV is able to cover the entire area.



(a) (b)

Figure 8. AUV zigzag manner of unknown area for covering procedure

However, as Fig. 8(b) illustrates, this simple zigzagging motion will not suffice to cover areas with boundaries that contain capes, nor will it work when the area is not simply connected or when the starting point is chosen arbitrarily. The inlet labeled I_1 in Fig. 8(b) is never encountered by the AUV as it zigzags along the grid lines, and thus remains

uncovered. In contrast, the AUV must resurvey its path in inlet I_2 in order to reach the inlet I_3. Inlets such as I_1 and I_2 are referred to as diversion inlets, or simply inlets, since they require that the AUV divert from its normal zigzagging motion in order to cover them efficiently.

However, this backtracking is both undesirable and unnecessary. Since the AUV can know each inlet as it is moving, it can immediately make a diversion from its path to cover this inlet and then return to the point at which the diversion was made, and continue on its covering way. In this way, the entire area will be covered without any extensive backtracking. This is the strategy of the method presented here. Further, the procedures presented assure that every inlet is covered only once.

In addition, the proposed method of coverage was the result of the consideration of all the variable factors that influence the actual navigation of the AUVs. Therefore, as Hert assumed the width of exploration path applied during zigzag navigation to be ideal, this study also used his concept as the basis and concentrated on the method of exploration. However, the question of how do we determine the ideal width of exploration path considering real situation should also be a very important point of consideration in this study. How can we find the most useful exploration width? We will try to find the answer from the cross track error (CTE) that arises during the exploration with AUV.

It is generally possible to assume that AUV can perform exploration without creating missing areas by setting the exploration intervals without external disturbances to twice the range of the sensor. However, the CTE in actual exploration still creates missing areas due to imprecise exploration width as shown in Fig. 1.

It means that the exploration width should be reduced from twice the sensing range to the marginal width that would not cause missing areas after all. However, as the velocity of the oceanic current in the exploration area is variable by time although its flow is consistent, it is not easy to find the marginal exploration width that would allow the AUV to navigate shortest distance without causing any missing areas. Therefore, we would suggest to efficient exploration width (EEW), if not marginal width, that would allow the AUV to navigate shortest distance without causing any missing areas. Hert has suggested an exploration method without considering EEW.

4.1 Conventional covering method of Hert

We will look into the method of the exploration of oceanic topography. The conventional method of Hert for the exploration of a given geographic area is based on continuous exploration following the exploration path using basically online-based zigzag navigation. The advantage of Hert method is obvious compared with previous exploration methods, which are based on the assumption of simple oceanic environment. Hert, on the other hand, assumes complex unknown area instead of known square-cut areas as an object of exploration. This approach has an advantage, apart from the fact that it shortens the path length, in that it is more realistic because, after finishing the exploration of a given area, many exceptional situations are considered also to ensure no missing areas arise.

The proposed coverage method in this paper also adopts this advantage of Hert method and is the result of further improvement of Hert algorithm, which we introduce here. Hert's method is based on largely dividing the oceanic environment into two categories: one with islands and the other without.

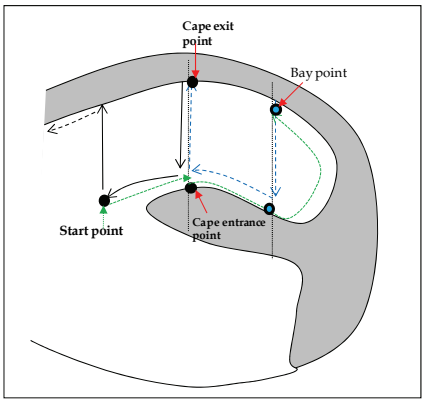
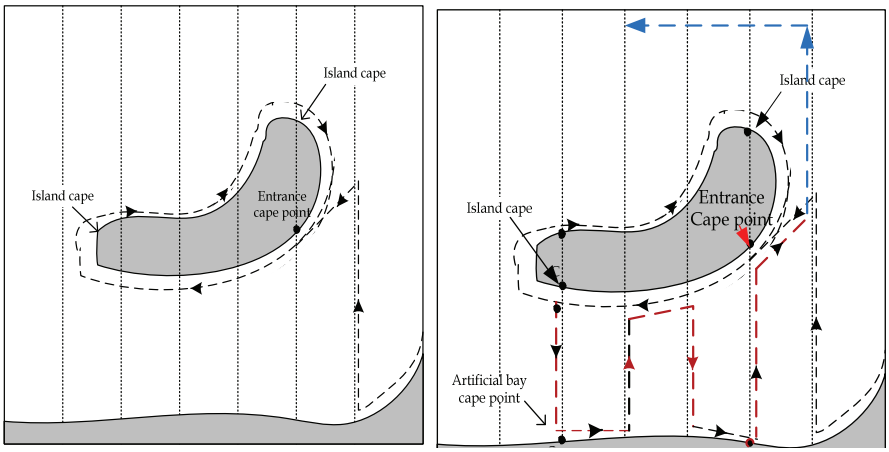


Figure 9. Hert's algorithm for simply connected area (inlet)

In the oceanic environment without islands as shown in Fig.9, AUV navigates in zigzag pattern until it encounters an inlet. Upon crossing the cape point, which is the entrance of an inlet, it navigates along the boundary of the inlet until it comes across with a bay point, which indicates that it reached the end of the inlet. Then the AUV moves to the other bay point and resumes zigzag navigation exploring to see if there is any other structure within the inlet until it exits from the inlet. Furthermore, it will store the information on the cape points in order to prevent from exploring the same area again later, which he termed as 'crossing the lock'.



(a) (b)

Figure10. Hert's algorithm for non-simply connected area (island)

In the oceanic environment with islands as shown in Fig. 10, exploration is performed in the same way as in the case when there is no island. An island characteristically has 4 cape points without any other points. As it is initially an unknown environment, as shown in Fig.10(a),the AUV starts navigating along the boundary of the island as it encounters a cape

point. The AUV will eventually return to the original position and recognize that it is an island. Then the AUV navigates along the boundary from the entry point to the opposite cape point and returns in zigzag pattern exploring the topography of the one side of the island as shown in Fig. 10(b). The AUV then travels to the other side of the island and navigates toward the original point in zigzag pattern exploring the topography of that side of the island as shown in Fig. 10(b).

4.2 Calculating an efficient exploring width (EEW)

We believe that Efficient Exploring Width (EEW) can be found from the model of AUV movement using CTE values.

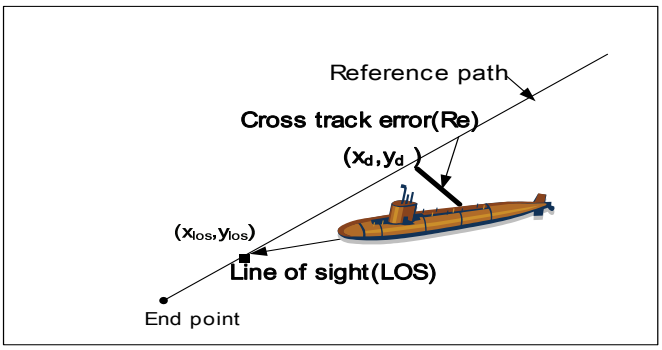


Figure 11. Illustrated cross track error by AUV

The value of CTE is, as shown in Fig. 11, a vertical distance of AUV from the reference path. The reference path is first determined but deviation occurs as the AUV navigates because of oceanic current. LOS controller sets a point on the reference path at the unspecific distance from the AUV as a new target point and determines the angle ψ_r from this set point (x_{los}, y_{los}) to return to the reference path. CTE(Re) is then calculated as shown in Fig. 11. The process of finding the efficient exploration width is as follows:

At first, the twice sensing range is used considering the sensing range of AUV when there is no external disturbance. When external disturbance exists, missing areas will occur due to deviation of AUV position. The deviation of the AUV is expressed as CTE. If the exploration width is reduced from the twice the sensing range by the distance that causes missing areas, the adjustment of the exploration width can be kept minimal while ensuring no occurrence of missing areas, thereby allowing to set the efficient exploration width. If the value of CTE, marked in dashed line in Fig. 11, can be calculated whenever AUV is in motion, EEW can also be calculated using the CTE value. At worst, the deviation of the AUV will be largest when the value of CTE is at maximum. If the exploration width is reduced by the maximum value of CTE, the resulting width would be the maximum width without causing any missing areas and would, therefore, be the efficient exploration width, which can be expressed with the following equation :

$$EEW = 2 * R_{MAX} - 2 * MAX\ CTE \tag{10}$$

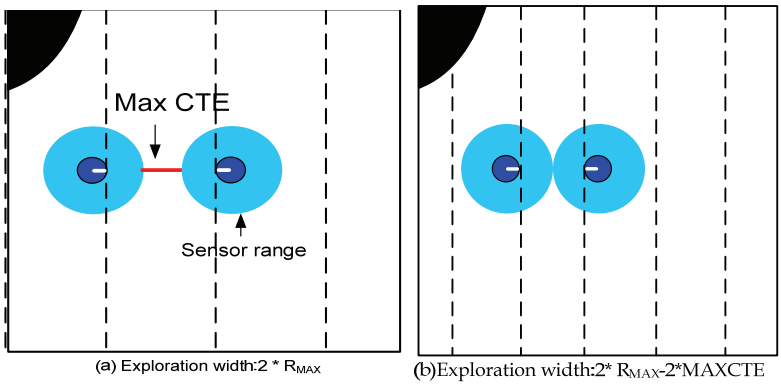


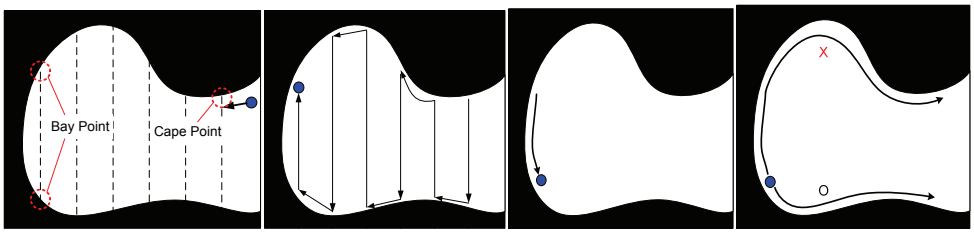
Figure 12. Calculating of EEW using CTE

Here, R_{MAX} is the maximum sensing range of AUV. As indicated in Fig.12(a), the exploration width calculated by subtracting the largest CTE value obtained during navigation from the twice the sensing range of AUV will not cause any missing areas. Although the deviation of AUV will not be as much as the maximum value of CTE every time external disturbance occurs, reducing the exploration width sufficiently using its maximum value would eliminate the possibility of missing areas occurring in advance as shown in Fig.12(b). This can be considered as a trade-off between accuracy and efficiency. Simulation may be used to verify the usefulness of this method of determining the exploration width. The exploration width calculating technique proposed in this study is based on the above described method.

4.3 Proposed coverage method

The bottom line of the limitations of existing Hert algorithm lie in the question of efficiency. One limitation is that internal information is not utilized for the planning of path for the exploration of inlets. The other is that the existence of island can only be established only after completely circulating the island and further exploration of surrounding areas has to follow even if these areas have already been covered. Our proposed technique to overcome such problems is as follows:

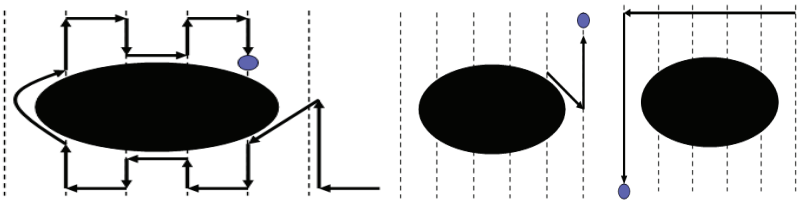
First, the exploration of inlets without islands is performed as shown in Fig. 13.



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

Figure 13. The proposed coverage method for simply connected area: (a) Entering inlet, (b) Exploring inside inlet in zigzag manner, (c) Moving along the boundary of the opposite bay point, (d) Escaping from inlet along the planned short path

As shown in Fig.13(a), the AUV which has been exploring in zigzag movement detects a cape point and enters the inlet. Unlike Hert's method, AUV does not enter the inlet by following the boundary but, instead, explores the inlet in zigzag movements, which will not only allow accurate establishment of the entry of the inlet but also complete coverage of the inside of the inlet by zigzag exploration. Therefore, ocean environment in the inlet can be explored faster than Hert's method. Also, in multi-AUV operation where two entrance areas of the inlet are locked, one AUV can alert the others while entering the inlet preventing collision. Covering the inlet with zigzag movements can also ensure finding of other topographic structure between the bay points as shown in Fig.13(b). By using the topographic structures in the inlet recognized by zigzag exploration, exit route from the inlet can be planned and short route can be identified as shown in Fig. 13(d). When there is another inlet inside the inlet, which is the recursive characteristic of the inlet structure, the AUV covers the inner inlet first, return to the original inlet and continue exploration.
When an AUV finds another inlet while exploring the inlet, it will cover the lower-level inlet first according to the depth-first order and if there is yet another lower level inlet, it will cover the new inlet first, and so on. In other words, exploration of multi-level inlets is performed in the opposite order of entry procedure covering the lowest level inlet first and the highest level inlet, which is entered first, last. So, it is in FILO (Fist-in-last-out) sequence. The following Fig. 14 shows the method of exploration in the area where islands exist.



(a) (b) (c)

Figure 14. Proposed coverage method for non-simply connected area (island)

Unlike Hert's method in which AUV circles around the island and covers the surrounding area again, this method can identify an island with only one circulation effectively saving time and navigation distance as shown in Fig.14(a). Once the AUV completes its path around the island, it can leave the island, move to the opposite side of the island and continue exploration as shown in Fig.14(c). This movement method will allow the AUV to explore the surrounding oceanic environment of the island. This improved covering procedure is well described in Table 1.

5. An efficient coverage method for Multi-AUV

The below proposed algorithm was developed by improving the efficiency of the Hert's method. The method of multi-AUV operation using this algorithm is discussed here. The most important point to consider for multi-AUV operation is that collision between the AUVs has to be prevented. Efficient distribution of AUVs for the saving of time and navigation distance can also be another point of consideration. This paper attempts to address these points using the technique of dynamic role assignment mechanism (Luiz Chaimowicz, 2002). By using this method, each AUV can be given a different assignment and their role can be dynamically modified to deal with changing such as avoiding collision.

```
Local Variable:
Sp:the first presumed start point during the AUV moving around
inlet or island.
Procedure CoverInletorIsland
 Step 1 <Detect Cape point>
          do
        move with zigzag manner for covering;
              if an inlet entrance cape point is detected then
                CoverInletorIsland();
          until AUV reaches a bay point or an artificial bay point or
              AUV returns to start point sp;
              if AUV reaches a bay point or an artificial bay point
then
                CoverInlet();
                move along planned short route to sp;
 Step 2 <Define types of terrain>
       return inlet's entrance point or return to Sp;
       if AUV detects 4 cape points and there is no bay then
              define island terrain;
            CoverIsland();
        else if AUV detects bays more then one then
              define inlet terrain;
              CoverInlet();
 Step 3 <End covering mission>
       back to an entrance point or sp;
            lock doorways capes;
       if AUV is at Sp then
         move along grid line of Sp until boundary is hit;
         end mission;
            else
             repeat Step 1 ~ Step 3
```

Table 1. The proposed coverage algorithm for non-simply connected area (island)

5.1 Definition of dynamic role assignment

The definition of dynamic role assignment is based on the technique of Luiz Chaimowicz (Luiz Chaimowicz, 2002). A team of AUVs must be coordinated to execute cooperative tasks and they have to synchronize their actions and exchange information. In this approach, each AUV performs a role that determines its actions during the cooperative task. According to its internal state and information about the other AUVs and the task received through communication, an AUV can dynamically change its role, adapting itself to changes and unexpected events in the environment. The mechanism for coordination is completely distributed. Each AUV has its own controllers and takes its own decisions based on local and global information. In general, each team members has to communicate explicitly with other AUVs to gather information but they normally need not construct a complete global state of the system for the cooperative execution. We consider that each team member has a specification of the possible actions that should be performed during each phase of the

cooperation in order to complete the task. These actions must be specified and synchronized considering several aspects, such as AUV properties, task requirements, and characteristics of the environment. The dynamic role assignment will be responsible for covering the correct actions to each AUV and synchronizing the cooperative execution.
Before describing in detail the role assignment mechanism, it is necessary to define what a role in a cooperative task is. Webster's Dictionary defines it as follows:

Definition 5.1 *(a) Role is a function or part performed especially in a particular operation or process and (b) role is a socially expected behavior pattern usually determined by an individual's status in a particular society.*

Here, a role is defined as a function that one or more AUVs perform during the execution of a cooperative task. Each AUV will be performing a role while certain internal and external conditions are satisfied, and will assume another role otherwise. Thus, a role depends on the internal AUV state and on information about the environment and other AUVs, and defines the set of controllers that will be controlling the AUV in that moment.
In (Shehory et al., 1998), a role is defined as the specification of an AUV's internal and external behaviors. A formation is a set of roles, decomposing the task space. Each AUV knows the current formation and keeps mapping from teammates to roles in the current formation. Our definition is similar, the main difference being that we do not have the concept of formation, and we use a more formal model to describe roles and role assignments, as it will be further explained in the next sections. As mentioned before, each role defines a AUV controller and the role assignment allows the AUVs to change their behaviors dynamically during the task execution.

5.2 Modeling

The dynamic role assignment can be described and modeled in a more formal framework. In general, a cooperative Multi-AUV system can be described by its state ***(X)***, which is a concatenation of the states of the individual AUVs:

$$X = [x_1, x_2, ..., x_N]^T \tag{11}$$

Considering a simple control system, the state of each AUV varies as a function of its continuous state (x_i) and the input vector (u_i). Also, each AUV may receive information about the rest of the system (z_i) that can be used in the controller. This information consists of estimates of the state of the other AUVs that are received mainly through communication. We use the hat (^) notation to emphasize that this information is an estimate because the communication can suffer delays, failures, etc. Using the role assignment mechanism, in each moment each AUV will be controlled by a different continuous equation according to its current role in the task. Therefore, we use the subscript q,q = 1,…,S, to indicate the current role of the AUV. Following this description, the state equation of each AUV i , i = 1,…,N, during the execution of the task can be defined as :

$$\dot{x}_i = f_{i,q}(x, u_i, \hat{z}_i) \tag{12}$$

Since each AUV is associated with a control policy,

$$u_i = g_{i,q}(x_i, \hat{z}_i) \tag{13}$$

And since $\hat{z}_i$ is a function of the state $\boldsymbol{X}$, we can rewrite the state equation:

$$\dot{x}_i = f_{i,q}(X) \tag{14}$$

or, for the whole team,

$$\dot{X} = F_{\Sigma}(X), \text{ where } F_{\Sigma} = [f_{1,q1,\ldots,}f_{N,qN}]^T, \; q_i \in \{1,\ldots,S\} \tag{15}$$

The equations shown above model the continuous behavior of each AUV and consequently the continuous behavior of the team during the execution of a cooperative task.

5.3 Role assignment

The role assignment mechanism allows Multi-AUV coordination in the execution of cooperative tasks. As mentioned before, dynamically assigning and exchanging roles, the AUVs are able to perform the task more efficiently, adapting to unexpected events in the environment and improving their individual performance in benefit of the team.
Basically, there are two types of role assignment as shown in Fig.15
The main jobs of AUV are scanning and covering which are defined as follows:

- ***SCANNING:*** Movement activity in zigzag pattern to identify inlets. The purpose is to locate the cape points rather than to enter the inlets. When an inlet is identified, its entrance data are collected and transmitted to the central database.
- ***COVERING:*** Exploration activity performed in the inlet using the scanning data from the database.

Covering activities can be performed using improved method described in this paper Section 4. To do this, a central database through which AUVs can communicate as well as a database manager is required as shown in Fig. 15.

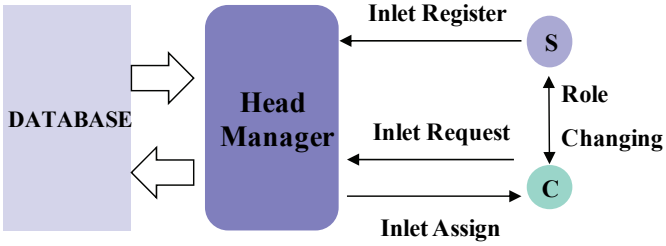


Figure 15. System configuration of dynamic role assignment

Not all the AUVs need to be assigned of a role from the beginning. Instead, only one AUV may take up the role of scanning and begin exploration. Once it encounters an inlet, then each of the other AUVs is given an assignment of covering in sequence and enters the inlet. If an AUV with covering assignment completes its coverage when the scanning AUV is still unable to find the entrance to another inlet, then the covering AUV will have to wait as it has further covering to do. In order to reduce waiting time to improve the efficiency of the whole operation of topographic exploration, the roles of AUVs need be modified. For effective role distribution and prevention of collision during the performance of changed roles, the below rules must be observed.

- *Rule 1*: **There must be only one AUV present in one inlet.**
- *Rule 2*: **If two AUVs must be present because of the existence of an island, the one that entered the inlet later exits the inlet first.**

The Rule 1 must be observed to avoid collision between AUVs. If an island is large in size, two AUVs may have to work in the same inlet risking the danger of collision. However, if the one that entered the inlet later exits the inlet first, the danger can be minimized. Fig. 15 shows role change in the work flow when two AUVs are operated.

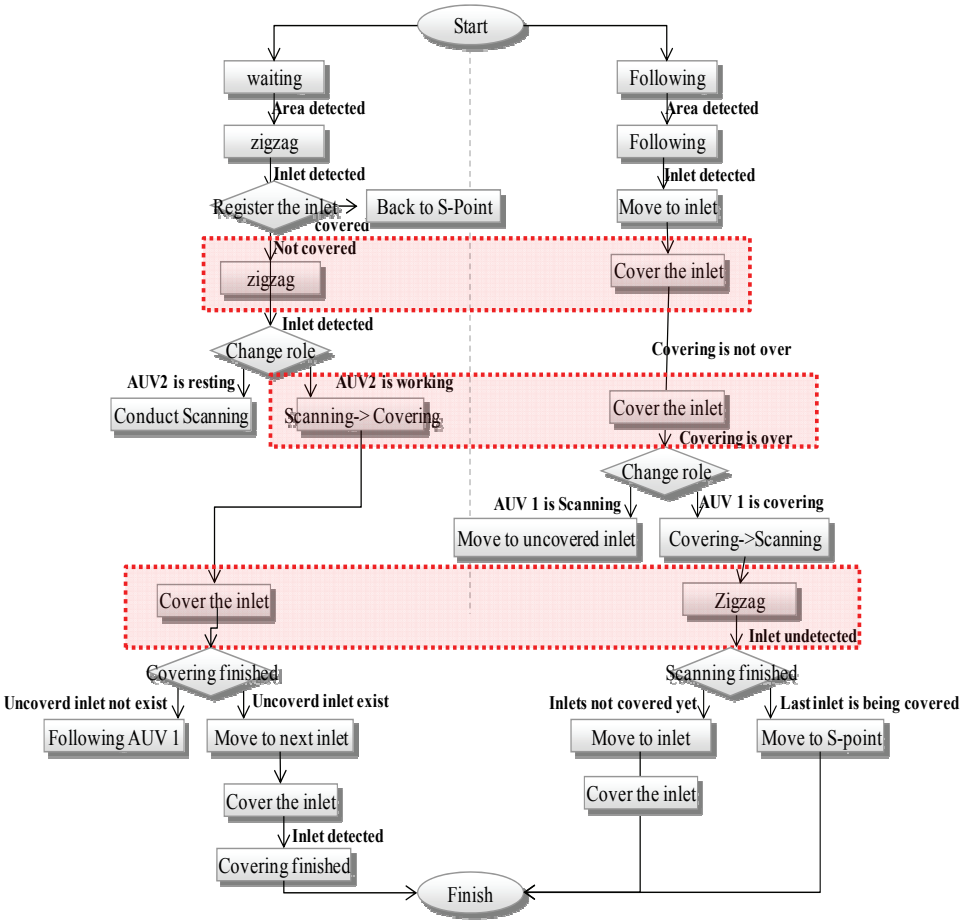


Figure 16. Simulated Multi-AUV work flow

6. Mathematical verification and comparison

The covering algorithm described above is an improved version of Hert's algorithm and is intended to eliminate inefficiency problems of the Hert's algorithm. However, it will be validated in the simulated test environment. Before the simulation test, the proposed

method will first be verified mathematically to prove that it is an effective method in terms of total path length.

Area exploration method is broadly divided into the exploration of known topography and that of unknown. Hert's algorithm is based on Seed Spreader (Lumelsky,1990) technique developed for the exploration of known area and the validity of the method has been proved in terms of path length compared with the Seed Spreader technique. Total path length based on the worst situation using Hert's algorithm (Hert et al,1996) can be described as in Equation (16).

$$l_{tp} \leq l_{gs} + 3l_{id} + 2l_{bc} + 3l_{ib} + 2l_{cb} \tag{16}$$

Here, l_{tp} is the maximum covered path length according to the Hert algorithm. In Hert's algorithm, an exploration consists of 4 procedures: *coverArea, coverInletorIsland, ExitInlet, and ReturntoS.* The total path length $l_{tp,}$ is calculated by adding the maximum navigation distance in each process. The definitions of the parameters used in Equation (16) are listed in Table 2.

Parameters	Descriptions
l_{tp}	Total path length
l_{bc}	Length of the planar outer boundary curve
l_{ib}	Sum of the lengths of the planar island boundary
l_{gs}	Sum of the lengths of all grid line segments in area
l_{id}	Sum of the lengths of all inlet doorways
l_{cb}	Sum of the lengths of all capes on boundary

Table 2. Parameters defined in the Hert's algorithm

The exploration procedures in Hert's method that make up the total path length are as described below. When the entire coverage area is divided into a number of uniform exploration width, AUV will navigate along the grid line in each path in zigzag pattern. Each grid line will be passed only once and each boundary area will be covered not more than once as well. First, the coverArea stage spans the area of coverage along the boundary and ends when the AUV reaches a bay. Coverage of the entire area is completed by repeating this procedure by locking the inlet that has been covered, repeating the coverage of the same area can be avoided. The maximum path length required to cover this procedure is l_{gs}+l_{id}+l_{bc}+$l_{ib.}$ Second, during the procedure of CoverInletorIsland, AUV first explores the inlet which has not been covered yet. The path length for this procedure should not exceed the total length of capes in the inlet. For all inlets, the path length of this procedure is not greater than l_{cb}. When the AUV reaches an inlet, it begins to move along its boundary until it reaches a bay if it is an inlet or until it returns to its original position if it is an island. The total path length of this procedure does not exceed l_{bc}+l_{ib}. When the AUV identifies the current topographic structure as an island, it will continue to explore along the artificial bay for half a circle and the path length of this procedure does not exceed l_{ib}. When the AUV leaves the inlet, it should return to the starting point S_p which will not exceed l_{ib}. The total path length for coverInltorIsland procedure, therefore, shall not exceed $l_{bc} + 2l_{ib} + l_{cb}$.

Finally, the procedure for each AUV to exit the inlet is called exitInlet in which AUV will pass the doorway at least once. As the AUV has already passed the doorway during coverArea procedure, this segment is not calculated again here. When the AUV returns to the starting point, a length of l_{id} is added. It is the maximum length required for the AUV to cover the locked doorway.

6.1 Path length of the proposed coverage method

It is not easy to describe the total path length in the proposed method of coverage using the same notations used in Hert's algorithm. In order to display how much the total path length can be reduced in the proposed algorithm, it is necessary to split the boundary, island and inlet, which are the coverage areas of Hert's algorithm, into (l_{lb},l_{sb},l_{ba}), (l_{bi},l_{si}), (l_{ld},l_{sd},l_{bd}) so that the total path length can be compared with the result from proposed algorithm using new notations. The total path length in the proposed algorithm is calculated by the following Equation (17):

$$l_{tp} \leq l_{gs} + 4l_{id} + 2l_{ba} + \frac{1}{2}(3l_{sb} + l_{lb}) + \frac{1}{2}(l_{bi} + l_{si}) + l_{ex} + 2l_{bd} + \frac{3}{2}l_{sd} + \frac{1}{2}l_{ld} + l_{cb} \quad (17)$$

The definitions of the parameters used in the Equation (17) are listed in Table 3.

Parameters	Descriptions
l_{pt}	Total path length of proposed method
l_{sb}	Sum of the lengths of the short boundary of area
l_{lb}	Sum of the lengths of the long boundary of area
l_{ba}	Sum of the lengths of the bay of area
l_{sd}	Sum of the lengths of the short boundary of inlet
l_{ld}	Sum of the lengths of the long boundary of inlet
l_{bd}	Sum of the lengths of the bay of inlet
l_{si}	Sum of the lengths of the short boundary of inlet
l_{bi}	Sum of the lengths of the long boundary of Island
l_{ex}	Length of getting out of the area based on calculated short path

Table 3. Parameters defined in proposed coverage method to compare with Hert's

Using the parameters define in Table 3, the total path length can be calculated as follows:
As in Hert's algorithm, new proposed algorithm also consists of 4 procedures. The required path length for the first procedure, coverArea is:

$$l_{gs} + l_{id} + 2l_{ba} + \frac{1}{2}(3l_{sb} + l_{lb})$$

The required path length for the second procedure, coverInletorIsland is:

$$2l_{ib} + 2l_{bd} + l_{sd} + \frac{1}{2}(l_{bi} + l_{ex}) + l_{ex} + \frac{1}{2}(l_{ld} + 3l_{sd})$$

This includes the distance of movement from the start point to the inlet and the distance of returning to the entrance of the inlet after covering the bay. The required path length for the third procedure for leaving the inlet, ExitInlt is l_{cd}. The required path length for the final procedure, ReturntoS is l_{gs}.

6.2 Comparison of two algorithms

The total path length required in Hert's algorithm and in the proposed algorithm was described in the above section 6.1. However, these two calculation method can not be compared easily as they do not share common parameters. Therefore, the Equation (16) for Hert's algorithm is modified relative to the Equation (17) as follows:

$$l_{tp} \leq l_{gs} + 5l_{id} + l_{ba} + l_{bd} + \frac{1}{2}(3l_{lb} + l_{sb}) + \frac{1}{2}(5l_{bi} + 3l_{si}) + \frac{1}{2}(3l_{ld} + l_{sd}) + l_{cb} \tag{18}$$

This Equation (18) is the modification of Equation (17), only using the same parameters as used in Hert's algorithm. This new Equation (18) is identical to the original equation except that it was designed for comparison purpose using the same parameters. For direct comparison, Equation (17) and (18) are combined as follows:

$$\begin{aligned} l_{pt} &\cong l_{tp} + (l_{bd} + l_{sb} + l_{bd} + l_{sd} + l_{ex}) - (l_{lb} + l_{ld} + l_{id} + 2l_{bis} + l_{si}) \\ &\cong l_{tp} + (l_{ba} + l_{sb} - l_{lb}) + (l_{bd} + l_{sd} - l_{lb}) + (l_{ex} - l_{id} + 2l_{bi} + l_{si}) \end{aligned} \tag{19}$$

Equation (19) can be split as follows:

$$l_{lb} > l_{ba} + l_{sb} \tag{20}$$

$$l_{lb} > l_{bd} + l_{sd} \tag{21}$$

$$l_{id} + 2l_{bi} + l_{si} > l_{ex} \tag{22}$$

Equation(20) and (21) can be explained as follows: Comparison of path length between Hert's algorithm and proposed improved algorithm is based on the worst scenario. The proposed algorithm was developed to supplement the lack of scheme in Hert's algorithm. The implication of this modification is, as explained in the previous section, that the proposed method, which uses environmental data for the coverage of inlets and islands, results in the reduction of path length. Equation (22) represents the shortest total path length from the coverage area after the exploration. This may differ depending on the characteristics of the topography, but it will certainly reduce the navigating distance as it uses the shortest path. The efficiency of the proposed algorithm was proved to have path reducing effect in the above 3 aspects.

7. Simulation results

In order to verify the performance of the proposed method, a simulation of the environment was designed. In order to minimize the difference from the real oceanic environment, REMUS developed by Hydroid was selected as a model as described in Section 3 and the characteristics of real oceanic environment were applied as much as possible using the oceanic current model. REMUS as shown in Fig.17 is torpedo-shaped flight vehicle of 4 feet in length and maximal 7.5 inch in diameter. The cruise speed of the vehicle is V=1.53 m/s and operating depth range is from 40 to 100 ft deep. Fig .18 shows that the vehicle which has roughly 80

pound as vehicle. Its four fins on either side or forward of the propeller allow pitch and yaw motions for maneuvering. The simulation mission scenario is simply designed to explain how the algorithm is working. The sea current flow speed is defined as a = 0.5 m/s which is slower than the vehicle speed, but changes in direction in different simulation.

Figure 17. The REMUS was developed by Woods Hole Oceanographic Institution (WHOI)

7.1 Single AUV Coverage without sea current disturbance

In the simulated environment, the verification of the efficiency of the proposed algorithm was performed first under the ideal conditions without applying AUV dynamics and sea current disturbances. Simulation was performed using complex terrain and simple terrain.

(a) (b)

Figure 18.Types of underwater terrain: (a) Simple Terrain. (b) Complex terrain

Coverage method	Covered path length(m)	Covering time(sec)
Hert's	1353.99	1354
Proposed	1311.33	1311
improvement	42.66(3.17%reduction)	43

Table 4. Simulation result of simple terrain

The maps of the oceanic terrains were prepared as BMP files of 800 (H) x 600 (V) pixels. One pixel represents 100cm x 100cm and sensing range and exploration width were specified to be 20 pixels and 40 pixels, respectively. The results of the test in the terrains described in Fig. 18(a) are shown in Table 4. Table 5 shows that path length was reduced by 3.17% and time by 43 seconds using the proposed method. These results are the proof of the efficiency of the proposed exploration method in finding an inlet, entering it in zigzag pattern up to the bay point and returning via shortest distance, as explained above. This is more confirmed by the test using more complex terrain which resulted in greater difference as shown in Table 5.

Coverage method	Covered path length(m)	Covering time(sec)
Hert's	2022.75	2023
Proposed	1795.32	1795
improvement	227.43 (11.27% reduction)	228

Table 5. Simulation result of complex terrain

As indicated by the results in Table 5, it can be predicted that the efficiency of the proposed method will be more pronounced in the exploration of complex terrains in shallow sea. This also indicates that the proposed method would be useful for operations that are usually carried out in shallow seawater such as mine countermeasure (MCM) operations, rescue operations and assaults.

7.2 Single AUV coverage with sea current disturbance

In this section, we will look into the results of exploration using the proposed algorithm applying internal and external factors of cross track errors. From simulation test, missing areas caused by cross track errors could be identified as shown in Fig. 19.

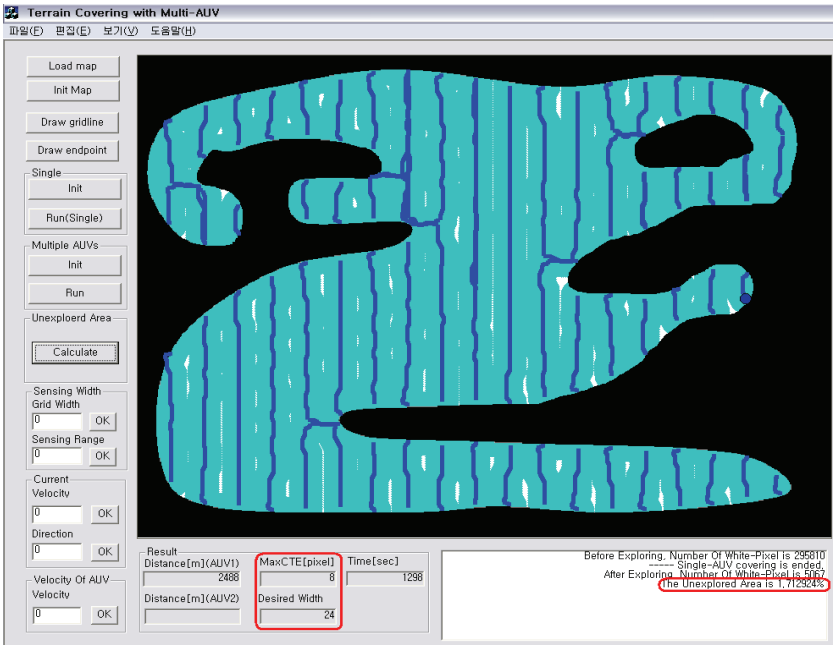
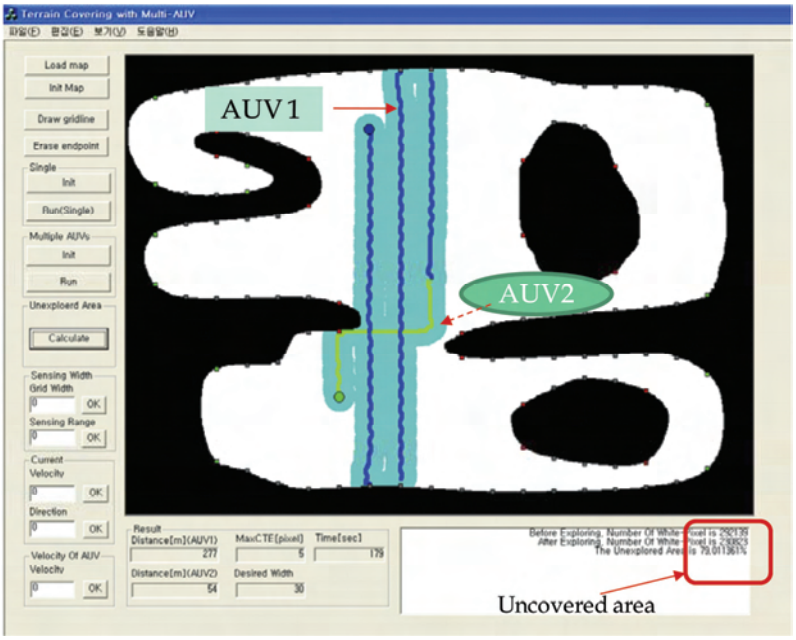


Figure 19. Single AUV coverage with sea current in simulation

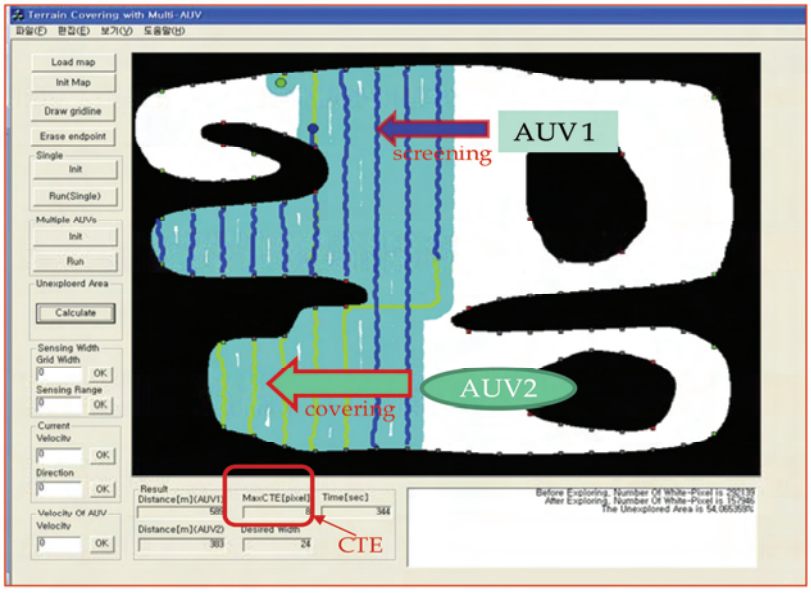
The results of simulation test are shown in Fig. 19 where missing areas are revealed as white areas and indicated at the right bottom of the screen. The exploration width and maximum CTE required for resolving these missing areas are indicated at the bottom center of the screen. The existence of missing areas such as these can be fatal in military operations where accuracy is critical. The EEW value from this test was modified to 24 (2x20 – 2x8) after re-calculation. White missing areas were eliminated completely from the results of the test performed again using this new value. The new results indicated that using the new value 24 as EEW, variability of the path caused by CTE has been fully optimized although exploration path length and time were longer because of reduced exploration width.

7.3 Multi-AUV coverage with sea current disturbances

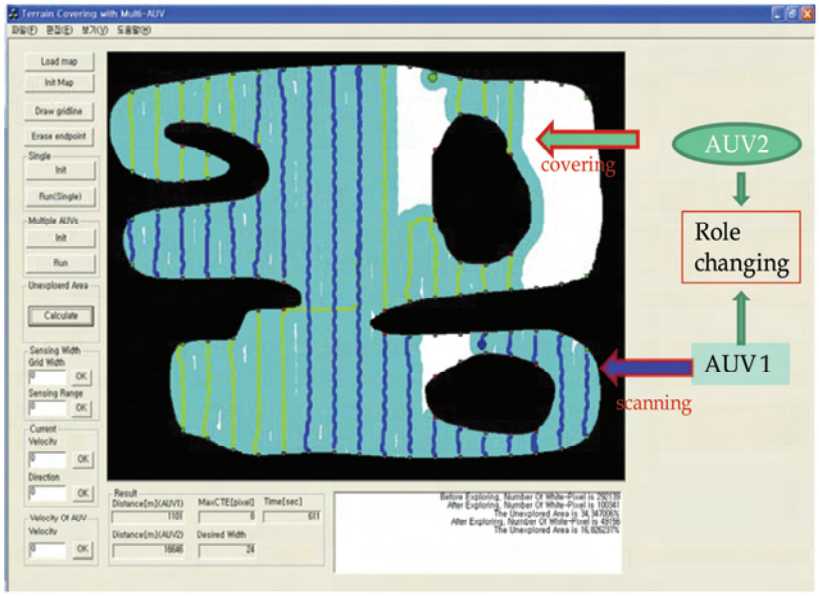
The results of the exploration test using dynamic role assignment mechanism, as shown in Fig.20, which was proposed in this study as the exploration method for two-AUV operation, are summarized in Table 6.



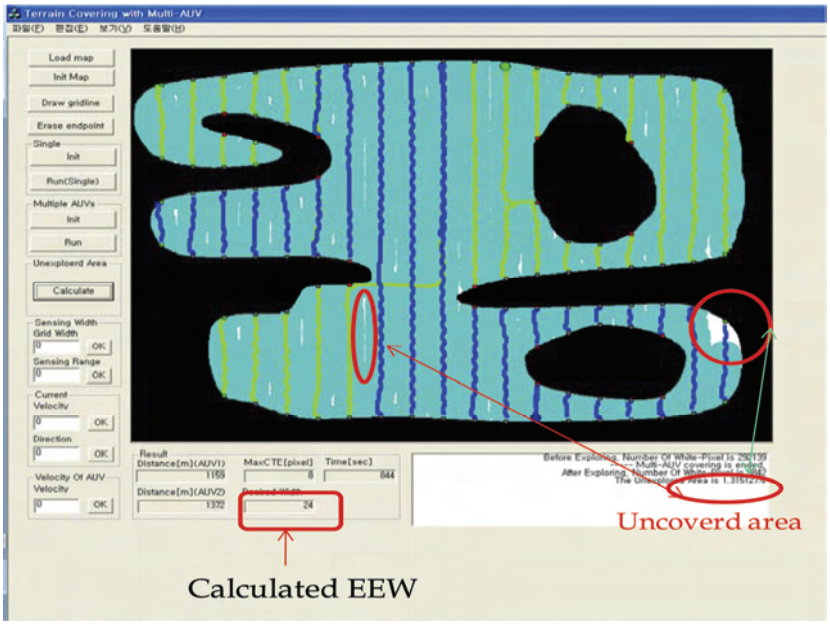
(a)



(b)



(c)



(d)

Figure 20. Multi-AUV simulated the covering method in an unknown terrain without EEW by following steps: (a) The AUV starts covering to detect cape. (b) Two AUVs define coverage roles.(c) Two AUVs change role considering coverage situation.(d) Results after completed coverage

Mode	Total Distance(m)	Time(sec)
Single-AUV	1797.72	1298
Multi-AUV	1979.32	800

Table 6. Simulation result compared to single AUV and Multi- AUV

As can be seen from these results, total time taken for entire exploration was reduced considerably although total path length of the two AUVs was increased. Also, it could be confirmed that dynamic assignment of the roles of covering and screening in this multi-AUV operation using the new proposed method was successfully carried out in accordance with the environmental characteristics.

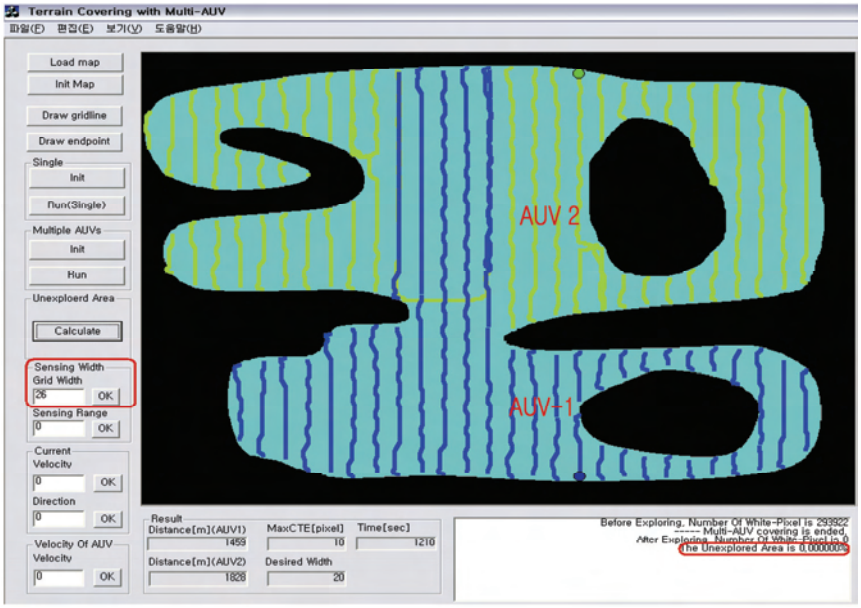


Figure 21. Simulation result of Multi-AUV with EEW

Fig. 21 shows the results of exploration using two AUVs. These results also confirmed that the problem of missing areas could be resolved by applying re-calculated EEW.

8. Conclusion

This paper presents an on-line coverage method for the exploration of unknown 3-dimensional oceanic terrains using multiple autonomous underwater vehicles (AUVs). Based on the concept of planar algorithm developed by Hert, this study attempts to develop an improved method. Instead of theoretical research, it focuses on practical aspect of exploration considering the equation of motion for AUVs that are actually used in oceanic exploration as well as characteristics of complex oceanic topography and other realistic variables such as sea current. These elements of consideration are used to calculate cross track error (CTE) and path width. The validity of the improved algorithm for terrain

coverage was first verified mathematically and then simulation of the real underwater environment was used to prove it by analyzing the path length and time taken for the coverage as well as missing areas, which is the key element of efficiency. In order to apply the improved method to the Multi-AUV operation, each AUV was assigned of covering or screening role by means of dynamic role assignment mechanism. Finally, the proposed method was tested in simulated environment where optimal exploration width was calculated and complete exploration without missing areas was successfully proven. We expect future study to employ REMUS, which we used as a model, and to resolve the hypothesis-based communication problems as well.
Also, the results showed that Multi-AUV operation has advantage over single AUV operation. The proposed method of this study will not only be useful to commercial applications but to mine counter-measure (MCM) and rapid environmental assessment (REA) as part of a naval military operation as well. We also believe that it will be ideal for use in variable oceanic environment particularly in shallow water terrains

9. Acknowledgement

This work was supported in part by MOCIE Industrial Technology Development Program, the ASRI, and BK21 Information Technology at Seoul National University.

10. References

A. J. Healey and D. Lienard (1993). Multivariable sliding mode control for autonomous diving and steering of manned underwater vehicles, *IEEE Journal of Oceanic Engineering*, vol.18, no.3, pp.327-339.

A. Zelinsky, R. A. Jarvis, J. C. Byrne, and S. Yuta (1993). Planning paths of complete coverage of an unstructured environment by a mobile robot, *Proceedings of International Conference on Advanced Robotics*, pp.533-538, Tokyo, Japan.

B. Yamauchi, A. Schultz, and W. Adams (1998). Mobile Robot Exploration and Map-building with continuous Localization, *Proceedings of International conference on Robotics and Automation*, pp.3715-3720.

C. S. Kong, N. I. Peng, and I. Rekleitis (2006). Distributed Coverage with Multi-Robot System, *Proceedings of the IEEE International Conference on Robotics and Automation*, Orlando, Florida, USA.

D.Kurabayashi,J.Ota,T.Arai and E.Yoshida (1996). Cooperative sweeping by multiple mobile robots,*Conference on Robotics and Automation* .

D. MacKenzie and T. Balch (1996). Making a clean sweep: Behavior based vacuuming, *AAAI Fall Symposium*, Instationating Real-World Agents.

DSME(INC) (1999). AUV 'OKPO-6000' Development paper, Korea.

E. Fiorelli, N.E. Leonard, P. Bhatta, D.A. Paley, R. Bachmayer, and D.M. Fratantoni (2004). Multi-AUV Control and Adaptive Sampling in Monterey Bay, *Proceedings of the IEEE Autonomous Underwater Vehicles: Workshop on Multiple AUV Operations*.

E. U. Acar, H. Choset, and P. N. Atkar (2001). Complete Sensor-based Coverage with Extended-range Detectors: A Hierarchical Decomposition in Terms of Critical Points and Voronoi Diagrams, *Proceedings of International conference on Intelligent Robots and Systems*, pp.1305-1311.

F.Belkhouche,B.Belkhouche, and P.Rastgoufard (2006). Line of sight robot navigation toward a moving goal.*IEEE Transactions on System,Man, and Cybernetics* 36 (2),255-267.

Fodrea, Lynn (2002). Obstacle avoidance control for the REMUS autonomous underwater vehicle, I, Thesis, Naval postgraduate school.

H. Bash and Susan Hert (1995). The Area Partitioning Problem, *Proceedings of the 12th Canadian Conference on Computational Geometry*.

H. Choset (2000). Coverage of Know Spaces: The Boustrophedon Cellular Decomposition, *Autonomous Robots*, vol.9, pp.247-253.

H. Choset (2001). Coverage for robotics-A survey of recent results, *Annals of Mathematics and Artificial Intelligence*, vol.31, pp.113-126.

H. T. Choi, A. Hanai, S. K. Choi, and J. Yuh (2003). Development of an Underwater Robot, ODIN-III, *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Las Vegas, Nevada, USA.

I.A.Wagner ,M.Lindenbaum and A.M.Bruckstein (1996). Smell as a computational resource – a lesson we can learn from the ant,*ISTCS* pp.219-230.

I.A.Wagner ,M.Lindenbaum and A.M.Bruckstein (1998). Efficiently searching a dynamic graph by a smell oriented vertex process,*Ann.Math.Artif.Intell.*24 211-223.

I. Rekleitis, V. Lee-Shue, A. P. New, and H. Choset (2004). Limited Communication, Multi-Robot Team Based Coverage, *Proceedings International Conference on Robotics and Automation*, pp.3462-3468, New Orleans, L. A., USA.

J. C. Latombe. Robot Motion Planning , *Kluwer Academic*, Boston, M. A., USA

Ko, Yu-hyun (2007). Estimation of the path deviation of an underwater vehicle due to currents and its application, *M.S. Thesis*, Seoul National University, Korea.

L. Chaimowicz, M. F. M. Campos, and V. Kumar (2002). Dynamic Role Assignment for Cooperative Robots, *Proceedings of the IEEE International Conference on Robotics and Automation*, Washington DC, USA.

Lee, Seong-yong (2007). A Study on terrain covering for Multi-AUV, *M.S. Thesis*, Seoul National University, Korea.

N. Agmon, N. Hazon, and G.A. Kaminka (2006). Constructing Spanning Trees for Efficient Multi-Robot Coverage, *Proceedings of the IEEE International Conference on Robotics and Automation*.

N. Hazon and G.A. Kaminka (2005). Redundancy, Efficiency and Robustness in Multi-Robot Coverage, *Proceedings of the IEEE International Conference on Robotics and Automation*.

N. R. Gracias, S. Zaan, A. Bernardino, and J. Santos-Victor (2003). Mosaic-Based Navigation for Autonomous Underwater Vehicles, *IEEE Journal of Oceanic Engineering*, VOL.28, NO.4.

P. K. Paim, B. Jouvencel, and L. Lapierre (2005). A Reactive Control Approach for Pipeline Inspection with an AUV, *Proceedings of MTS/IEEE*, OCEANS.

R. Alur, C. Coucoubetis, T. A. Henzinger, P. H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine (1995). The algorithmic analysis of hybrid systems, *Theoretical Computer Science*, 138:3-34.

R. Emery, K. Sikorski, and T. Balch (2002). Protocols for Collaboration, Coordination and Dynamic Role Assignment for a Robot Team, *Proceedings of the IEEE International Conference on Robotics and Automation*, Washington DC, USA.

R. Zlot, A. Stentz, M. B. Dias, and S. Thayer (2002). Multi Robot Exploration Controlled by a Market Economy, *Proceedings of IEEE International Conference on Robotics and Automation*, pp.3016-3023.

S.G.Shuzhi and L.L Frank (2006).Autonomous Mobile Robots,*CRC Press*,Boca Raton,FL pp .418-459

S. S. Ge, C-H. Fua, and K. W. Lim (2004). Multi-Robot Formations: Queues and Artificial Potential Trenches, *Proceedings International Conference on Robotics and Automation*, pp.3345-3350.

S. Hert, S. Tiwari, and V. Lumelsky (1996). A Terrain-Covering Algorithm for an AUV, *Underwater Robots*, Kluwer Academic Publishers, pp.17-45, Boston.

S. X. Yang and C. Luo (2004). A Neural Network Approach to Complete Coverage Path Planning, *IEEE Transactions on Systems, Man and Cybernetics - Part B*, vol.34, no.1, pp.718-725.

Son-Cheol Yu, Tamaki Ura, and Nose Yoshiaki, "Multi-AUV based cooperative observations", *IEEE/OAS Autonomous Underwater Vehicles*.

T. I. Fossen (1994). Guidance and Control of Ocean Vehicle, *Chichester: John wiley & sons*.

T. Prestero (2001). Verification of a six-degree of freedom simulation model for the REMUS autonomous underwater vehicles, *Master thesis*, Department of ocean engineering and mechanical engineering , MIT, USA.

V. Lumelsky, S. Mukhopadhyay, and K. sun (1990). Dynamic path planning in sensor-based terrain acquisition, *IEEE Transactions on Robotics* pp.462-472.

W. H. Huang (2001). Optimal Line-sweep-based Decompositions for Coverage Algorithms, *Proceedings of the IEEE Conference on Robotics and Automation*.

W. H. Huang (2000). The Minimal Sum of Altitudes Decomposition for Coverage Algorithms, *Rensselaer Polytechnic Institute Computer Science Technical Report* 00-3.

W. Huang (2000). Optimal line-sweep decompositions for coverage algorithms, Technical Report 00-3, Rensselaer Polytechnic Institute, Department of Computer Science, Troy, NY, USA.

W.Naeem, R.Sutton,S.M.Ahmad, and R.S.Burns (2003).A review of guidance laws applicable to unmanned underwater vehicles.*The Journal of Navigation* 56,15-29

X. Zheng, S. Jain, S. Koenig, and D. Kempe (2005). Multi-Robot Forest Coverage, *Proceedings of IROS 2005*, Edmonton, Canada.

Y. Gabriely and E. Rimon (2003). Competitive On-line Coverage of Grid Environments by a Mobile Robot, *Computational Geometry*, vol.24, no.3, pp.197-224.

Yan Hou and Robot Allen (2007).Intelligent behavior-based team UUVs cooperation and navigation in a water flow environment, *Ocean engineering,Volumes 35,Issues 3-4,March 2008,*Page 400-416.

Yoav Gabriely and Elon Rimon (2001). Spanning-Tree Based Coverage of Continuous Areas by a Mobile Robot, *Proceedings of the IEEE International Conference on Robotics & Automation*, p.p.21-26, Seoul, Korea.

Z. J. Butler, A. A. Rizzi and R. S. Hollis (2000). Complete distributed coverage of rectilinear environments, *Proceedings of the Workshop on the Algorithmic Foundations of Robotics*.

6

Dispersion and Dispatch Movement Design for a Multi-Robot Searching Team Using Communication Density

Feng-Li Lian, You-Ling Jian and Wei-Hao Hsu
Department of Electrical Engineering, National Taiwan University
Taipei, Taiwan

1. Introduction

In recent years, since the cooperation of multiple robots provides numerous advantages in engineering applications, the research topics on multi-robot systems have received great attention. Due to the complexity and/or difficulty of a task, the performance by one single robot is usually limited. However, the performance limitation could be overcome by the cooperation of multiple robots. Compared with one single robot, a multi-robot team could provide advantages such as better efficiency, robustness, execution time saving, and so on, which are mainly achieved because of the characteristics of multi-robot cooperation. In a multi-robot team, the possible abilities of each robot can be usually classified as the following four types: communication, sensing, computation, and actuation. Among these abilities, the communication between robots is the most important factor for the cooperative characteristics of a multi-robot team. If every robot is considered as a networking node, the communication links between any pair of robots form a communication network. In order to guarantee the cooperative characteristics of a multi-robot team, the method of maintaining a good communication network thus becomes a critical issue.

Considering the robot mobility, environmental limitation, user requests or other factors, many moving algorithms driving robots to form a good communication network are previously proposed. In most of these algorithms, robots are assumed to be well equipped with advanced sensing or positioning devices such as global positioning system, radar and/or camera. These equipments could provide enough information for the moving algorithms and hence output good performance. Sometimes, because of the computation complexity of the algorithm, other advanced processing units are also needed. These assumptions, however, are not easily implemented in real applications. Moreover, cost or retrieve would be another important design issue for the searching task by a robot team.

To achieve the cooperative characteristics and information sharing, the team of robots must maintain a good communication connection while moving or performing tasks. In real implementation, sharing enough information quantity among multiple robots could be a challenging problem. In this chapter, inspired by the bacteria mobility behavior, a dispersion movement design for a multi-robot team is presented. The goal of the dispersion algorithm is to maintain the communication network among robots and enlarge the coverage area as widely as possible. Unlike other previously proposed algorithms, the

design of the dispersion algorithm is distributed and only utilizes simple information. The communication density, or the number of communication links, is the only information needed in the algorithm. Although more information brings better efficiency, it also causes problems like cost and retrieve. Hence, in the proposed algorithm, the position information is not required that greatly reduces the implementation cost on positioning devices. This information could be easily obtained without complex equipments such as global positioning system, camera or radar. Hence the advantage of low information quantity can be achieved. Moreover, the bacteria foraging behavior along with attraction and repulsion model of an organism is also integrated into the algorithm. These biomimic behaviors help the robot team overcome the limitations caused by lacking of position information.
For a base station that releases robots to search for targets, a dispatch rule is also designed based on the dispersion algorithm. Because of the communication and sensing limitations, the coverage area by robots is limited. In order to enlarge the coverage area, "the base station needs to add new robots. Two key factors decided by the base station are "when to release robots" and "how many robots to be released." Therefore, with the dispatch rule, the base station could flexibly re-supply robots to enlarge the coverage area adaptively.

1.1 Related Work

Multi-robot systems are nowadays a popular research topic. In many applications, exploring an unknown environment is an important research task. Robots equipped with sensors form a so-called "sensor network" to explore or map the environment. By considering energy consumption, communication quality and scalability, the communication topology of a team of robots is an important issue related to the sensor network system. Wireless sensor networks are generally used in environment monitoring or exploring. The robots are equipped with sensors such as thermal, pressure or moisture sensors. With information collected by the networked sensors, the status about the environment could be completely obtained. The robots in wireless sensor networks are generally static and distributed. Due to the issue of signal transmission degradation, these robots should select a proper set of neighbors for establishing communication links. This kind of mechanism is called topology control. Topology control for a static sensor network is widely studied and many protocols are proposed: (Santi, 2005; Rodoplu & Meng, 1999; Li et al., 2005; Blough et al., 2003). However, a multi-robot system is usually mobile. Hence the deployment of a team of robots becomes an important issue in a mobile sensor network (Heo & Varshney, 2005; Chellappan et al., 2007).
Mobile sensor networks are not fully considered in many proposed topology control protocols. In the robotics research area, movement algorithms for the mobile multi-robot systems have been significantly studied. Based on the theory of graph theory or potential field, a variety of algorithms are proposed for achieving different movement purposes. For exploring or monitoring the environment, a team of robots need to be distributed uniformly in an environment, and/or they need to form a certain formation for improving the operation efficiency. To achieve these requirements, many movement algorithms are proposed. For example, Tan proposed a distributed model for a multi-robot system by utilizing Delaunay Triangulation and an autonomous self-deployment algorithm is then discussed (Tan, 2005). Cortes et al. proposed a coverage control by focusing on the Voronoi cell maintenance of each robot. A gradient descent algorithm for a class of utility functions is then proposed to encode the optimal coverage and sensing policy (Cortes et al., 2004). Burgard et al. focused on the target points and set cost functions to decide the movement of robots (Burgard et al., 2005).

Poduli and Sukhatme considered the problem of maximizing the total sensor coverage area with the constraint of at least K neighbors linked with every robot. Two virtual forces of attraction and repulsion are defined between robots to force them to move until at least K neighbors is within the neighbor set (Poduli & Sukhatme, 2004).
In addition to the algorithms developed based on the analytic method of potential field and graph theory, each robot in a multi-robot team can be considered as an organism, and the network formed by robots can be viewed as an organism aggregation. The aggregation behavior in nature is widely studied in biological research area. Typical force components of an aggregation model include locomotory, aggregative, arrayal and random (Flierl et al., 1999; Parrish et al., 2002; Czirok et al., 1996; Gueron et al., 1996). For the aggregative force, long-range attraction and short-range repulsion is the most typical model (Gueron et al., 1996; Gazi & Paasino, 2003; Gazi & Passino, 2004). When the neighbors get too close to the individual, a repulsion force is caused to part the individual from those too-closed neighbors. On the other hand, the neighbors that are a little far from the individual will cause an attraction force to the individual. The concept of the attraction-repulsion force model is later applied into the multi-robot dispersion movement algorithm to maintain the communication network.
In nature, the E. coli bacteria have a foraging mechanism for searching for food and avoiding noxious substances. The flagellum of bacteria causes two kinds of actions: (1) the swimming action, which is to make the cell move forward, and (2) the tumbling action, which is to make the cell change direction. When in the isotropic homogeneous environment, the bacteria alternate swim with tumble, and, hence, they move in random walks. This enables the bacteria to search for nutrients. While the bacteria encounter a nutrient gradient, they spend more time swimming and less time tumbling. The direction is biased towards increasing nutrient gradient, and the nutrient concentration data of past steps are used to decide how long to run. In other words, although the bacteria do not have any positional information about the environment concentration, they use the "time density" instead of "spacial density" to make movement decisions (Passino, 2002). In the target search problem studied, to maintain a proper communication network, one robot need to keep a certain number of communication links. In other words, the "communication density" is desired to reach a certain value. This is similar to the bacteria searching of nutrient. Hence the bacteria foraging behavior is applied as a driving force for robots to move to the area with a desired communication density.
Many algorithms utilizing large information quantity are proposed for multi-robot systems applied to environment exploration, sensor network, formation control and other topics. In this chapter, the multi-robot deployment movement problem with a dispatch mechanism is discussed. The design of the dispersion algorithm is distributed and only utilizes simple information. Although more information brings better efficiency, it also causes problems of high cost and retrieving difficulty. Hence, in the proposed algorithms, the position information is not required that greatly reduces the implementation cost on positioning devices. Moreover, the bacteria foraging behavior along with the attraction and repulsion model of organism is also integrated into the algorithm. These biomimic behaviors help the robot team to overcome limitations for lack of position information.

1.2 Outline

The rest of the chapter is organized as follows. Section 2 formulates the dispatch and dispersion problem. Section 3 presents the dispersion movement algorithm of multi-robot

systems inspired by the bacteria foraging behavior and the aggregation of creatures. Section 4 presents the control rule of dispatching robots at the base station. Section 5 discusses the scenario and the statistics of related simulation results. Finally Section 6 summarizes this chpater.

2. Problem Formulation

In the section, a target search problem combined with a multi-robot dispersion movement and a robot dispatch rule in a base station is defined. First, the unknown environment is assumed to be on a two-dimensional plane without boundary and obstacles. Certain target exists at unknown position in the environment, and robots are released from a base station for searching for the target. Since the position of the target is unknown, an appropriate number of robots are required to be able to cover a certain area and the number is not decided in advance. Hence the base station needs to release robots continuously until the target is found. Once the target is sensed by any of the robots, the target search task is finished.
Figure 1 shows a schematic diagram of the target search problem. The square in the center of the plane denotes the base station which is responsible for releasing robots. The circular patterns denote the robots which have been released, and the star symbol denotes the target. Since the position of the target is unknown, the size of required coverage area by a team of robots could not be estimated in advance. That is, the appropriate number of robots that are able to sense the target can not be decided in advance. To solve this problem, the base station needs to release robots continuously for enlarging the coverage area until the target is found. Once the target is sensed by one of the robots, the target search task is finished.

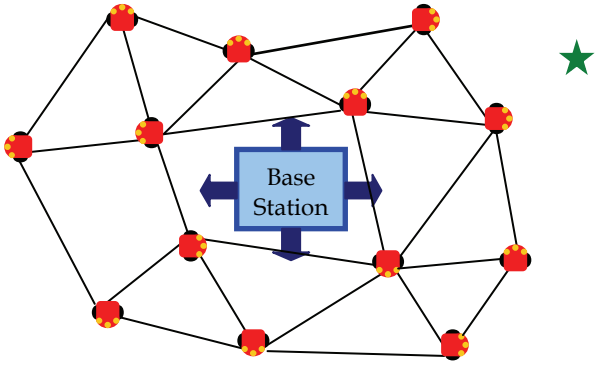


Figure 1. Schematic diagram showing the target search problem

For a multi-robot system, information sharing is one of the key advantages compared with a single-robot system. In general, the communication between robots has a limited range, defined as the "*communication range*" of the robots. A schematic diagram is shown in Figure 2. The black dots denote the robots. The large circle denotes the communication range of Robot 1. Robot 2 is inside the communication range of Robot 1. Hence Robot 2 is called a "*neighbor*" of Robot 1. On the other hand, Robot 3 is outside the communication range of Robot 1. Therefore Robot 3 is not a neighbor of Robot 1. The sensible range of robots for the target is also limited. The range is defined as "*sensing range*", which is denoted by the small gray circle in Figure 1.

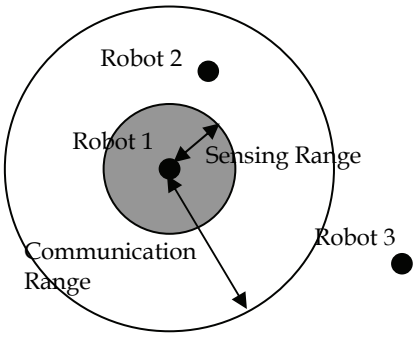


Figure 2. Schematic diagram showing the sensing and communication ranges of a robot

For the target search problem, there are two major tasks: (1) the dispatch manner at the base station, and (2) the dispersion manner on each individual robot.

2.1 Dispatching at the Base Station

The duty of the base station is to release robots until the target is found. Since there is no prior information of the location of the target, and these robots need time to spread out, the base station should decide the following two issues: "*when* to release new robots" and "*how many* new robots to be released." To make a proper decision, the base station needs some information of the dispersion manner of the robots in advance, or it can receive real-time information from the robots.

2.2 Dispersion of Robots

Once robots are released from the base station, they start to move. To simplify the problem, these robots are set to move along the following four directions: go forward, go backward, move right and move left, as shown in Figure 3. Also the velocity of each robot is assumed to be the same. During the movement, the robots should spread out as widely as possible while keeping the communication network connected. Many movement algorithms are developed previously and can provide good efficiency. However, they are usually designed based on rich information such as the position of robots with specialized equipments. Sometimes a leader is also needed to direct other robots. The main objective of this chapter is to develop a dispersion algorithm for robots that only uses simple and easily obtained information. Hence, simple distributed computation can be implemented on each robot.

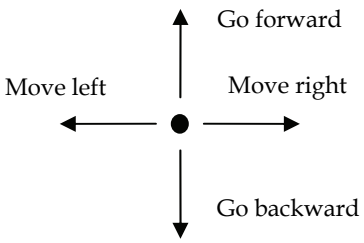


Figure 3. Action definition of a single robot

3. Dispersion Algorithm

In the target search problem studied, to maintain a proper communication network, robots need to keep a certain number of communication links. In other words, the "communication density" is desired to reach a certain value. This is similar to the bacteria searching of nutrient. Hence the bacteria foraging behavior is applied as a driving force for robots to move to the area with a desired communication density. Moreover, the advantage of not using the position information is achieved. This helps decrease the implementation cost and solve problems like retrieving. Simply speaking, the algorithm works as follows. When the neighbors of the robot get too close to the robot, a repulsion force is caused to separate the robot from those too-closed neighbors. On the other hand, the neighbors that are a little far from the robot generate an attraction force to the robot. This attraction-repulsion concept is combined together with the bacteria foraging behavior to design the dispersion algorithm.

Symbol	Definition
i	Robot ID
k	Time step index
t	Releasing index of base station
R_s	Radius of sensing range
R_c	Radius of communication range
R_r	Radius of repulsion range
$n_c^i(k)$	Number of neighbors in communication ring range of Robot i at Step k
$n_r^i(k)$	Number of neighbors in repulsion range of Robot i at Step k
N_c	Desired number of neighbors in communication ring range
$A^{total}(k)$	Real coverage area of total robots at Step k
$A_{est}^{total}(k)$	Estimated coverage area of total robots at Step k
$R_a^{total}(k)$	Radius of real coverage area of total robots at Step k
$R_{a,est}^{total}(k)$	Radius of estimated coverage area of total robots at Step k
$A^i(k)$	Real coverage area of Robot i at Step k
$A_{est}^i(k)$	Estimated coverage area of Robot i at Step k
$R_{a,eff}$	Radius of effective coverage area of a single robot
T_{strait}	Length of steps compared in Case Strait
T_{normal}	Length of steps compared in Case Normal
T_{escape}	Length of moving steps in Case Strait
$p(k)$	Stop percentage of robots at Step k
P	Desired value of stop percentage of robots
T_t	Releasing time of the t^{th} releasing of base station
u_t	Number of releasing robots at the t^{th} releasing of base station
ΔR	Desired increasing radius of total coverage area
$N(k)$	Number of total robots at Step k
$D^i(k)$	Moving direction of Robot i at Step k

Table 1. Parameter Definitions of Dispersion Algorithm and Dispatch Rule

The parameters of the dispersion algorithm are defined in Table 1. These parameters are grouped as indexes, the radiuses of different ranges, the number of neighbors, the definitions of different areas and radiuses, the lengths of time steps, the stop percentage of

robots, the two releasing parameters of base station, and other parameters. All of these parameters will be explained in detail later when mentioned in this discussion.

A new range, called "*repulsion range*", is defined as shown in Figure 4. It is for designing the repulsion force for the neighbors of the robot. Moreover, the communication range excluding the repulsion range is called the "*communication ring range*," inside which its neighbors cause an attraction force to keep the network from being partitioned. Let the number of neighbors inside the communication ring range be $n_c^i(k)$, where *k* denotes the time step and *i* is the robot ID. A desired value of $n_c^i(k)$ is set as N_c. Also let the number of neighbors inside the repulsion range be $n_r^i(k)$, and the desired value of $n_r^i(k)$ is zero because of the repulsion. An example of $n_c^1(k)$ = 3 and $n_r^1(k)$ = 1 is shown in Figure 4.

The flowchart of the dispersion algorithm is shown in Figure 5. First the robot needs to decide whether it should move or not. Since it wants to keep $n_c^i(k)$ equal to N_c and $n_r^i(k)$ equal to zero, then the *moving condition* is defined as follows.

$$n_c^i(k) \neq N_c \;\|\; n_r^i(k) \neq 0 \tag{1}$$

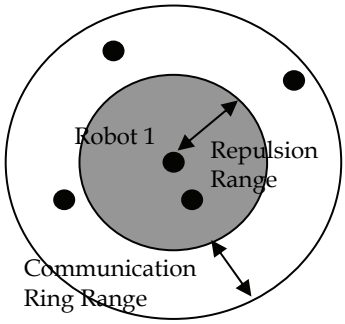


Figure 4. Schematic diagram showing the repulsion and communication ranges of a robot

However, in order to avoid the partition of the communication network, a slight adjustment is applied to the moving condition. The probability of network partition increases when the moving robots have only a few neighbors within the communication range. Therefore they should take a conservative action to avoid partition; that is, they should stop at such situation. Hence the moving condition is further adjusted as follows.

$$(n_c^i(k) \neq N_c \;\&\&\; n_c^i(k) > \frac{N_c}{2}) \;\|\; n_r^i(k) \neq 0 \tag{2}$$

For the condition that $n_c^i(k) \neq N_c$ and $n_c^i(k) > N_c/2$, it is defined as the "*Phase A*," indicating the attraction force stage. For the condition that $n_r^i(k) \neq 0$, it is defined as the "*Phase R*," indicating the repulsion force stage. Since keeping the network connected is of most important, the attraction force is considered prior to the repulsion force.

Once the moving condition is satisfied, the robot starts to move. For simplicity, the robot moving velocity is set to be the same. Hence the only one variable that the robot needs to

decide is the movement direction. Here the bacteria foraging behavior is applied for choosing the direction.

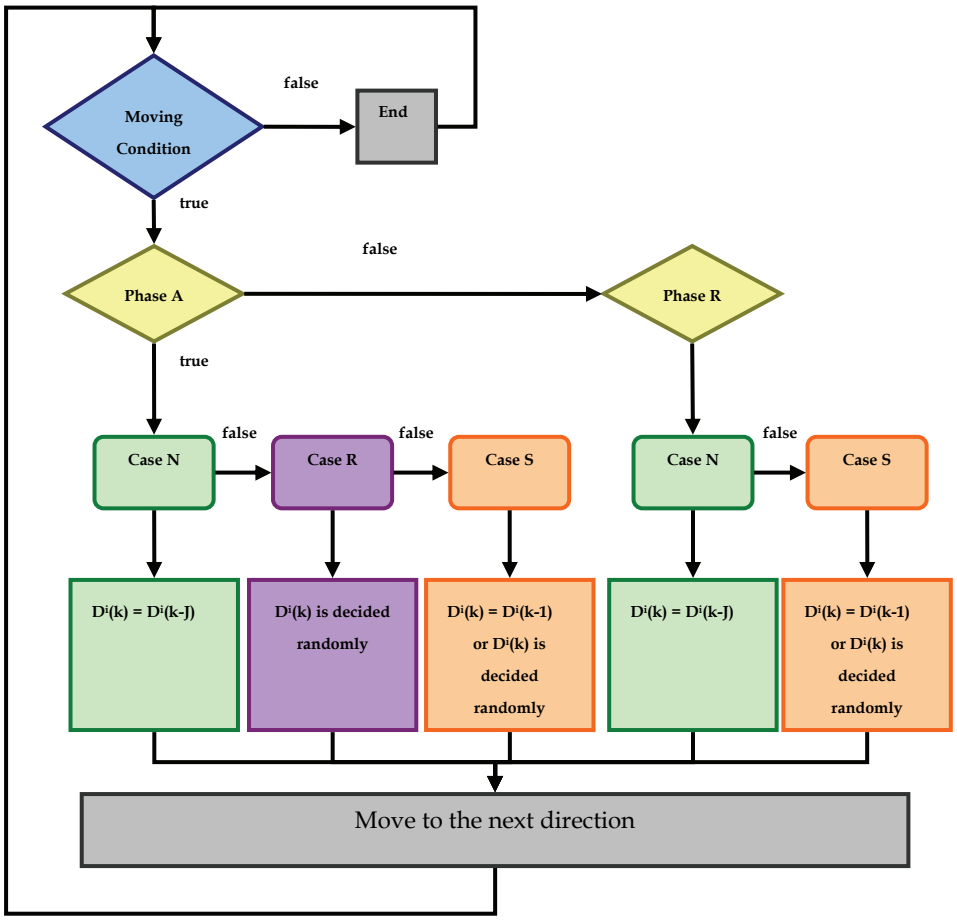


Figure 5. Flowchart of dispersion algorithm

Let the moving direction of Robot i at Step k be $D^i(k)$. First, every robot records $D^i(k)$, $n_c^i(k)$, and $n_r^i(k)$ over the last T_{strait} steps. Moreover, a small value of buffer length T_{normal} is defined. T_{strait} is set as several times of T_{normal}. According to the recorded values of $n_c^i(k)$ and $n_r^i(k)$, three cases, namely "Case Normal", "Case Strait", and "Case Random" are specified. Each case also includes Phase A and Phase R, respectively. The scenario in Phase A is discussed in the following, and minor adjustment is given for that used in Phase R.

Case Normal: The values of $n_c^i(k)$ recoded in the T_{normal} buffer are not all the same.

In Case Normal, the robot implements the behavior of climbing nutrient gradient of the E. coli bacterium. The direction of the E coli bacterium is biased toward increasing nutrient gradient. The bacterial cell compares the concentration of the environment over the past 1

time step with the concentration over the last 3 time steps before that. Hence it uses the nutrient concentration data of the past 4 time steps to decide its next movement.

$n_c^i(k)$ here is similar to the concentration of nutrition. Therefore, $n_c^i(k)$ is compared with those of the last T_{normal} steps. The direction $D^i(k-j)$ at Step *k-j* ($j \in \{1,...,T_{strait}\}$) where $n_c^i(k-j)$ is close to N_c is believed to be the direction leading $n_c^i(k)$ to N_c. So the robot chooses $D^i(k-j)$ as the next direction. That is,

$$D^i(k) = D^i(k-j) \quad j = \arg\min_{t=1\sim T_{buffer}} \left| n_c^i(k-t) - N_c \right| \tag{4}$$

An example of the recorded buffer of n_c at Step *k* is shown in Table 2. $T_{normal} = 10$ and D and n_c recorded in the last T_{normal} steps are shown in Table 2. If N_c is assumed to be 5, n_c in Step *k-4* and Step *k-9* are both closest to the desired valued of N_c. Since Step *k-4* is the later step, the direction of Step *k-4* is then adopted and the next moving direction is determined as "Forward".

Step Index	*k-1*	*k-2*	*k-3*	*k-4*	*k-5*	*k-6*	*k-7*	*k-8*	*k-9*	*k-10*
$n_c(k-j)$	4	3	4	5	6	6	7	6	5	3
$D(k-j)$	Left	Right	Left	Forward	Back	Back	Right	Left	Left	Forward

Table 2. An example of Case Normal

Case Strait: The values of $n_c^i(k)$ recoded in the T_{strait} buffer are all the same.

In Case Strait, the robot implements the escaping behavior of the E. coli bacterium from a homogeneous concentration environment. If the concentration of the environment does not change for a long time, the bacterium increases the mean run length and decreases the mean tumble time. In other words, the bacterium runs a longer distance with a fixed direction.

Once $n_c^i(k)$ recorded in the T_{strait} buffer are all the same, the robot randomly selects a direction and runs for T_{escape} steps. T_{escape} is about the order of T_{strait} and is defined by users.

An example of the recorded buffer of n_c at Step *k* is shown in Table 3. $T_{normal} = 5$, $T_{strait} = 10$, $T_{escape} = 20$ and D and n_c recorded in the last T_{strait} steps are as shown in Table 3. Since all data of n_c in the last T_{strait} steps are all the same, the robot will choose a random direction and run for T_{escape} steps. For instance, the robot may keep going forward for the next 20 steps.

Step Index	k-1	k-2	k-3	k-4	k-5	k-6	k-7	k-8	k-9	k-10
$n_c(k-j)$	4	4	4	4	4	4	4	4	4	4
$D(k-j)$	Left	Right	Left	Forward	Back	Back	Right	Left	Left	Forward

Table 3. An example of Case Strait

Case Random: The values of $n_c^i(k)$ recoded in the T_{normal} buffer are all the same, but the T_{strait} buffer are not all the same, or the robot does not have enough data of $n_c^i(k)$.

In this case, it is assumed that the robot does not have enough information about the environment. Therefore, it chooses a random direction for the next movement. The action is continued until there is enough information about the environment and the situation then changes to Case Normal or Case Strait.

An example of the recorded buffer of n_c at Step *k* is shown in Table 4. $T_{normal} = 5$, $T_{strait} = 10$, $T_{escape} = 20$ and D and n_c recorded in the last T_{strait} steps are as shown in Table 4. Since n_c are the same in the last five steps but different in the last six to last ten steps, the robot then chooses a random direction as the next moving direction. For instance, the robot may move right in the next direction and again compare n_c in the last ten steps to decide the next direction.

Step Index	k-1	k-2	k-3	k-4	k-5	k-6	k-7	k-8	k-9	k-10
$n_c(k-j)$	4	4	4	4	4	6	7	6	5	3
$D(k-j)$	Left	Right	Left	Forward	Back	Back	Right	Left	Left	Forward

Table 4: An example of Case Random

These three cases are also applied to Phase R with minor adjustment. Note that the repulsion range is smaller than the communication range. Once $n_r^i(k)$ does not change for a period of time, it is likely to be in the strait situation. Hence, Case Random is omitted in Phase R, and T_{strait} in Case Strait is changed to T_{normal}.

With this dispersion algorithm, the final coverage area of the communication network would be mainly decided by the communication range and the repulsion range. With a larger communication range, a larger coverage area is formed. Also, with a larger repulsion range, the distance between robots increases. Hence the coverage area again gets larger. Let the radius of the communication range be R_c, and the radius of the repulsion range be R_r. Also, the *radius of effective coverage area*, $R_{a,eff}$, is defined as the radius of the average coverage area. Then, $R_{a,eff}$ is a linear function of R_c and R_r, and can be expressed as follows:

$$R_{a,eff} = k_c R_c + k_r R_r \tag{5}$$

where k_c and k_r are weighting constants of R_c and R_r, respectively. Eqn. (5) can be rearranged as follows:

$$\frac{R_{a,eff}}{R_c} = k_r \frac{R_r}{R_c} + k_c \tag{6}$$

That is, the ratio of $R_{a,eff}$ and R_c and the ratio of R_r and R_c would have a linear relationship.

4. Dispatching at the Base Station

The task at the base station is to release robots into the environment for searching for unknown targets. Because of the communication and sensing limitations, the coverage area by one robot is limited. Therefore, the base station needs to add new robots to enlarge the

coverage area. Two questions decided by the base station are "when to release robots" and "how many robots to be released." For the first question, when most robots are stopped, the coverage area could approach to a limit value. Hence, the base station should release more robots when the percentage of stop robots exceeds a certain value. Assume $p(k)$ is the percentage of stopped robots at Step *k*, and P is a decision value defined by the user. Then, the releasing time T_t can be defined as follows,

$$T_t = \text{sign}(p(k) - P) \times k \tag{7}$$

where $t = \{1, 2, 3, ...\}$ is the releasing index, and

$$\text{sign}(x) = \begin{cases} 1 \ , x \geq 0 \\ 0 \ , x < 0 \end{cases} \tag{8}$$

For the second question, the goal of releasing new robots is to enlarge the coverage area. Thus, u_t, the number of additional robots at time T_t, is defined as follows:

$$u_t(k = T_t) = \frac{\Delta A_t}{\pi R_{a,eff}^2} \tag{9}$$

where ΔA_t is the desired additional area to be increased at time T_t. To determine ΔA_t, the coverage area $A^{total}(k)$ is regarded as a circle, and the desired increasing radius ΔR is also defined. Then ΔA_t is the ring area as shown in Figure 6.

However, without the position information, the total coverage area $A^{total}(k)$ is unable to compute and consequently ΔA_t is unable to determine. Hence, two methods are proposed to estimate $A^{total}(k)$, namely, (1) the feedforward estimation, and (2) the feedback estimation.

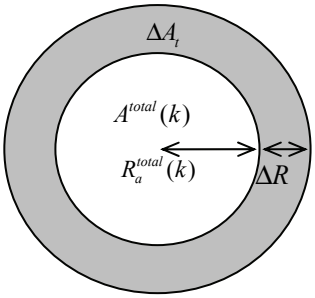


Figure 6. The diagram of ΔA_t

4.1 Area Estimation with Prior Information

Figure 7 shows the diagram of the feedforward estimation. While the percentage of stopped robots is computed, the total coverage area is estimated by the base station itself. Since most robots are stopped at the releasing times, each of them should have reached a certain coverage area. The effective coverage area of single robot is estimated by utilizing $R_{a,eff}$ and

the total coverage area is then estimated by summing these single effective coverage areas. Let $A_{est}^{total}(k)$ be the estimated total coverage area at Step k, then

$$A_{est}^{total}(k = T_t) = N(k)\pi R_{a,eff}^2 \tag{10}$$

where $N(k)$ is the number of robots at Step k. This estimate does not need any information returned by robots and is determined only with former information $R_{a,eff}$. Hence it is named as "feedforward estimation."

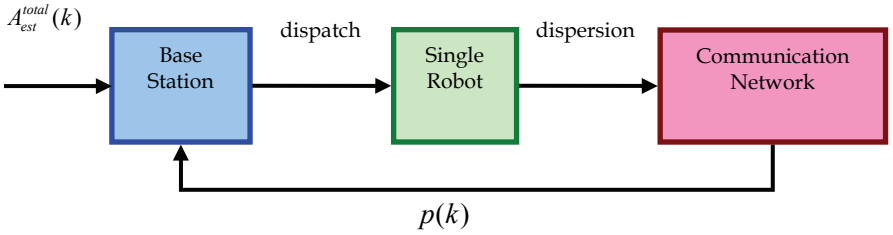


Figure 7. The flowchart of the dispatch-dispersion mechanism with feedforward estimation

4.2 Area Estimation with Feedback Information

Although the actual effective coverage area of each robot is unable to obtain, each robot can still estimate it. Figure 8 shows the diagram of the feedback estimation. Same as the feedforward estimation, the percentage of stopped robots is returned by robots. Also the effective coverage area is estimated by each robot itself and then returned to the base station. Then the total coverage area could be estimated by summing these estimated effective areas. The following estimation is then used:

$$A_{est}^{i}(k) = \frac{\pi R_c^2}{n_c^i(k)+1} \tag{11}$$

where $A_{est}^{i}(k)$ is the estimated effective coverage area of robot i at step k. Therefore the total coverage area is computed as follows:

$$A_{est}^{total}(k) = \sum_{i=1}^{N(k)} A_{est}^{i}(k) \tag{12}$$

With $A_{est}^{total}(k)$ substituted for $A^{total}(k)$, ΔA_t and u_t can then be computed.

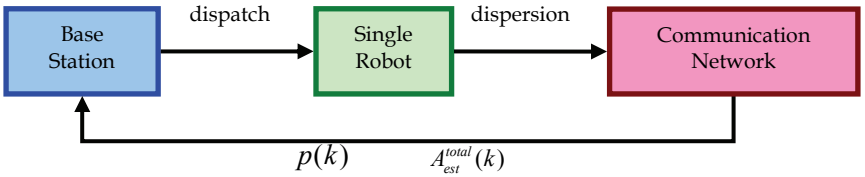


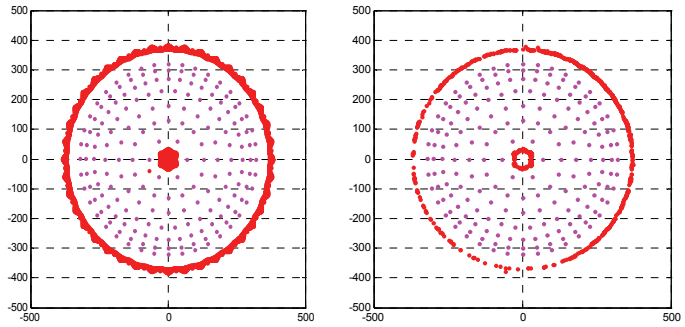
Figure 8. The flowchart of the dispatch-dispersion mechanism with feedback estimation

5. Simulation Study

In this section, several simulation studies are conducted to demonstrate the performance of the proposed movement algorithm. First, the stability of the dispersion algorithm with one single robot moving in static environment is studied. Second, the dispersion algorithm used in a multi-robot system is demonstrated and the statistics of partition rate, coverage area, spending time and stop rate are summarized. Finally, the dispatch rule at the base station is combined to execute the scenario of a target search problem.

5.1 Static Environment: Dispersion of One Robot

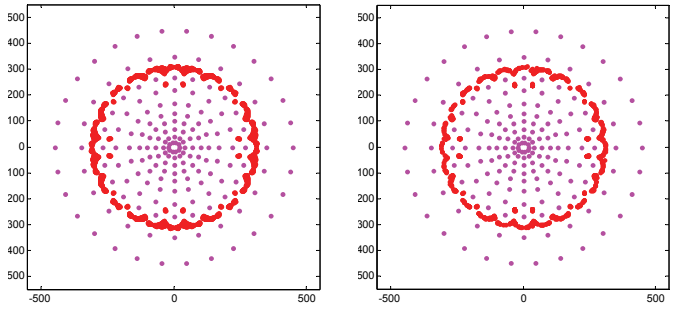
Figure 9 shows the simulation results on a static environment. In this simulation, the following parameters are used. R_c = 100, R_r = 40, N_c = 6, T_{normal} = 10, T_{strait} = 20 and T_{escape} = 50. The light (purple) dots in Figure 9 denote stationary robots. With these stationary robots, the equilibrium region that only one single moveable robot can satisfy the requirement of $n_c^i(k) = 6$ and $n_r^i(k) = 0$ are computed. The equilibrium region is denoted by the dark (red) hexagon symbols as the outer circular region and the inner circular region near the center as shown in Figure 9(a). The two similarly circular regions in Figure 9(b) are the final stop positions of one single moveable robot with the same requirement using 1500 simulations with initial positions set at (100,100) or (-100,-100). From the two plots, it can be observed that the two set of dark (red) regions are almost the same. The reason that the inner part of the hexagon symbols of Figure 9(a) is missing in Figure 9(b) is that the robot stops right as the requirements are satisfied. Hence, no robots stop inside the inner circle. Therefore, the dispersion algorithm indeed leads the robot to the regions that satisfy the requirement.



(a) (b)

Figure 9. Static environment simulation. (a) The analytic region of equilibrium points. (b) The simulation result of stop region

Figure 10 is another static environment simulation result. All the parameters are the same as the previous example. The dark (red) circular region in Figure 10(a) is the analytic regions of equilibrium points that satisfy the requirements. It is an open region which is slightly different from the regions in Figure 9(a). In Figure 10(b) the dark (red) circular region is the stop positions of one single moveable robot in 1000 simulations. The dark (red) region is almost similar to the one in Figure 10(a). Moreover, although the equilibrium region is open, the robot still stops at the right positions and does not wander to the faraway positions. Hence the algorithm indeed leads the robot to the desired position.



(a) (b)

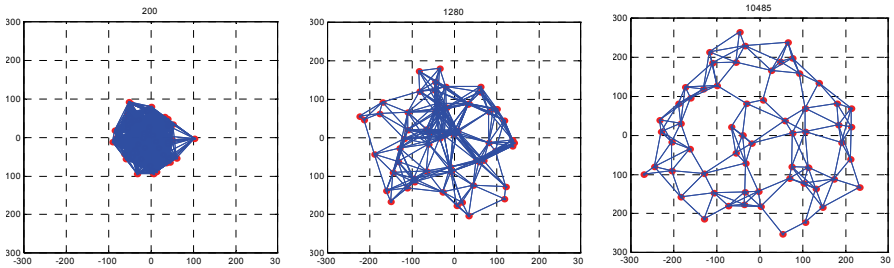
Figure 10. Static environment simulation. (a) The analytic region of equilibrium points. (b) The simulation result of stop region

5.2 Multi-Robot System: Dispersion of n Robots

Figure 11 shows the dispersion progress of a group of 60 robots. In this simulation, the following parameters are used: N = 60, R_c = 100, R_r = 30, N_c = 6, T_{normal} = 10, T_{strait} = 20 and T_{escape} = 50. The final balanced distribution forms a good communication network. It can be seen that, due to the repulsion force of Phase R, there are no robots staying too close and hence the coverage area has been enlarged to a certain value. Related statistic analysis of the dispersion algorithm is discussed in the following.

5.2.1 Partition Rate

The zero desired value of $n_r^i(k)$ provides a repulsion force to robots. If R_r is set too large, the network partition is very likely to happen. But with a small R_r, the repulsion mechanism would not be obvious. It is important to choose an appropriate value of R_r.



(a) (b) (c)

Figure 11. The dispersion of 60 robots. (a) Step 200: The dispersion has just begun, and the robots still stay close. (b) Step 1280: The network has dispersed obviously. (c) Stop 10485: The final result of the dispersion. The robots form a dispersed communication network

Figure 12 shows the relation between the partition rate and R_r / R_c. From this figure, it can be seen that, when R_r is below about 0.4 times R_c, the partition hardly happens. However,

when R_r is set too large to about 0.6 times R_c, the probability of partition is almost equal to 100%. Hence, the best value of R_r is set to be about 0.4 times R_c.

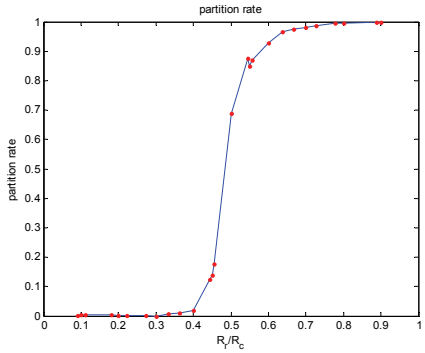


Figure 12. The partition rate

5.2.2 Coverage Area/Effective Area Radius

In addition to the partition rate, the value of R_r / R_c also affects the effective coverage area radius $R_{a,eff}$. As shown in Eqn. (6), R_r / R_c and $R_{a,eff} / R_c$ should have a linear relationship.

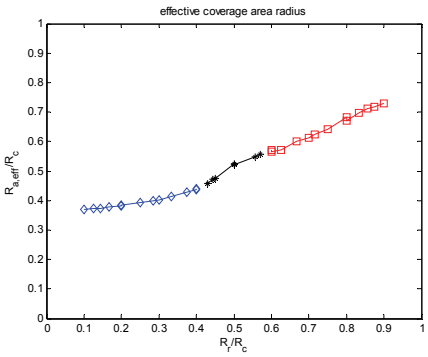


Figure 13. The effective area radius

Figure 13 is a statistical result of 2000 simulations. The final average coverage area of different values of R_c, R_r and N are computed. The slope of the average area and N with fixed R_c and R_r is considered as the effective coverage area of a single robot, and then $R_{a,eff}$ is derived. In Figure 13, it can be seen that, when R_r / R_c is below about 0.4, R_r / R_c and $R_{a,eff} / R_c$ indeed have a perfect linear relationship. Moreover, they have another linear relationship when R_r / R_c exceeds about 0.6. The two regions of R_r / R_c that R_r / R_c and $R_{a,eff} / R_c$ have good linear relationships are just the regions that the partition rate is almost equal to 0 or 100% as shown in Figure 9. This indicates that the linear relationship of R_c, R_r and $R_{a,eff}$ exists for both that the communication network is not partitioned, and that the

communication network is completely partitioned. In this case, k_r is 0.228 and k_c is 0.340 when R_r / R_c is below 0.4. These values indicate that the weighting of attraction is larger than that of repulsion since the network is pulled together by the attraction. When R_r / R_c is above 0.6, k_r is 0.563 and k_c is 0.225. These values indicate that the weighting of repulsion is larger than that of attraction since the network is partitioned due to the repulsion force.

5.2.3 Spending Time

The spending time is defined as the time when all robots are stopped. Figure 14 shows the relationship of the spending time and the number of robots with R_c = 100 and R_r = 10, ..., 40. From the figure, it can be observed that the spending time is roughly proportional to the number of robots. Moreover, the slope is only determined by R_r. Figure 15 shows the value of slopes with different R_r and R_c. It can be seen that even with different R_c, the slopes of the spending time and the number of robots are almost the same as long as R_r is the same. This statistics indicates that the time is mainly spent on repulsing too-close neighbors.

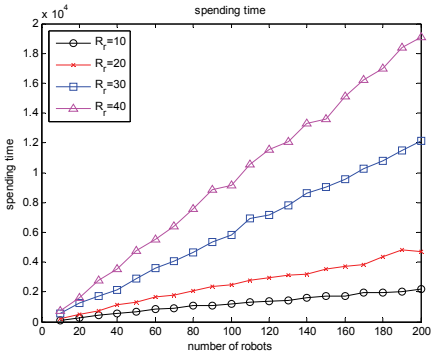


Figure 14. The spending time. With fixed R_r and R_c the spending time is roughly proportional to the number of robots

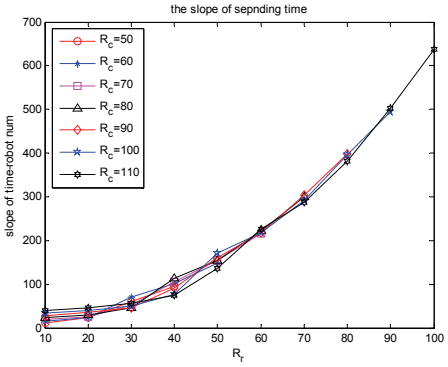


Figure 15. The slope of spending time and number of robots which is determined only by R_r

5.2.4 Stop Rate

The average spending times are also recorded when the stop rate exceeds 10%, 20%, …, 100%. The result is shown in Figure 16. The vertical axis is the percentage of the time spent, and the horizontal axis is the stop rate. It can be seen that these two ratios have a roughly exponential relationship regardless of the values of R_r and R_c. This combined with the spending time would be useful information for estimating the time when the stop rate exceeds a certain value. Later this information is used in the dispatch rule.

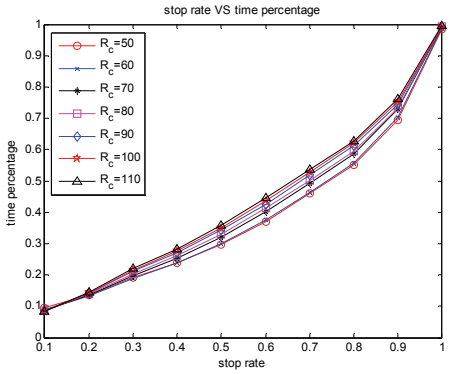


Figure 16. Stop rate and time percentage. They have a rough exponential relationship regardless of R_r and R_c

5.3 Target Search: Dispersion and Dispatch

In this subsection, the dispatch rule is combined with the dispersion algorithm to execute the target search problem. Figure 14 shows the progress of a target search problem utilizing the feedforward estimation dispatch. The initial number of robots is 10. R_c, R_r, N_c, T_{normal}, T_{strait} and T_{escape} are set as the values in previous subsection. ΔR = 50, P = 0.8 and the target position is at (200, 200). k_r and k_c are set as 0.228 and 0.340, respectively.

New robots are released form the center of the plane when $p(t)$ exceeds P, as shown in Figure 17(b), (d), and (f). The target is found after the three dispatches as shown in Figure 17(g).

The objective of the dispatch at the base station is to enlarge the coverage area timely. Figure 18 shows the coverage area versus the time of two simulations. The longer (red) line is a simulation result of feedforward estimation, and the shorter (blue) line is a simulation result of feedback estimation. It can be seen that the coverage area increases almost linearly with time, which indicates that by using the dispatch rule the base indeed releases appropriate number of robots at right time.

Moreover, the final spending time when the stop rate $p(t)$ reaches P can be estimated in advance with the simulation statistics. Hence another dispatch rule is studied where the coverage area and the stop rate of robots are both estimated by the base station. The flowchart is shown in Figure 19. Figure 20 is a simulation result of applying the statistics of the spending time and the stop rate. It can be seen that the coverage area increases roughly

linear with time as well as the two former estimation methods. Hence this estimation provides a good performance. Moreover, no information returned by robots is needed, therefore the base station can even determine u_k and T_t before the task starts. This is a very beneficial advantage.

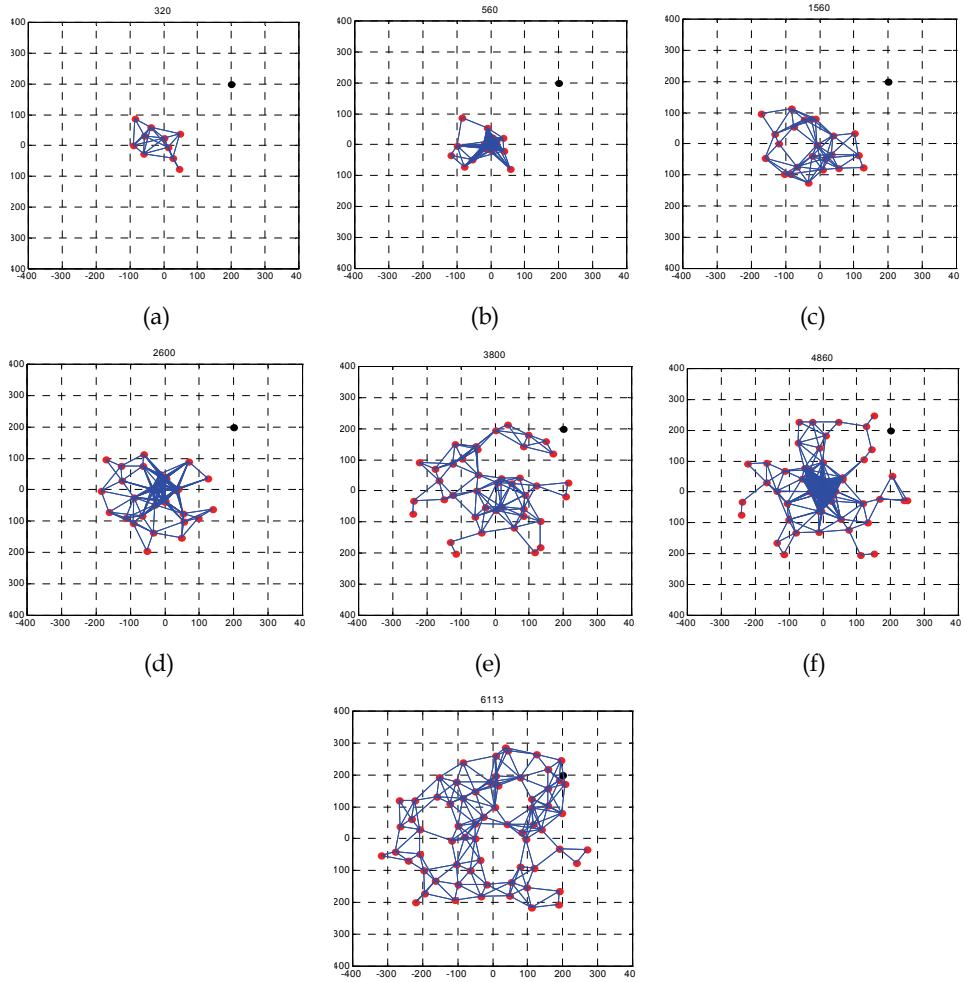


Figure 17. The progress of target search. (a) Step 320: Initially, the network formed by 10 robots which is too small to cover the target. (b) Step 560: After the stop rate reaches 0.8, new robots are released. (c) Step 1560: The network is enlarged but still unable to cover the target. (d) Step 2600: The second releasing. (e) Step 3800: The network is enlarged again. (f) Step 4860: The third releasing. (f) Step 6113: After 3 times of releasing, the target is successfully found by the enlarged network

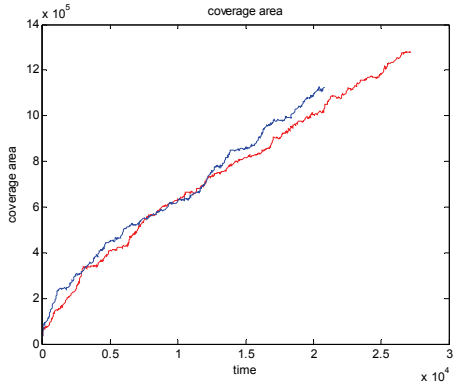


Figure 18. Coverage area versus time. The coverage area increases roughly linear with time

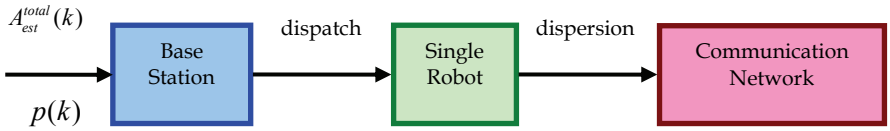


Figure 19. Flowchart of the dispatch rule utilizing area and stop percentage estimation with prior information

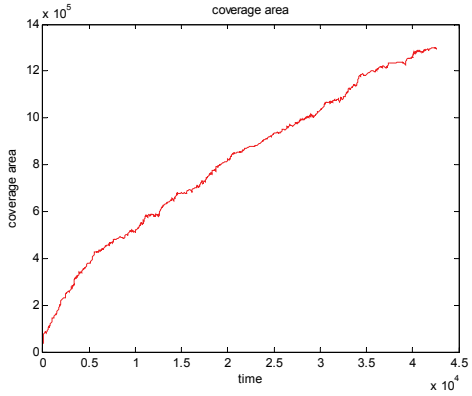


Figure 20. Coverage area versus time

5.3 Estimation Accuracy of Dispatch Rule

In the target search task, the base station decides the releasing time and the number of released robots by estimating the coverage area and the stop percentage of robots with three kinds of methods, namely the "area estimation with prior information", "area estimation with feedback information" and "area and stop percentage estimation with prior information." For these three estimations, the accuracy is one of major concerns. Figure 21 and Figure 22 show the estimation accuracy of area and stop percentage of the three

methods. The following parameters are used: R_c = 100, R_r = 40, N_c = 6, T_{normal} = 10, T_{strait} = 20 and T_{escape} = 50, and P is set as 0.8 and the initial number of robots is set to 10. Robots are released form the origin of the plane, and the target is set at (350, 350). 50 simulations for each estimation method are done to compute the average values. The statistics of spending time, stop percentage and coverage area presented in Section 5.2 are used in the dispatching. Figure 21(a) and Figure 21(b) shows the ratio of the estimated area and the real area at each releasing time with "area estimation with prior information" and "area estimation with feedback information," respectively. From the figures it can be observed that the estimation accuracy becomes better as the time increases. The two results of estimation accuracy are similar to each other, but the one with prior information has better accuracy in the beginning. Moreover, it remains a value between 0.9 and 1 in the later period while the other one may exceed 1.

Figure 21(c) and Figure 22 show the estimation accuracy of "area and stop percentage estimate with prior information." Compared with Figure 21(a) and Figure 21(b), Figure 21(c) shows that the accuracy of area estimation is a little worse than the one of the former two estimations but still remain a value larger than 0.5. And for the estimation of stop percentage, the estimation value is between 0.6 and 0.95, which is about the range of 0.8 ± 0.2. Hence the estimation of stop percentage shows a good result.

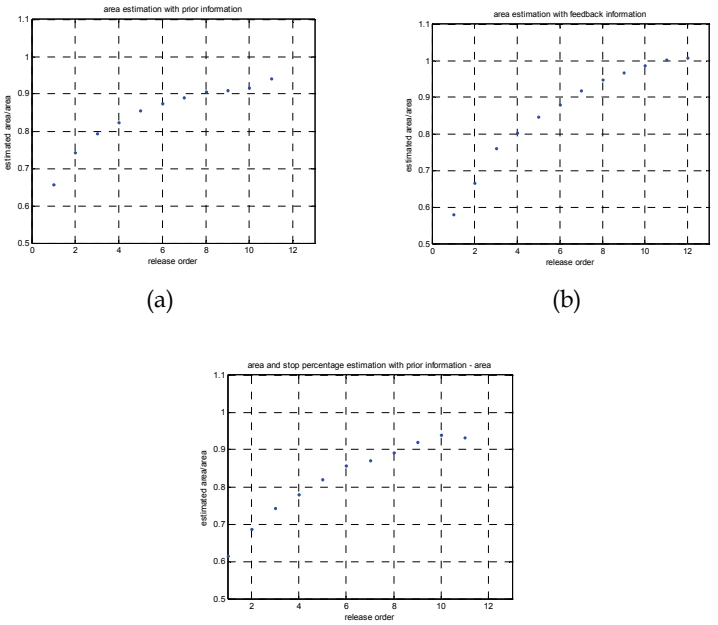


Figure 21. Ratio of estimated area and real area at releasing times with (a) area estimation with prior information; (b) area estimation with feedback information; (c) area and stop percentage estimation with prior information

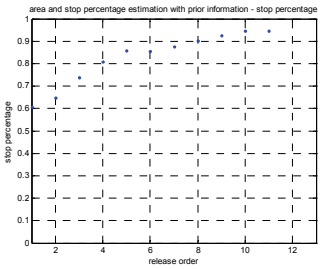


Figure 22. Estimated stop percentage at releasing times with "Area and stop percentage estimation with prior information"

6. Conclusions and Future Work

In this chapter, a dispersion movement algorithm for multi-robot systems with simple computation and easily obtainable information is proposed. The only information needed is the communication density, i.e., the number of communication links of each individual robot. In addition, a dispatch control rule is proposed based on the dispersion algorithm. With some parameters known in advance, the base station could then estimate an appropriate time to release new robots. The dispersion and dispatch control rules are easy to implement for a practical multi-robot system to act like a natural creature system. The dispersion movement algorithm itself still executes as a natural system. And with the dispatch rule, the dispersion algorithm can be used in tasks with more variety. Simulation results of the dispersion and dispatch control rules are presented, and statistics of the coverage area, partition rate, spending time and stop rate show the advantage of these algorithms. In the future, the research will focus on the mechanism of adaptively adjusting of the dispatch control rule, and the theoretical analysis of the algorithm performance. The implementation of the algorithm on practical robots and further applications are also under planning.

7. Acknowledgement

This work was supported in part by the National Science Council, Taiwan, ROC, under the grants: NSC 95-2221-E-002-303-MY3, and NSC 96-2218-E-002-030, and by DOIT/TDPA: 95-EC-17-A-04-S1-054.

8. References

Blough, D.M.; Leoncini, M.; Resta, G. & Santi, P. (2003) The K-Neigh Protocol for Symmetric Topology Control in Ad Hoc Networks, *Proceedings of ACM MobiHoc 2003*, pp. 141-152, ISBN: 1-58113-684-6, Jun. 2003, Association for Computing Machinery, Annapolis, MD, USA

Burgard, W.; Moors, M.; Stachniss, C. & Schneider, F.E. (2005) Coordinated Multi-Agent Exploration, *IEEE Transactions on Robotics*, Vol. 21, No. 3, (Jun. 2005) pp. 376-386, ISSN: 1552–3098

Chellappan, S.; Bai, X.; Ma, B.; Xuan, D. & Xu, C. (2007) Mobility Limited Flip-Based Sensor Networks Deployment, *IEEE Transactions on Parallel and Distributed Systems*, Vol. 18, No. 2, (Feb. 2007) pp. 199-211, ISSN: 1045-9219

Cortes, J.; Martinez, S.; Karatas, T. & Bullo, F. (2004) Coverage Control for Mobile Sensing Networks, *IEEE Transactions Robotics and Automation*, Vol. 20, No. 2, (Apr. 2004) pp. 243-255, ISSN: 1042-296X

Czirok, A.; Ben-Jacob, E.; Cohen, I. & Vicsek, T. (1996) Formation of complex bacterial colonies via self-generated vortices, *Physical Review E*, Vol, 54, No. 2, (Aug. 1996) pp. 1971-1801, ISSN: 1063-651X

Flierl, G.; Grunbaum, D.; Levin, S. & Olson, D. (1999) From Individuals to Aggregations: the Interplay between Behavior and Physics, *Journal of Theoretical Biology*, Vol. 196, No. 4, (Feb. 1999) pp. 397-525, ISSN: 9911-4082

Gazi, V. & Passino, K.M. (2003) Stability Analysis of Swarms, *IEEE Transactions on Automatic Control*, Vol. 48, No. 4, (Apr. 2003) pp. 692-697, ISSN: 0018-9286

Gazi, V. & Passino, K.M. (2004) A class of attractions/repulsion functions for stable swarm aggregations, *International Journal of Control*, Vol. 77, No. 18, (Dec. 2004) pp. 1567-1579, ISBN: ISSN: 0020–7179

Gueron, S.; Levin, S.A. & Rubenstein, D. I. (1996) The Dynamics of Herds: From Individuals to Aggregations, *Journal of Theoretical Biology*, Vol. 182, No. 1, (Sep. 1996) pp. 85-98, ISSN: 9911-4082

Heo, N. & Varshney, P.K. (2005) Energy-Efficient Deployment of Intelligent Mobile Sensor Networks, *IEEE Transactions on Systems, Man, and Cybernetics—Part A: Systems and Humans*, Vol. 35, No. 1, (Jan. 2005) pp. 78-92,. ISSN: 1083-4427

Li, L.; Halpern, J.Y.; Bahl, P.; Wang, Y. & Wattenhofer, R. (2005) A Cone-Based Distributed Topology Control Algorithm for Wireless Multi-Hop Networks, *IEEE/ACM Transactions on Networking*, Vol. 13, No. 1, (Feb. 2005) pp. 147-159, ISSN: 1063-6692

Parrish, J.K.; Viscido, S.V. & Grunbaum, D. (2002) Self-Organized Fish Schools: An Extraction of Emergent Properties, *The Biological Bulletin*, Vol. 202, No. 3, (June 2002) pp. 296-305, ISSN: 0006-3185

Passino, K.M. (2002) Biomimicry of Bacterial Foraging for Distributed Optimization and Control, *IEEE Control Systems Magazine*, Vol.22, No.3, (June 2002) pp. 52-67, ISSN: 0272-1708

Poduri, S. and Sukhatme, G.S. (2004) Constrained Coverage for Mobile Sensor Networks, *Proceedings of IEEE 2004 International Conference on Robotics and Automation*, pp. 165-171, ISBN: 0-7803-8232-3, Apr. 2004, Institute of Electrical and Electronics Engineers, New Orieans, LA, USA

Rodoplu, V. & Meng, T.H. (1999) Minimum Energy Mobile Wireless Networks, *IEEE Journal on Selected Areas in Communications*, Vol. 17, No. 8, (Aug. 1999) pp. 1333-1344, ISSN: 0733–8716

Santi, P. (2005) *Topology Control in Wireless Ad Hoc and Sensor Networks*, John Wiley & Sons, ISBN: 978-0470094532, Hoboken, NJ, USA

Tan, J. (2005) A Scalable Graph Model and Coordination Algorithms for Multi-agent Systems, *Proceedings of the 2005 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, pp. 1529-1534, ISBN: 0-7803-9046-6, Jul. 2005, Institute of Electrical and Electronics Engineers, Monterey, CA, USA

7

Spatiotemporal MCA Approach for the Motion Coordination of Heterogeneous MRS

Fabio M. Marchese
Università degli Studi di Milano-Bicocca
Italy

1. Introduction

In this work, a collision-free motion-planning technique for multiple robots based on Multilayered Cellular Automata (MCA) has been studied. What we want to realize is a Coordinator for multi-robot systems which decides the motions of a team of robots while they interact with the environment and react as fast as possible to the dynamical events.
Many authors have proposed different solutions for the Path/Motion Planning problem during the last twenty-five years for single and multiple robots. For example, a solution based on a geometrical description of the environment had been proposed since 1979 (e.g., Lozano-Pérez & Wesley, 1979, Lozano-Pérez, 1983). The motion-planners (path-planners) working on these types of models generate very precise optimal trajectories and they can solve really difficult problems, also taking into account non-holonomic constraints, but they are also very time consuming. To face a real dynamical world with many events, a robot must constantly sense the world and re-plan as fast as possible, according to the newly acquired information. Other authors have developed alternative approaches less precise but more efficient: the Artificial Potential Fields Methods.
In the eighties, Khatib first proposed this method for the real-time collision avoidance problem of a manipulator in a continuous space (Khatib, 1986). Jahanbin and Fallside first introduced a wave propagation algorithm in the Configuration Space C-Space on discrete maps (*Distance Transform,* Jahanbin & Fallside, 1988). In (Barraquand et al., 1992), the authors used the Numerical Potential Field Technique on the C-Space to build a generalized Voronoi Diagram. Zelinsky extended the *Distance Transform* to the *Path Transform* (Zelinsky, 1994). Tzionas et al. in (Tzionas et al., 1997) described an algorithm for a diamond-shaped holonomic robot in a static environment where they let a CA build a Voronoi Diagram. In (Warren, 1990) the coordination of robots is proposed using a discretized 3D C-Space-Time (2D workspace plus Time) for robots with the same shape (only square and circle) and a quite simple kinematics (translation only). LaValle in (LaValle, 1998) applies the concepts of the Game Theory and multi-objective optimization to the centralized and decoupled planning. A solution in the C-Space-Time is proposed in (Bennewitz, 2001), where the authors use a decoupled and prioritized path planning in which they repeatedly reorder the robots to try to find a solution. It can be proven that these approaches are not complete.
In this work, we want to design a motion coordinator for a set of heterogeneous mobile robot (different shapes and kinematics), able to determine the motions of the mobile agents

in order to avoid collisions with obstacles and with other robots. We have adopted Cellular Automata as formalism for merging a grid model of the world (Occupancy Grid) with the C-Space-Time of multiple robots and Numerical (Artificial) Potential Field Methods, with the purpose to give a simple and fast solution for the motion-planning problem for multiple mobile robots, in particular for robots with different shapes and kinematics. This method uses a directional (anisotropic) propagation of distance values between adjacent automata to build a potential hypersurface embedded in a 5D space. Applying a constrained version of the descending gradient on the hypersurface, it is possible to find out all the admissible, equivalent and shortest (for a given metric of the discretized space) trajectories connecting two poses for each robot C-Space-Time.

2. Context: a M^{n}RS self-similar layered architecture

The area of the cooperative/distributed robotics moved its first steps in the 80's, and since then it has received ever increasing attention, especially in the last decade, involving many application domains. A MultiRobot System is characterized by a set of robots working in the same environment, interacting between them inside the system and toward the outside of the system with the external environment, and last but not least, sharing their resources to achieve a general common task. With respect to an equivalent single robot achieving the same task, a MRS improves the robustness and the reliability thanks to its modularization. Many are the application domains where MRSs are involved: from service robotics to planets exploration. In (Dudek et al., 1996) an analysis and a classification of the typical tasks for MRS have been proposed. Taxonomy of robots collectives is based on the following main dimensions: size, composition, communication, re-configurability and computation. In the editorial (Arai et al., 2002) the authors identify seven important topics of the MRS research area: biological inspirations, communication, architectures and control, localization/mapping/exploration, object transport and manipulation, motion coordination, and reconfigurable robots. A survey of the area has been proposed more recently in (Farinelli et al., 2004). In this paper, an interesting classification about MRS architectures is described. It is mainly based on two primary types of features: coordination dimensions and system dimensions. The first is a layered taxonomy structured on four levels: Cooperation level, Knowledge level, Coordination level and Organization level. The system dimensions are subdivided into four groups: Communication, Team composition, System architecture, Team size.

On the basis of this taxonomy, we can give a classification for our coordination architecture: cooperative (Cooperation level); unaware (Knowledge level); strongly coordinated (Coordination level); strongly centralized (Organization level). On the standpoint of the system dimensions: direct communication, throughout a central dispatcher; heterogeneous (Team composition); deliberative (System architecture); small-medium Team size. The framework can be intended as *cooperative* because the robots reaching their goals accomplish a single global task specified at a higher level of abstraction. At the *Organization level*, it is strongly centralized: there is a leader robot (or an external supervisor, but it is only a deployment consideration) that is omniscient with respect to its team mates, and which organizes, synchronizes and controls the other robots (*strongly coordinate*). The other team mates do not have any knowledge about each other (*unaware*). In a strongly centralized architecture, the central unit is responsible for taking any decision, and the peripheral units operate consequently. This must not be intended as a severe restriction to the autonomy of

the single unit: every robot can decide how to realize the goal/command that the leader provides for it. Besides this, the autonomy restriction will also depend on the type of commands: for example, the central unit can provide a gross trajectory, which will be refined by the single agent. *Heterogeneity*: the task is to design an architecture operating independently from the robot characteristics.

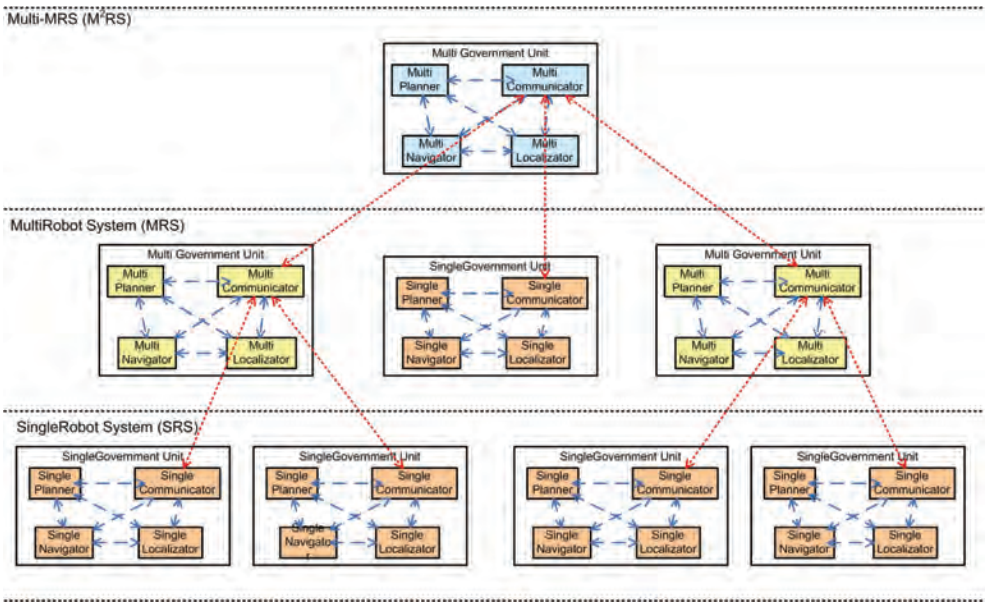


Figure 1. An example of Multi-MRS at the second level

The proposed approach is coherent with a MRS architecture (briefly described in this paragraph), where the coordination system is just a subpart of it. It is basically organized into self-similar objects, where essentially the MRS is reified as an object similar to the single robot composing it. The architecture can be further extended to the coordination of groups of teams (let's say Multi-MRS or M^2RS) extending the number of layers, but not changing the significance: each M^nRS is an object similar to all the other objects at the lower levels and to the single robots. Each level is an aggregation of entities of a lower order, till a single robot system {M^0RS, MRS, M^2RS, … $M^{n-1}RS$}, where M^0RS = SRS (Single-Robot System).

The framework has to be able to control a group of robots with quite different hardware and software devices. In particular, from the point of view of the motion control, the robots are quite different in sizes, shapes and kinematics.

Our task is to define a framework able to work independently from the single robot hardware and software characteristics. We have designed a layered architecture (an example in Fig. 1), called Multi-Robot Layered architecture (MRL architecture). The two lower layers concerns to the multi-robot level (MRS) and the single-robot level (SRS). It is interesting to note that the entities inside the layers have the same structure and expose similar functionalities/modules. In this context, we are considering the multirobot as a single (abstract) separated entity which performs its own functionalities and to which the single robots are interfaced throughout a communication system. From the conceptual point

of view, the multirobot has similar functionalities as the single robot, for example it navigates, communicates, and so on, as the single robot does. The differences arise at the implementation/deployment level.

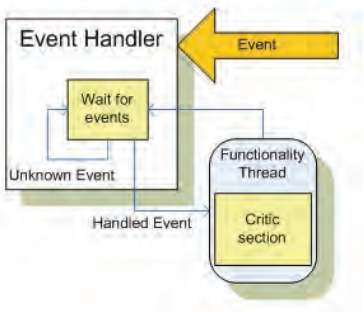


Figure 2. The core of a module: the event-handler

The concept can be abstracted further to higher level where a MnRS is seen as an entity with the same role of the others, thus realizing a self-similar structure, with similar modules which can interact with each other even at different levels, as shown in Fig. 1. The architecture has been thought as a concurrent event driven system. The modules inside each entity (single or multi) are synchronized with the other modules using events. The basic structure of a module is essentially a thread controlling an event-handler (Fig. 2). The main task of the thread is to wait for a set of events. When one of the events is signaled, it triggers a separate sub-thread that realizes the corresponding functionality and immediately returns to wait for other events. Unexpected events are simply ignored. In this way, we realize a simple ad efficient mechanism that it is not blocked during the execution of a functionality, and thus it is independent to how much it is complex and how long it takes to be completed. Therefore, every event receives an immediate response: an important characteristic for a real-time system that permits to handle fast dynamics of the environment.
Besides architectural issues, which are mainly concerned with the design and then the handling of multi-threads/concurrent systems (a more detailed description in (Marchese, 2007)), the real-time coordination of the motion of n bodies is the major problem.

3. The Multirobot Motion-Planner Coordination System

3.1 Problem statement: from Motion-Planning to Spatiotemporal MCA

The Multirobot Motion Planner is "just" a module (*Multi-Planner*) inside the entities at the MRS level above described.
To solve the Motion-Planning Problem we need some essential features: a world representation, one or more motion models of the robots, a substrate on which the two previous entities interact to generate a (multi-)plan.
There is a very large variety of world models in literature able to describe the interaction between autonomous agents and their environment. A very well-known model and probably the most important one is the Configuration Space (Lozano-Pérez, 1983; Latombe, 1991). Let us consider a simple rigid body $\mathcal{R}$ with a generic shape, a finite extension and an orientation (a preferential direction of movement). The C-Space $\mathcal{C}$ of a rigid body is the set of all its configurations $\mathbf{q}$ (i.e. all its poses). $\mathcal{R}$ moves in a workspace $\mathcal{W}$

embedded in a physical n-dimensional space (we do not necessarily consider the Euclidean Space $\mathcal{R}^n$: also n-D manifold are admissible (Latombe, 1991)). In the workspace, a finite number of obstacles $\mathcal{O}_i$ are placed. A configuration $\mathbf{q}$ of $\mathcal{R}$ is a compact representation of all the points of its shape. Because the robot is a rigid body, any configuration $\mathbf{q}$ corresponds to a set of n independent parameters, corresponding to the n DOF of the robot. If the robot can freely translate and rotate on a 2D surface, then its C-Space is a 3D manifold $\mathbb{R}^2 \times \mathbb{S}^1$, where $\mathbb{S}^1$ is the unit circle $\mathbf{SO}(2)$. $\mathcal{R}(\mathbf{q})$ is the set of points occupied by the robot in $\mathcal{W}$ at the configuration $\mathbf{q}$. Every obstacle $\mathcal{O}_i$ is mapped in the C-Space as a set of configuration $\mathcal{CO}_i$ (*C-Obstacle*): $\mathcal{CO}_i = \{\mathbf{q} \in \mathcal{C} \,/\, \mathcal{R}(\mathbf{q}) \cap \mathcal{O}_i \neq 0\}$. In other words, a C-Obstacle is the set of configuration $\mathbf{q}$ at which the robot collides with the obstacle. The free space is the subset of points of $\mathcal{C}$ where the robot and the obstacles do not have any intersection: $\mathcal{C}_{free} = \mathcal{C} - \cup_i \mathcal{CO}_i$. A collision-free path between two configuration $\mathbf{q}_S$ (starting pose) and $\mathbf{q}_G$ (goal pose) is any continuous map $\tau : [0, 1] \rightarrow \mathcal{C}_{free}$ with $t(0) = \mathbf{q}_S$ and $t(1) = \mathbf{q}_G$.

Using a regular decomposition in (hyper-)cells, the C-Space can be easily represented using a n-D bitmap $\mathcal{GC}$ (*C-Space bitmap*). The *C-Potential* is a function $\mathbf{U}(\mathbf{q})$ defined over the C-Space that *drives* the robot through the sequence of configuration points to reach the goal pose (Barraquand et al., 1992). These definitions can be extended to an arbitrary number of robots with an arbitrary number of DOF working in the same space (LaValle & Hutchinson, 1998).

Another feature needed to plan is the substrate where to make the environment and the robots models to interact. We use Cellular Automata (CA) for this purpose. CA are automata distributed over the cells of a Cellular Space $\mathbb{Z}^n$ (a regular lattice) with transition functions invariant under translation (Goles & Martinez, 1990): $f_c(\cdot) = f(\cdot)$, $\forall\, c \in \mathbb{Z}^n$, where $f(\cdot)$: $\mathbf{S}^{|\mathbf{A}_0|} \rightarrow \mathbf{S}$, where c is the coordinate vector identifying a cell, $\mathbf{S}$ is the set of states of a FSM, $\mathbf{A}_0$ is the set of arcs outgoing from a cell towards the neighbors and $f(\cdot)$ is the transition function. This definition introduces an n-D grid of hyper-cells connected to the surrounding cells with the arcs $\mathbf{A}_0$. In each cell it is embedded an automaton, and the automaton of a cell is equal to the automaton of any other cell. The state $s \in \mathbf{S}$ of the automaton evolves (function $f(\cdot)$) on the basis of the states of the neighbors (arcs $\mathbf{A}_0$). The global behavior of the whole cellular automata is influenced by the local rules (transition function) and the updating rules, i.e., it is governed by the chronological sequence of cells states updating (e.g. Synchronous, Asynchronous, Block-Sequential, etc.).

It is possible to organize the CA in layers of homogenous automata, but with different transition functions across the layers, thus having a Multilayered Cellular Automata (MCA). The mapping between the Robot Motion-Planning Problem and MCA is quite simple: every "pixel" of the C-Space bitmap $\mathcal{GC}$, corresponding to a configuration $\mathbf{q}$, is a hyper-cell of an n-D CA. The state inside the cell represents a potential value and it contributes to build the *C-Potential* $\mathbf{U}(\mathbf{q})$ through a diffusion mechanism over the neighbors. The *C*-Potential is a potential hyper-surface defined on the n-D space (it is embedded in an n+1-D space). The trajectories are found following the minimum valley of this hyper-surface.

Extending the concept, we associate a vector of attributes (a state vector instead of a single state) to every cell. Each state vector depends on the state vectors of the cells in the neighborhood. An alternative interpretation is the following: this is a Multilayered Cellular Automaton, where each layer deals with a subset of components of the state vector. Each subset is evaluated in a single layer and it depends on the same attribute of the neighbor

cells in the same layer, but it also depends on the values of the corresponding cells in other layers. In the following sections, we are going to describe the layers and their behaviors.

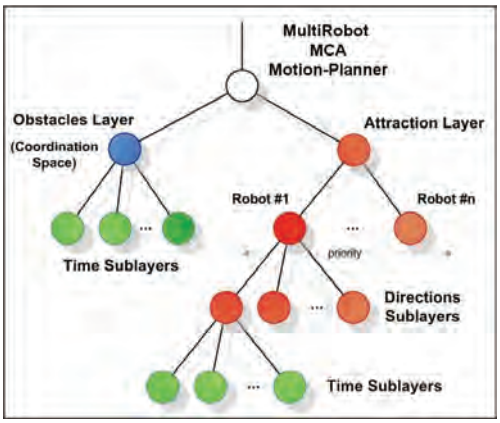


Figure 3. MultiLayered Motion-Planner Architecture

3.2 Multilayered/Multidimensional Architecture

All the planning system is based on a layered architecture, as shown in Fig. 3, where the layers and sub-layers structure and their dependencies are defined.

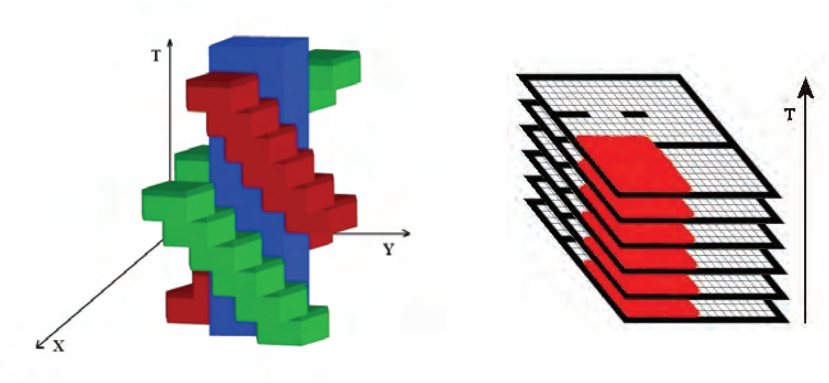
There are two main layers: the *Obstacles Layer* and the *Attraction Layer*. Each layer is subdivided into more sub-layers: the *Obstacles L.* has 3 dimensions (2 for the workspace (X, Y) and 1 for the time), while the *Attraction L.* has up to 5 dimensions (1 for the robots, 3 for the robots C-Spaces (X, Y, θ) and 1 for the time). The *Obstacles L.* conceptually depends on the outside environment by means of an obstacle detector or a world modeler. Its sub-layers should react to the "external" changes: the changes of the environment, i.e. the movements of the obstacles in a dynamical world. Through a sensorial system (not described here), these changes are detected and the information is stored in *Obstacles L.* permitting the planner to re-plan as needed.

3.3 The Obstacles Layer

The main role of the *Obstacles L.* is to map the obstacles of the environment and to dynamically create a repulsive force that keeps the robots away from them.

Up to now, only static obstacles are considered (e.g. walls). For the single robot, the other robots are seen as moving obstacles, with unknown and unpredictable motions. We are considering a centralized motion-planner/coordinator, which can decide the timed trajectories of all the supervised robots. Thanks to this information, the planner considers the silhouette of a robot as an obstacle for the other robots. In this work, we introduce a discretized version of the C-Space-Time (Erdmann & Lozano-Pérez, 1986; Warren, 1990).

With the introduction of the Time axis, a static obstacle becomes a hyper-obstacle extended along the time direction in the Spacetime 4D space. In Fig. 4.a a discretized Spacetime is shown, where the blue object is a static obstacle extended vertically along the time. The robots moving in the environment become "static" hyper-obstacles in the Spacetime, having different poses in each time slice (red and green objects in Fig. 4.a).



a) b)

Figure 4. Discretized C-Space-Time and C-Obstacles: a) both dynamical (red & green) and static (blue) objects becomes static hyper-objects in the Spacetime; b) robot silhouette at different time slice

We can still call them as C-Obstacles, remembering that they have been extended in the time dimension. In the case of the *Obstacles Layer*, we have 3D Spacetime: R^2 x R, where R^2 is the planar workspace. The overall "static" spacetime obstacle, representing one robot, is the composition of all the robot silhouettes in each time slice (Fig. 4.b). If we consider all the robots moving at the same speed, the time step (slice) is the time needed by a robot to move to an adjacent cell (the robots motions are synchronized). This layer is also called Coordination Layer (/Space) because the motions of all the robots are coordinated in this layer (see § 3.5).

3.4 The Attraction Layer

The *Attraction Layer* is the core of the algorithm. It computes the shortest collision-free path for a single robot, from the starting pose to the goal pose, while considering its real occupation (shape and size = silhouette). To pass from one cell to another one, the robot can follow different paths, combining different atomic moves, such as *straight_forward* move, *diagonal* move, *rotation*, and so on. Each move has its own cost (always positive); the entire set of admissible movements defines the robot kinematics. The kinematics together with the silhouette is the motion model of the robot (the third "ingredient" needed to make planning). This planner is able to handle different types of robots at the same time, with different shapes, sizes and kinematics (e.g. car-like kinematics, holonomic, Dubins' car, etc.), providing the proper trajectory for each one. The moves are space-temporal, i.e. every time the robot moves in the Space, it also moves in the Time. Even when the robot stands in the same place, it moves (vertically) in the Spacetime (also this particular move has its own cost). The moves costs are used to build incrementally a potential surface starting from the goal cell, placed at a high time slice (the desired time of arrival), and expanding it in all the free Spacetime. The hyper-surface is defined on a 4D space: R^2 x **SO**(2) x R, where R^2 x **SO**(2) is the C-Space. The goal cell receives the first value (a seed), from which a potential bowl is built adding the cost of the movement that would bring the robot from a surrounding cell to it. The computation is performed by the automata in the spacetime cell, which compute the potential value depending on the potential values of the surrounding

spacetime cells. Iterating this process for all the free cells (avoiding the cells occupied by a hyper obstacle), the potential surface is built leaving the goal cell with the minimum potential. The potential value of a cell has a particular mean: it is the total (minimal) cost needed to reach the goal from the current cell. Since the costs are positive, the bowl always grows with increasing values and no other minimum is generated, thus avoiding the well-known problem of the "local minima". The entire trajectory is computed just following the direction of the negated gradient of the surface from the starting point. The path results to be at the minimum valleys of the surface. Every robot has its own goal, kinematics and shape, hence a potential bowl has to be generated separately for each one. Since the potential bowl is built only in the free space, the robot *Attraction L.* depends on the contents of the *Obstacles L.*

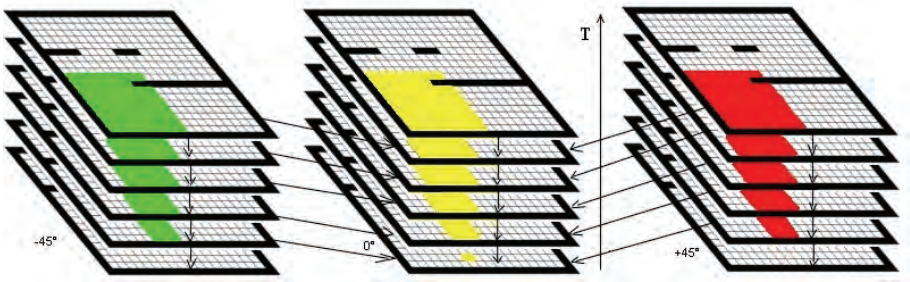


Figure 5. Dependencies between Attraction Layers at different orientations

The last one includes the Time, thus the potential bowl varies from time to time, generating different potential surfaces at different starting time and consequently, different trajectories depending on the time of arrival. In Fig. 5 is shown the dependencies between Layers and Sub-layers of the overall *Attraction Layer*. Each sub-layer, at a given robot orientation, depends on the adjacent sub-layers at different orientations (aside in the figure), and depends also on the below temporal sub-layer.

3.5 The Planning Algorithm

Entailing the layer structure previously described, it is now possible to introduce the algorithm that computes the robots movements.

To avoid the burden of the complexity of the problem using a centralized solution (LaValle & Hutchinson, 1998), we have adopted a Priority Planning approach. Therefore, the first step is to establish the priority level of every robot. Up to now, there is no evident heuristic to be used in this phase and the assignment is defined by the user or casually. The following step is to initialize every temporal sub-layer with the obstacle distribution known from the map of the workspace (a simple occupancy grid). Then, the algorithm computes the *Attraction L.* (potential bowl) of the robot with the highest priority level, also considering the information of the *Obstacles L.* Setting the minimum potential value to the goal cell makes the potential bowl grow while surrounding the obstacles (green cells in Fig. 6).

Once the potential surface is known, the shortest path is extracted, following the negated gradient from the starting cell. For each passing point, then it adds the robot silhouettes, properly oriented, in temporal sub-layer of the *Obstacles L.* In this way, the first robot becomes a spacetime obstacle for the successive robots having lower priorities. The *Obstacles Layer* plays an important role in this phase: it ensures that the robots do not collide, even

taking into account their real extensions. For this reason, in this context it is also called the Coordination Space. The next phase is to repeat the procedure for the second robot, starting to build its *Attraction L.* on the base of the modified *Obstacles L.*, and then adding to it the sequence of its silhouettes along the spacetime trajectory. The algorithm terminates when it has computed all the robots trajectories.

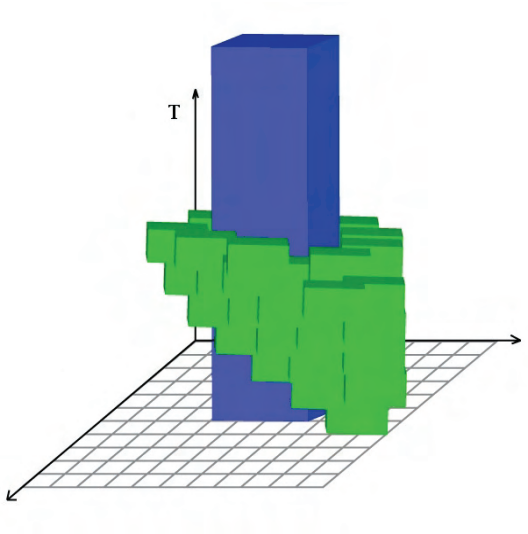


Figure 6. Spacetime Potential bowl growing around a static obstacle

4. Simulation Results

The first consideration is that the priority planning is not complete: there are situations for which there is a solution, but this type of algorithm is not able to find it. In Fig. 7 a simple counterexample is shown (a similar one in (Bennewitz, 2001)) of this type of problems with a simple solution. Adopting a specific priority order (e.g., the green robot moves before the red one) there is no solution. Because this problem is symmetric (the world is symmetric with respect to a horizontal axis, the two robots have the same shape and kinematics), even swapping the priority order (e.g. the red one passes before the green robot) there is still not a solution with a priority schema. Fortunately, for most of the situations we do not have this type of problem and the algorithm can easily find a solution.

In the example of Fig. 8, the red robot has to free the way to permit the green one to get out of the room, and then it goes back to the original position. This is an interesting situation, where a robot, which should stand without moving, is enforced to move by the coordinator to free the passage. The triangles represent the kinematic center and the robot orientation.

The following example (Fig. 9) shows a classical problem where the robots trajectories have to cross to exchange their positions in the four corners. In the middle of the environment there is a conflict zone (interference zone) where all the robots have to pass. All the robots start contemporarily, but the red and yellow robots have to stop and wait for the blue and the green ones (with higher priority values) to pass and exit from the conflict zone. Then, also the red and yellow ones can complete their motions.

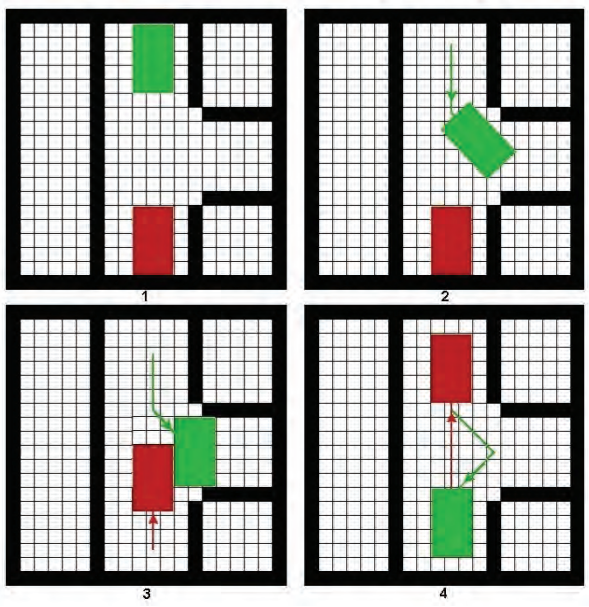


Figure 7. Counterexample: an obvious solution of a problem for which the priority planning fails

In the example of Fig. 10, four robots have to permutate (right rotation) their docking poses. Because the green robot has the highest priority, it enforces the other robots to wait before entering in the new stations.
The example of Fig. 11 concerns an O-ring shaped robot that closes the way to a second robot. The O-ring robot has its kinematics center over a static obstacle, thus it can only rotate to free the passage to the other robot.
In Fig. 12, two robots with unusual shapes and different kinematics (holonomic the green one, car-like the red one) swap their positions between two rooms passing through a narrow door. The silhouettes of the two "robots" are the projections on the plane of objects like a "Truck" and a "Portainer crane" moving on wheels. The task is to make passing the truck under (between the uprights) the crane. This is a very unusual example of planning for a vehicle with a silhouette composed by two disjoined parts and where a second object passes over the kinematic center of the first.
All the times reported have been measured on a PC Intel Core 2 Duo CPU 2.20 GHz. In parenthesis it is also indicated the number of hyper-cells composing the entire C-Space-Time.

5. Conclusions

In this work we have proposed a new version of a decoupled and prioritized motion-planning algorithm for the coordination of a Multi-Robot composed by mobile robots. This Motion-Planner is a single module of a wider multilayered architecture for the coordination of MRS. Using a Multilayered Cellular Automata as paradigm for a joined space representation and planning technique, we face contemporarily mobile robots having

generic shapes and sizes. It is based on the Priority Planning approach (decoupled approach) in the C-Space-Time where the Time has been added to the normal C-Space. The Priority Planning is not complete, but it works very well for most of the situations, finding all the collision-free equivalent paths for each robot, while simplifying the complexity of the problem. Differently from most of the other planners, the algorithm is also able to handle the robots orientations and robots with different kinematics. This property avoids wasting a lot of space during the motion, and permits to find paths in cluttered workspaces. The trajectories found are smoothed and respect the kinematics constraints of each robot.

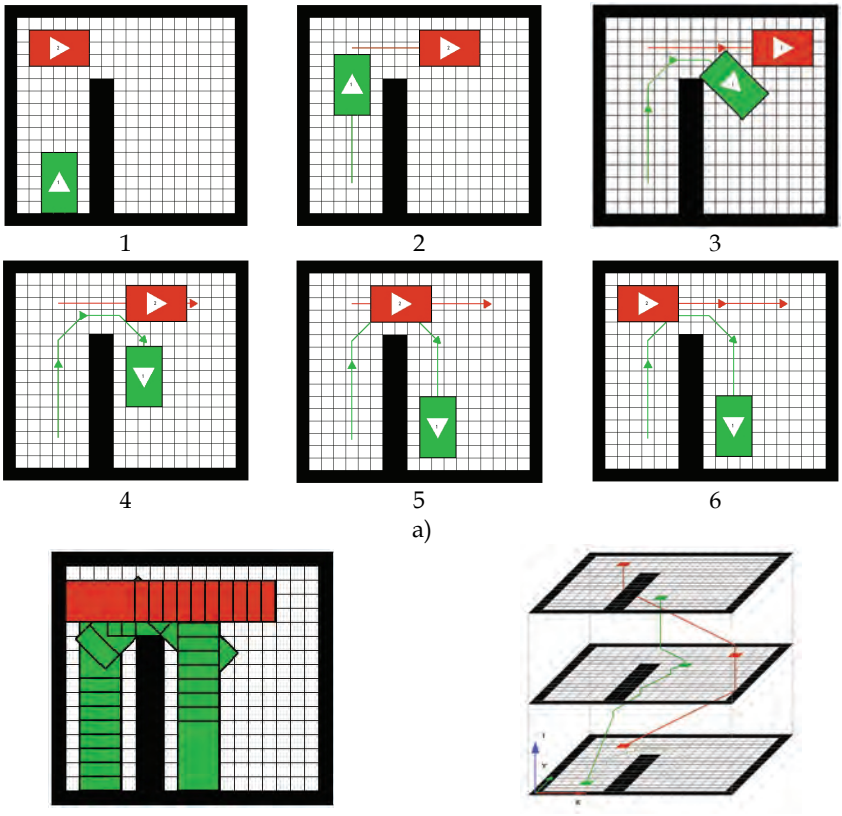


Figure 8. Robot clearing the way: a) snapshots sequence; b) overall movements; c) C-Space-Time movements (planning time: 0.19 s, 259'200 cells)

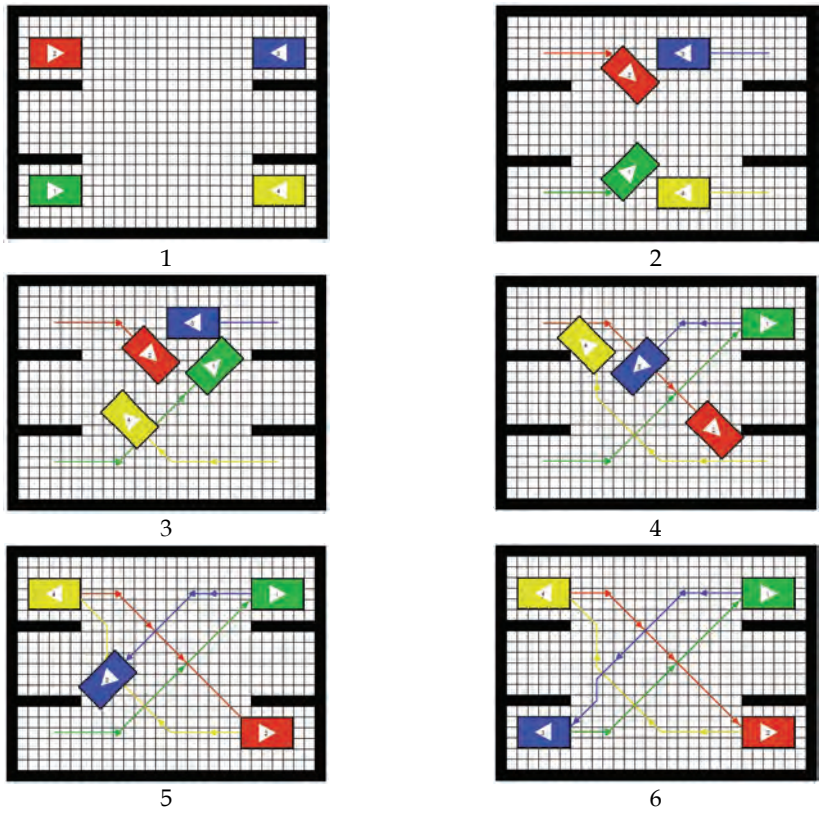
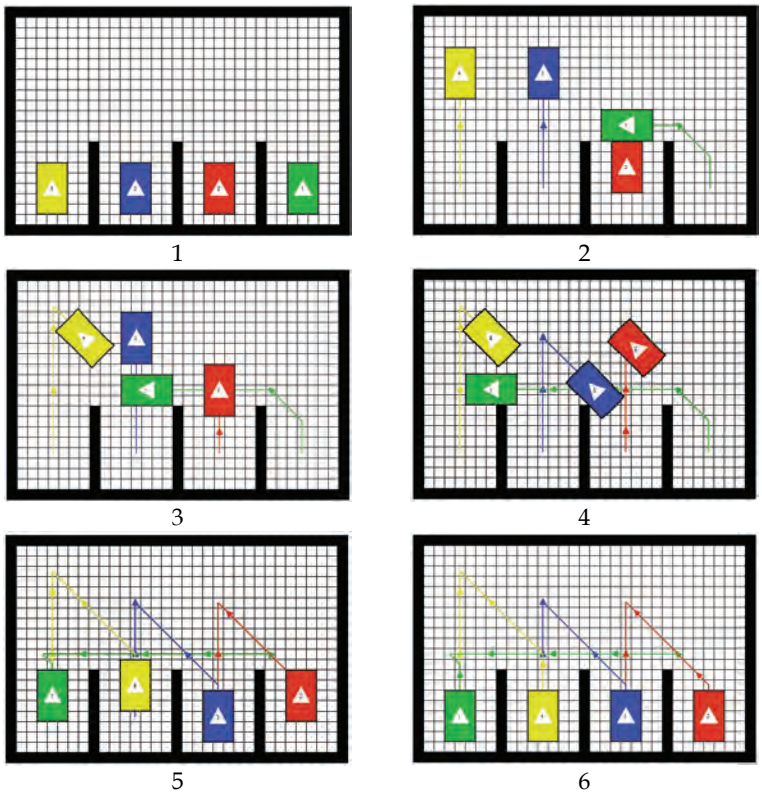


Figure 9. Crossing robots: a) snapshots sequence; b) overall movements; c) C-Space-Time movements (planning time: 1.26 s, 897'600 cells)



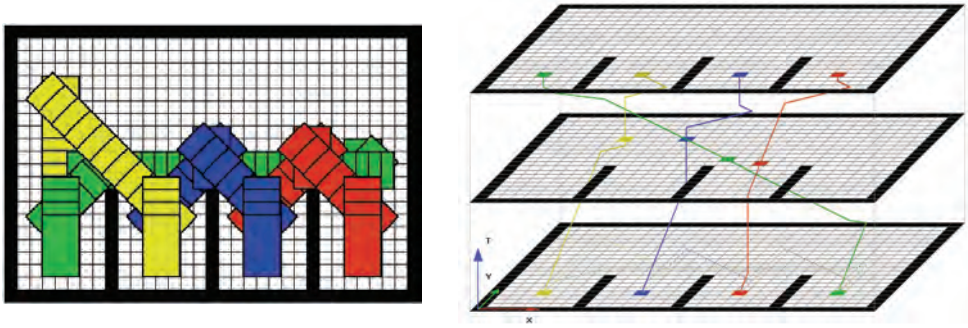
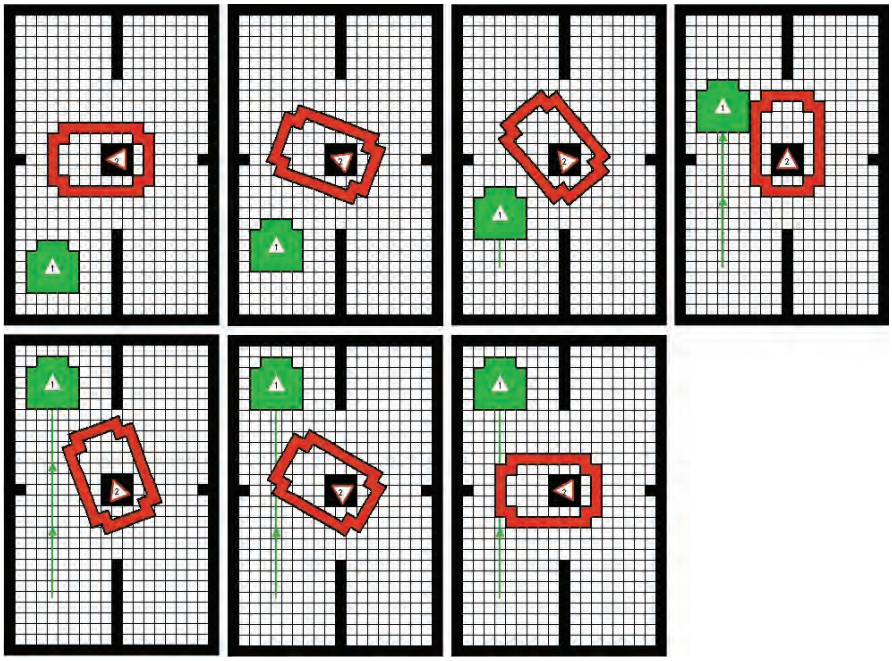
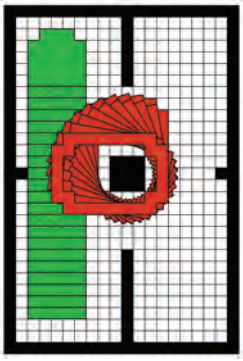


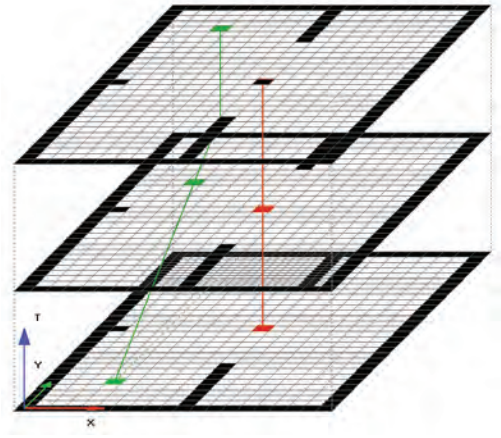
Figure 10. Circular permutation: a) snapshots sequence; b) overall movements;
c) C-Space-Time movements (planning time: 3.03 s, 987'360 cells)



a)

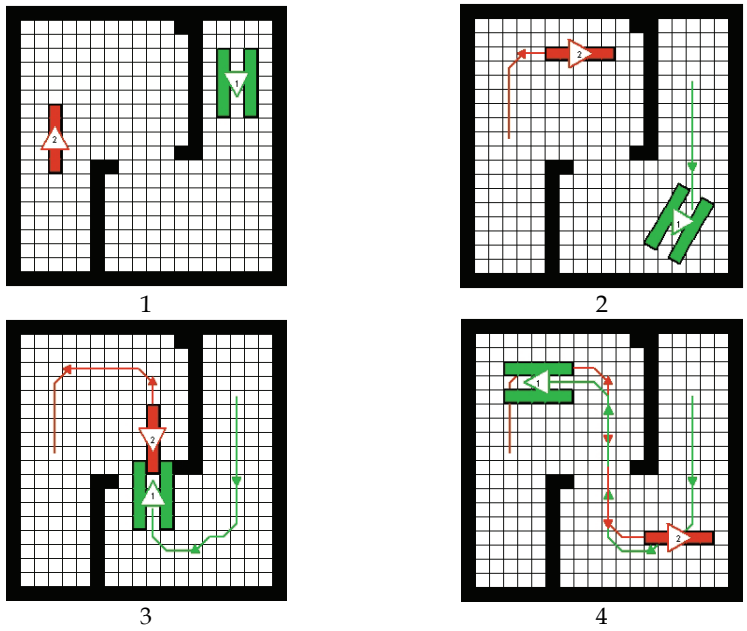


b)

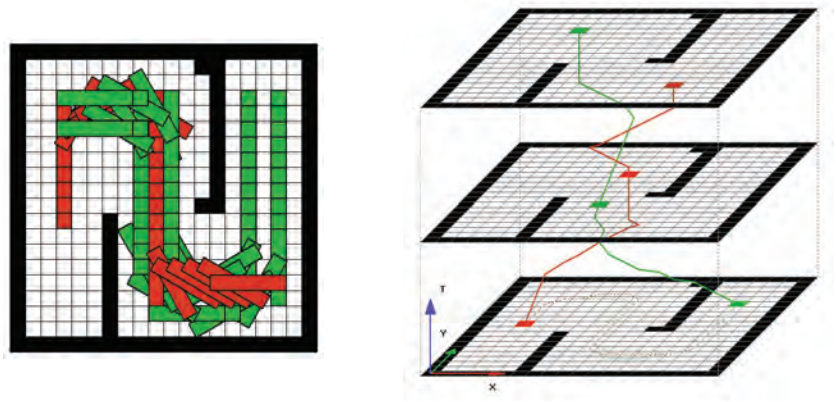


c)

Figure 11. O-Ring example: a) snapshots sequence; b) overall movements; c) C-Space-Time movements (planning time: 0.16 s, 1'554'000 cells)



a)



b) c)

Figure 12. Truck & Portainer Crane example: a) snapshots sequence (crane in green, truck in red); b) overall movements; c) C-Space-Time movements (planning time: 0.33 s, 416'000 cells)

6. References

Arai, T.; Pagello, E. & Parker, L.E. (2002). Editorial: Advances in Multi-Robot Systems, *IEEE Trans. on Robotics and Automation*, Vol. 18, No. 5, Oct. 2002, 655-661, ISSN=1042-296X

Barraquand, J.; Langlois, B. & Latombe, J. C. (1992). Numerical potential field techniques for robot path planning, *IEEE Trans. on Systems, Man and Cybernetics*, Vol. 22, No. 2, 224-241, ISSN:0018-9472

Bennewitz, M.; Burgard, W. & Thrun, S. (2001). Optimizing schedules for prioritized path planning of multi-robot systems, *Proceedings of IEEE Int. Conf. on Robotics and Automation (ICRA)*, 271-276, ISSN:1050-4729, Seoul (Korea), May 2001

Dudek, G.; Jenkin, M.; Milios, E. & Wilkes D. (1996). A taxonomy for multiagent robotics, *Autonomous Robots*, Vol. 3, 375-397

Erdmann M. & Lozano-Pérez, T. (1986). On multiple moving obstacles, Proceedings of IEEE International Conference on Robotics and Automation, pp. 1419-1424, San Francisco, April, 1986

Farinelli, A.; Iocchi, L. & Nardi, D. (2004). Multirobot systems: a classification focused on coordination, *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, Vol. 34, No. 5, Oct. 2004, 2015-2028, ISSN: 1083-4419

Goles, E. & Martinez, S. (1990). *Neural and Automata Networks: dynamical behavior and applications*, Kluwer Academic Publishers, ISBN:0-7923-0632-5, Norwell, MA, USA

Jahanbin, M. R. & Fallside, F. (1988). Path planning using a wave simulation technique in the configuration space, In: *Artificial Intelligence in Engineering: Robotics and Processes*, J. Gero S. (Ed.), Computational Mechanics Inc., ISBN:0-931215-98-6, Billerica, MA, USA

Kathib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots, *Int. J. of Robotics Research*, Vol. 5, No. 1, 90-98

Latombe, J. C. (1991). *Robot Motion Planning*, Kluwer Academic Publishers, ISBN: 978-0792392064, Boston, MA

LaValle, S. M. & Hutchinson, S. A. (1998). Optimal motion planning for multiple robots having independent goals, *IEEE Trans. on Robotics and Automation*, Vol. 14, No. 6, 912-925, ISSN:1042-296X

Lozano-Pérez, T. (1983). Spatial planning: A configuration space approach, *IEEE Trans. on Computers*, Vol. C-32, No. 2, 108-120, ISSN:0018-9340

Lozano-Pérez, T. & Wesley, M. A. (1979). An algorithm for planning collision-free paths among polyhedral obstacles, *Comm. of the ACM*, Vol. 22, No. 10, 560-570, ISSN:0001-0782

Marchese, F. M. (2007). An Architecture for MultiRobot Motion Coordination, *Proceeding of Robotics and Applications and Telematics (RA 2007)*, 352-357, ISBN: 978-0-88986-685-0, Würzburg, Germany, August 2007

Tzionas, P. G.; Thanailakis, A. & Tsalides, P. G. (1997). Collision-free path planning for a diamond-shaped robot using two-dimensional cellular automata, *IEEE Trans. on Robotics and Automation*, Vol. 13, No. 2, 237-250, ISSN:1042-296X

Warren, C. (1990). Multiple robot path coordination using artificial potential fields, *Proceedings of IEEE Int. Conf. on Robotics and Automation*, 500-505, May 1990

Zelinsky, A. (1994). Using path transforms to guide the search for findpath in 2d, Int. J. of Robotics Research, Vol. 13, No. 4, 315-325

8

Q-Learning Adjusted Bio-Inspired Multi-Robot Coordination

Yan Meng
Department of Electrical and Computer Engineering, Stevens Institute of Technology
USA

1. Introduction

In a multi-robot system, each robot needs work together with the network of other robots, considering options for matching its capabilities with demand, negotiating on such constraints as quality, price and time, and then making decisions for committing resources to match demands. Multi-robot systems demand group coherence (robots need to have the incentive to work together faithfully) and group competence (robots need to know how to work together well). With recent advances in all aspects of the technology associated with computing, energy, sensing, and networking infrastructure, more progress has been made in the developing of multi-robot systems for a potentially dynamic, challenging, and hazardous environment. Some examples of such applications include search and rescue, mine detection, hazardous material collection (or cleanup), reconnaissance, smart home/office, surveillance, construction, planetary exploration, and transportation. Since human assistance in these challenging environments is limited due to distance or the need for quick response to changing circumstances, more advanced techniques, such as self-adaptive and self-evolving, would be desirable for these multi-robot systems, which are still unsolved research areas.

Some common challenges exist for this kind of systems. It is often not hard to implement a rudimentary controller that accomplishes the task, but achieving optimal performance can be very challenging. Coordination is hard when robots are really self-interested. If each individual robot is very complex with plenty of intelligent functionalities, through the interacting with other robots and environments, the overall systems will become computational intractable with the large scale agents.

To achieve the global intelligence of a cooperative multi-agent system, distributed coordination methods are more attractive compared to the centralized methods due to its robustness, flexibility, and adaptivity. However, designing a self-adaptive multi-robot system is not a trivial task. Nolfi and Floreano [Nolfi and Floreano, 2000] claim that, since the individual behavior is the emerging result of the interaction between agents and environment, it is difficult to predict which behavior results from a given set of rules, and which are the rules behind an observed behavior. Similar difficulties are present in the decomposition of the organized behaviors of the whole system into interactions among individual behaviors of the system components. The understanding of the mechanisms that led to the emergence of self-organization must take into account the dynamic interactions

among individual components of the system and between these components and the environment. Thus, it is difficult to predict, given a set of individual behaviors, which behavior at the system level will emerge, and it is also difficult to decompose the emergence of a desired global behavior into simple interaction among individuals [Dorigo, 2004]. To develop intelligent robots that can adapt their behaviors based on interaction with the environment and other robots, become more proficient in their tasks over time, and adapt to new situations as they occur, more researchers turned their attentions to bio-inspired systems, such as social insects. Swarm intelligence is an innovative computational and behavioral metaphor by taking its inspiration from the behavior of social insects swarming, flocking, herding, and shoaling phenomena in vertebrates, where social insect colonies are able to build sophisticated structures and regulate the activities of millions of individuals by endowing each individual with simple rules based on local perception.

In this chapter, we propose a novel bio-inspired coordination paradigm, i.e. QVP-PSO, to achieve an optimal group behavior for multi-robot systems, which is the combination of a reinforcement learning method and a bio-inspired Visual Pheromone based Particle Swarm Intelligence (VP-PSO). Basically, two coordination processes among the robots are established in the proposed QVP-PSO architecture. One is a virtual pheromone based algorithm to guide the robots' movements for targets, where each robot has its own virtual pheromone matrix, which can be created, enhanced, evaporated, and propagated to its neighboring robots. The other one is Particle Swarm Optimization (PSO)'s cognitive capabilities through local interaction, which aims to achieve the balance for each robot between the exploration and exploitation through the interactions among the robots using the PSO-based algorithm. To adapt to the changing environment for optimal group behaviors, a Q-learning method is applied to dynamically adjust the associated parameters of the PSO method so that the balance between the explorative, cognitive, and social factors can be optimized under different scenarios.

The paper is organized as follows: Section 2 describes the problem statement. The virtual pheromone based PSO (VP-PSO) scheme is explained in Section 3. Then the Q-learning adjusted VP-PSO method is discussed in Section 4. Section 5 presents the simulation results using the embodied robot simulator: Player/Stage. To conclude the paper, Section 6 outlines the research conclusions and the future work.

2. Related work

Extensive multi-robot coordination techniques have been developed for various applications, such as foraging, box-pushing, aggregation and segregation, formation forming, cooperative mapping, soccer tournaments, site preparation, sorting, and collective construction. [Balch and Arkin, 1999] [Stewart, 2006] [Dias et al., 2004] [Burgard et al., 2005] [Chaimonwicz, 2004] [Fernandez, 2005][Martinoli, 1999] [Weigel, 2006] [Parker and Zhang, 2006] [Holland and Melhuish, 1999][Correll and Martinoli, 2006]. All of these systems consist of multiple robots or embodied simulated robots acting autonomously based on their own individual decisions.

Some computational intelligence algorithms have been proposed, such as neural networks [Rumelhart and McLelland, 1986], genetic algorithms [Holland, 1975], evolution strategies [Rechenberg, 1973], immune networks [Bersini and Varela, 1991], Ant Colony Optimization

(ACO) [Dorigo et al, 1996] and Particle Swarm Optimization (PSO) [Kennedy and Eberhart, 1995], seeking to replicate the related natural behaviors. All of the above nature-inspired computational intelligence has proved to be effective and efficient approaches towards design and control of complex problems under dynamic environments.

Although artificial evolution has been often used for synthesizing behaviors for autonomous robots [Nolfi and Floreano, 2000], its use as a methodology to evolve behaviors for groups of robots is still limited. Reynolds [Reynolds, 1987] built a computer simulation to model the motion of a flock of birds, called *boids*. He believes the motion of the boids, as a whole, is the result of the actions of each individual member that follow some simple rules. Ward et al. [Ward, 2001] evolved *e-boids*, groups of artificial fish capable of displaying schooling behavior. Spector et al. [Spector et al., 2003] used genetic programming to evolve group behaviors for flying robots in a simulated environment. The above mentioned works suggest that artificial evolution can be successfully applied to synthesize effective collective behaviors. Dorigo et al. [Dorigo et al., 1996] developed a robotic system consisting of a swarm of s-bots, mobile robots with the ability to connect to and to disconnect from each other depends on different environments and applications, which is based on behaviors of ant systems. Payton et al. [Payton et al., 2001] proposed pheromone robotics, which was modeled after the chemical insects, such as ants, use to communicate. Instead of spreading a chemical landmark in the environment, they used a virtual pheromone to spread information and create gradients in the information space. By using the virtual pheromone, the robots can send and receive directional communications to each other. Pugh and Martinoli [Pugh and Martinoli, 2006] proposed a group learning algorithm using Particle Swarm Optimization for a multi-robot system, where aggregation behaviors were conducted to evaluate the proposed methods. Werfel and Nagpal [Werfel and Nagpal, 2006] proposed an extended pheromone by increasing the capabilities of environmental elements in swarm robots to automatically assemble solid structures of square building blocks in two dimensions according to a high-level user-specific design. In our previous work [Meng et al. 2007][Meng and Gan, 2007], swarm intelligence based robot coordination methods were proposed.

3. Virtual pheromone based PSO method

The objective of this study is to design an efficient and robust distributed coordination algorithm for a multi-robot system with limited on-board power, sensing and communication on each robot, aiming at optimizing the group behavior especially for a searching task under a dynamic environment. The targets can be defined as any kind of predefined object. It is assumed that the searching area is bounded and robots can detect the targets using special on-board sensors, such as camera systems. The robot can only detect the targets within its local sensing range. Once a robot detects a target, it processes the target assuming the processing time is proportional to the size of target. Assume that the robots are simple, and homogeneous. Each robot can only communicate with its neighbors. Two robots are defined as neighbors if the distance between them is less than a pre-specified communication range. The goal is to detect and process all of the targets within the searching area as soon as possible.

3.1 Virtual Pheromone

Pheromone is a class of mechanisms that mediate animal-animal interactions through artifacts or via indirect communication, providing a kind of environmental synergy, information gathered from work in progress, distributed incremental learning and memory among the society. To emulate pheromone-based communication in a multi-robot system, special pheromone materials and associated detectors need to be designed, and most of the time such chemical/physical pheromone is unreliable and easily be modified under some hazardous environments, such as urban search and rescue. A modification of this autocatalysis is necessary. Similar to [Payton, 2001], a unique virtual robot-to-robot interaction mechanism, i.e. *virtual pheromone,* was proposed as the message passing coordination scheme for the swarm robots.

Each target in the environment is associated with one unique pheromone, which can be enhanced or evaporated over time to adapt to a dynamic environment. Initially, each robot creates its own virtual pheromone matrix, which installs all pheromone information associated with different targets. Whenever a robot detects a target, it would update its own pheromone matrix and broadcast this target information to its neighbors through a visual pheromone package.

3.2 Fitness function

To emulate the pheromone creation, enhancement and elimination procedure in natural world, the pheromone density $\tau_{ij}^{k}(t)$ can be updated by the following equation:

$$\tau_i^k(t+1) = \rho(\tau_i^k(t) + \alpha) - (1-\rho)m\tau_i^k(t) \tag{1}$$

where $\tau_i^k(t)$ represents the pheromone density of target *i* at time *t* for agent *k*. $0<\rho<1$ is the enhancement factor of the pheromone density. α is the pheromone interaction intensity received from the neighboring robots for target *i*. Basically, α is used for pheromone enhancement, and *m* represents the elimination factor. In the ants system, the pheromone will be eliminated over time if it is not being enhanced by the ants, and the elimination procedure usually is slower than the enhancement. When the pheromone trail is totally eliminated, it means that no target is needed to be processed through this pheromone. To slow down the elimination relative to enhancement, *m* is set as less than 1.

To define the probability that robot *k* moves toward target *i* with pheromone density $\tau_i^k(t)$, the target utility function is defined as following:

$$\mu_i^k(t) = \tau_i^k(t)e^{-\gamma_i} \tag{2}$$

where γ_i represents local target redundancy, which is defined as the number of the local neighbors who have sent the pheromone referring to the same target *i* to robot *k*.

Generally speaking, the higher the target utility is, the more attractive the corresponding target is to the robot. Therefore, the benefit of moving to this target would be higher in terms of the global optimization. If the local target redundancy is high, it means that there will be more potential robots (globally) moving to this target, which may lead to the less available targets left in the future. Therefore, the benefit of moving to this target would be less in terms of the global optimization. With the local redundancy, we are trying to prevent

the scenarios that all of the robots within a local neighbor move to the same target instead of exploring new targets elsewhere.
Initially, the robots are randomly distributed in the searching environment, where multiple targets with different sizes and some static obstacles are randomly dispersed within the environment. At each iteration, if each robot adjusts its behavior based only on the target utility, it may lead the robot to be very greedy in terms of the robots' behaviors, since the robots would rather move to the target with higher utility than explore new areas. This greedy behavior of the robots may easily lead to local optima.
To prevent the local optima scenarios with utility-greedy method using (2), the target visibility has to be considered as well. Let $\eta_i^k(t)$ denotes the target visibility for agent k in terms of target i, which is defined as:

$$\eta_i^k(t) = \max\left(\frac{\delta^k}{d_i^k(t)}, 1\right) \tag{3}$$

where δ^k represents the local detection range of robot k, and the $d_i^k(t)$ represents the distance between the robot k and target i. When the target visibility is higher, it means the distance between the target and the robot is smaller, it would be more beneficial to move to this target due to its lower cost compared to moving to the more distinct target under the same environmental condition.

3.3 Coordination of Robot Behaviors

One of the objectives of the pheromone update rules is to prevent stagnation, which occurs when most of the robots follow the same path, or converge to the same target. In general, global updates will facilitate exploitation, while local updates will favor exploration by letting each robot update pheromone after each transition decision. During each local update, the pheromone will diminish due to evaporation. Over time, frequently visited links will become less attractive, thus favoring exploration of less frequently used links. The local update by each agent therefore will avoid a very strong link from dominating as a component of the final solution. The utility-greedy using (2) has the tendency which may lead to the stagnation.
Now the question is how to integrate the target utility and target visibility into an efficient fitness function to guide the movement behaviors of each robot so that the stagnation can be avoided. Stagnation occurs when most of the robots follow the same path, or converge to the same target. To tackle this issue, we turned our attention to the Particle Swarm Optimization (PSO) method. The PSO algorithm is a population-based optimization method, where a set of potential solutions evolves to approach a convenient solution (or set of solutions) for a problem. The social metaphor that led to this algorithm can be summarized as follows: the individuals that are part of a society hold an opinion that is part of a "belief space" (the search space) shared by every possible individual. Individuals may modify this "opinion state" based on three factors: (1) The knowledge of the environment (explorative factor); (2) The individual's previous history of states (cognitive factor); (3) The previous history of states of the individual's neighborhood (social factor).
Basically, the PSO algorithm can be represented as in (4), which is derived from the classical PSO algorithm [Kennedy and Eberhart, 1995] with minor redefinitions of formula variables:

$$v = explorative + cognitive + social \tag{4}$$

where v is the velocity of a robot. To determine which behavior is adopted by robot k, the velocity has to be decided first. If the received pheromone density is high, the robot would increase the weight of social factor, and decrease the weight of the cognitive and exploration factors. On the other hand, if the local visibility is of significant to the robot, then the velocity of the robot would prefer the cognitive factor to the social factor. If both social and cognitive factors are low, then the exploration factor would be increased. Furthermore, at any given time, the velocity of the robot would leave some spaces for the exploration of new areas no matter what. Therefore, the basic idea is to propel towards a probabilistic median, where explorative factor, cognitive factor (local robot respective views), and social factor (global swarm wide views) are considered simultaneously and try to merge these three factors into consistent behaviors for each robot. The exploration factor can be easily emulated by random movement.

The challenging part is how to define the local best (cognitive factor) and the global best (social factor). One straight forward method is to select the highest target visibility from a list of available targets as the local best. If only one target is on the list, then this target would be the local best. The easy way to select global best is to select the highest target utility from a list of available targets.

Instead of defining a fitness function, for a robot system, the robot velocity vector including both magnitude and direction would be a better representation to control the movement behavior. Based on the above discussion and the PSO algorithm, each robot would control its movement behaviors by following this equation:

$$v^k(t+1) = \psi_e rand_e(\,)v^k(t) + \psi_c rand_c(\,)(p_c - x^k(t)) + \psi_s rand_s(\,)(p_s - x^k(t)) \qquad (5)$$

where, $\psi_e, \psi_c, and \psi_s$ represent the propensity constraint factors for explosive, cognitive, and social behaviors, respectively, $0 \le rand_\Theta() < 1$ where $\Theta = e$, c, or s, and $x^k(t)$ represents the position of robot k at time t. $p_s = \max(\mu_i^k(t))$ represents the global best from the neighbors, and $p_c = \max(\eta_i^k(t))$ represents the local cognitive best. The position of each robot k at time $t+1$ can be updated by

$$x^k(t+1) = x^k(t) + v^k(t+1)\,. \qquad (6)$$

4. Q-Learning adjusted method

Since the environment is dynamic, it is difficult to guarantee that the initial parameters of the VP-PSO method will be the best-fit for the current scenario all the time. Therefore, it is necessary to dynamically adjust these parameters to achieve optimal global performance and expedite the convergence.

After some preliminary experiments using the VP-PSO method, it was observed that with different weight parameters of the PSO, the searching performance varied noticeably, i.e., the convergence rate and the target/robot distribution ratio may be different. For example, it is assumed that there are 10 targets which are distributed in groups in three different locations. There are 20 robots. We conduct 2 cases with different parameters of the PSO method. In case 1, the target/robot distribution map is 3(target)/10(robot), 5/6, and 2/4. In case 2, the distribution map becomes 3/7, 5/12, and 2/1. In case 1, after finishing the 3

targets, the first group of robots has to start searching again for other unfinished targets, such as second locations. Extensive travel cost would be consumed for the searching. In an ideal situation, each target should attract 2 robots by average. The processing time could be reduced significantly since extensive unnecessary travel cost has been saved. However, since the system is distributed, there is no global station to check the status of other targets or robots. Each robot has to make decisions based on its local sensor feedbacks.

It would be desirable that robots have the self-learning capability to dynamically adjust the parameters of the coordination algorithms according to their current status and sensor feedbacks. Since the environment of a multi-robot system is dynamically changing, the measurement feedbacks from the environment would be critical to help to adjust the coordination methods for the changing situations. To achieve a higher learning performance, it is assumed that the predefined expected robot/target ratio is provided. Q-learning method is applied to adjust the parameters of the PSO for better coordination behaviors. Q-learning is a learning technique for acquiring optimal actions based on the evaluation values $Q(s,a)$ (Q-value) for state-action sets. To simplify the method, the gradient-based action for parameter adjustment will be applied. For example, parameter X will be increased by 0.5 unit of gradient and Y will be decreased by 1.3 unit of gradient. The Q-learning method is summarized as followings:

1. Initialize all states $s \in S$ and action $a_i \in A_i$, and $Q_i(s, a_1 \cdots a_n) = 0$;
2. Repeat for every 10 step's state $s \in S$:
   1) Get target info $w_i(t)$ from the pheromone matrix;
   2) Choose an action a_i according to $a_i = \arg\max_{a_i \in A_i} Q_i(s, a_1^t \cdots a_n^t)$
   3) Observe rewards r_i^t , and the next state S^{t+1}

$$Q_i(s^{t+1}, a_1^{t+1} \cdots a_n^{t+1}) = (1-\beta) Q_i(s^t, a_1^t, \cdots, a_n^t) + \beta(r_i^t + \lambda \tilde{Q}_i(s^{t+1}, a_1^{t+1}, \cdots, a_n^{t+1}))$$

where $$\tilde{Q}_i = p_i max\left(\frac{w_i^t - \varepsilon\gamma_i^t}{d_i^t}, 0\right) + (1-p_i)\frac{(w^T - w_i^t)}{rand(t)(d_i^t - \delta)}$$

where β is the learning factor. w_i^t is the weight of detected target *i*.. γ_i^t represents the estimated value of robots redundancy at target *i* , ε is the expected target/robot ratio, and d_i^t is the distance between the robot's current position and the target *i*. Therefore, $\frac{w_i^t - \varepsilon\gamma_i^t}{d_i^t}$ represents the benefit a robot can gain by moving towards target *i*. When the number of robots around a target extends the expected target/robot ratio, the benefit is set to 0. w^T is the total target weights within the search area, and δ is the sensor detection range. It is assumed that the targets are distributed within the search area in uniform probability density, thus the benefit of random search can be represented by $\frac{(w^T - w_i^t)}{rand(t)(d_i^t - \xi)}$. p_i is the probability of going to target *i*, which is defined as $p_i = \frac{\psi_s}{\psi_e + \psi_c + \psi_s}$, where ψ_e, ψ_c,

and ψ_s are the weight parameters of Equation (5). The reward r_i^t = 1 if the robot number around the target *i* is approximately equal to ε, otherwise, r_i^t = -1 .

5. Experimental results

5.1 Experimental setup

Player/Stage is selected as our embodied robot simulator to implement the QVP-PSO algorithm using swarming robots. As shown in Fig. 1, the environment is an open space with the size of 41.8m x 45.1m, where several targets are distributed randomly. 20 homogeneous Pioneer 3DX robots are used as our robot model, which is equipped with a camera system to detect and track targets, a laser range finder to measure the distance between the target and itself, a sonar sensor to avoid obstacles (i.e. both static obstacles and mobile obstacles, such as other robots), and a wireless communication card to communicate with other robots. The arc shape in front of each robot represents the field of view of the vision system on each robot. The communication range is set up as the same range of the vision but using a circle instead of an arc. Whenever the robots are within other robot's communication range, they would exchange the information between them.

5.2 Robot localization and path planning

Self-localization is critical for multi-robot to coordinate with each other. Since the robots are working in an unknown environment, an a priori map is not available. Building a map using distributed multi robots is a challenging task requiring more computational complexity, which is not the focus of this paper. Since it is assumed that each robot is simple and small with limited on-board battery, a simplified localization is required. It is assumed that the initial positions of all robots relative to a global coordinate frame of searching area are given. Since each robot has encoders attached with wheel motors, odometry-based localization is employed here.

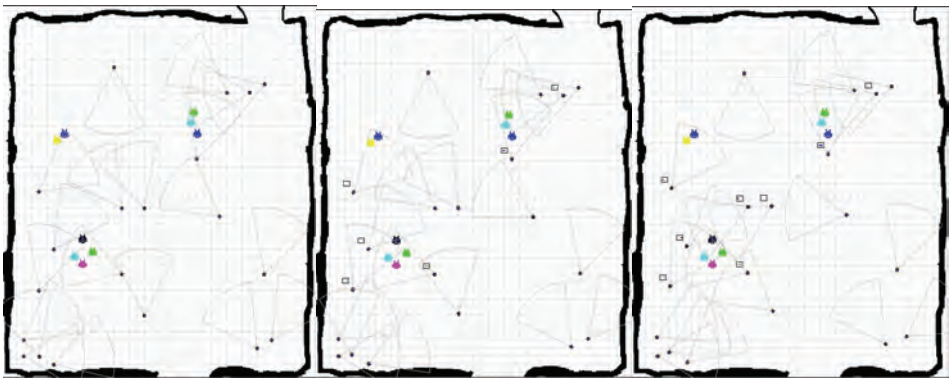
Since the targets are designed with different colors, a color blob detection using on-board camera system is carried out for target detection. Once a robot detects one or more targets, it uses on-board laser range finder to estimate the distance of the target relative to its own position. Then the target information can be propagated to its neighbors with the target position. When a robot picks a target, it would track the target color using the vision system while moving toward the target.

There are two important factors contributing to a dynamic environment. One is that robots always have to avoid obstacles, including static obstacles (i.e. walls) and dynamic obstacles (i.e. other mobile robots). It would be too computationally expensive for a robot since the global path has to be recalculated every time the robot avoids an obstacle. On the other hand, once a robot turns around to avoid one obstacle, it may become closer to other targets or it may receive new pheromone from its new neighbors, which naturally leads to new path planning for this robot. Therefore, it would be inefficient to use any complex global path planning. Here, a simplified path planning method is employed, where a robot sets up a destination location based on the detected or received target information, then the robot moves directly toward this destination. If an obstacle is in its way, the robot turns 45 degrees left if the obstacle is on the right side or turns right if the obstacle is on the left to

avoid the obstacle, then continue moving towards the destination. The destination may be changed due to new detection or new pheromone.

5.3 Experiments under an open-space environment

An open space experiment is conducted as shown in Fig. 1. Initially, the robots are randomly searching for targets at t = 1. Once a robot detects a target, it would propagate the pheromone of this target to its neighbors, as shown in Fig. 1(b), where a small rectangle beside a robot indicates that the on-board vision system has detected the targets. After receiving a pheromone message, robots make their own movement decisions based on the QVP-PSO algorithm, as shown in Fig. 1(c), Fig. 1(d), and Fig. 1(e). The simulation stops when all of the targets have been found and processed, as shown in Fig. 1(f).



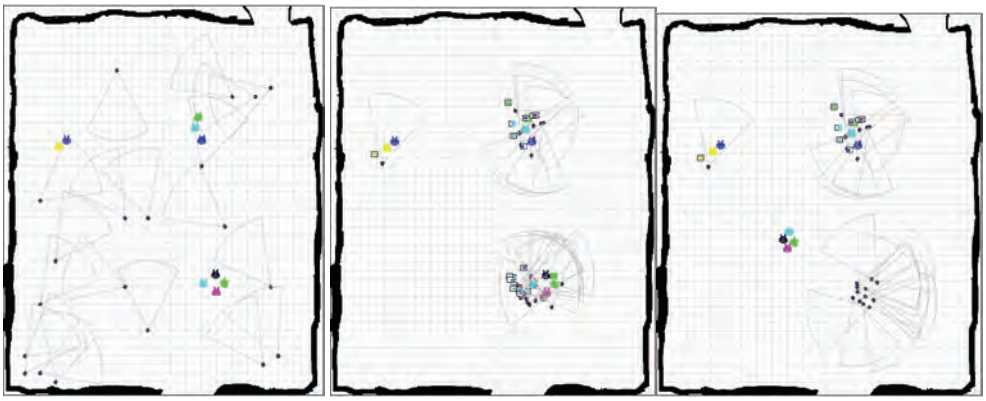
(a) t =1 (b) t = 17 (c) t = 20

(d) t = 50 (e) t = 203 (f) t = 277

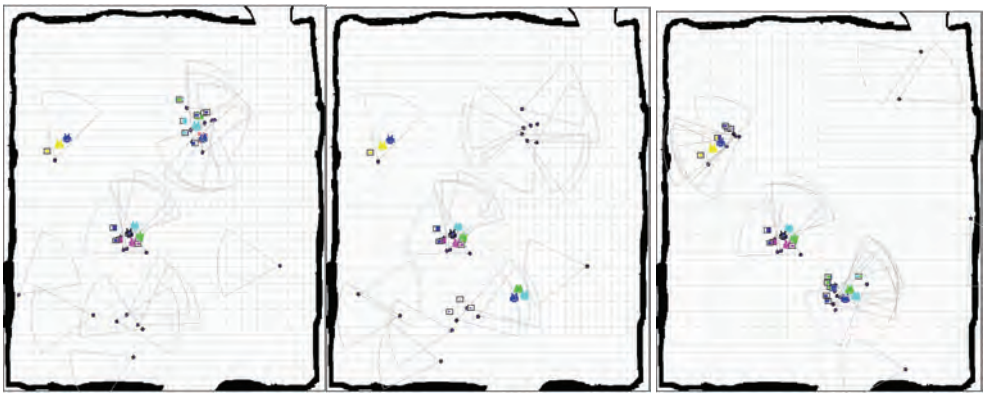
Figure 1. 20 robots search for randomly distributed targets in an open space on a player/stage simulator at t = 1, 17, 20, 50, 203, and 277 time steps

To evaluate the robustness of the QVP-PSO algorithm under a dynamic environment, another set of experiments are conducted in an open space, as shown in Fig.2. Since it is not allowed to dynamically change the target configuration in Player/Stage, the target relocations are conducted manually. As shown in Fig. 4, initially, 20 robots search for

targets. After some robots have reached to a target, the target is manually relocated to somewhere else. All of the robots around this target would disperse to explore new areas for new targets until all of the robots converge to the targets. It can be seen that the QVP-PSO algorithm is very robust to adapt to the unexpected events, such as target relocation.



(a) t = 1 (b) t=271 (c) t=288



(d) t = 682 (e) t = 693 (f) t = 987

Figure 2. Snapshots of simulation in an open space on Player/Stage with dynamic target relocations. (a) initial state; (b) all robot converge to targets; (c) relocate targets; (d) converge to new targets; (e) relocate targets; (f) converge to new targets

5.4 Experiments under an indoor office environment

Second set of experiment is conducted in an indoor office environment, as shown in Fig. 3, where the size of environment is 41.8m x 45.1m. Several targets with different colors are randomly distributed in the environment, and 20 robots search for them. The major difference of this experiment compared to the open space lies in the fact that even if the robot received the target information from its neighbors, it may not be able to move toward the target if there is a wall exits between the robot and target. The robot may have to spend some time to avoid the wall, and during this procedure, if new target information comes in, the robot may change its mind to move forward to new target based on the QVP-PSO algorithm.

Figure 3. 20 robots search for randomly distributed targets in an indoor office environment on a player/stage simulator at t = 1, 13, 15, 58, 423, and 495 time steps

5.5 Experimental Results

To evaluate the performance of the QVS-PSO algorithm, two other methods are carried out for comparison. One is random movement (Random), where all robots search for targets in a random manner without communicating with others. Second one is the VS-PSO algorithm without Q-learning adjustment. To obtain the statistic performance, we implemented the following experiments. 10 targets are distributed in the environment with fixed positions for the simulations. Then, we start running the simulations with the swarm size of 20 using three methods, each method runs 35 times to obtain result of ending time, the total travel time of all robots, and the mean squared error of robot/target distribution ratio. The results are shown in Fig. 4, Fig.5, and Fig.6, respectively.

The search ending time represents the searching performance. The less ending time the system takes, the faster the robots detect and process all the targets. To take the overall power consumption into the considerations, the total travel time of all the robots is a good measurement since robot movements usually consume higher percentage of power compared to the power consumption for the communication and onboard computation processing of the robots. The robot/target distribution ratio errors could tell us how good the algorithm can achieve to dynamically allocate the robots into different targets dynamically in a more reasonable manner.

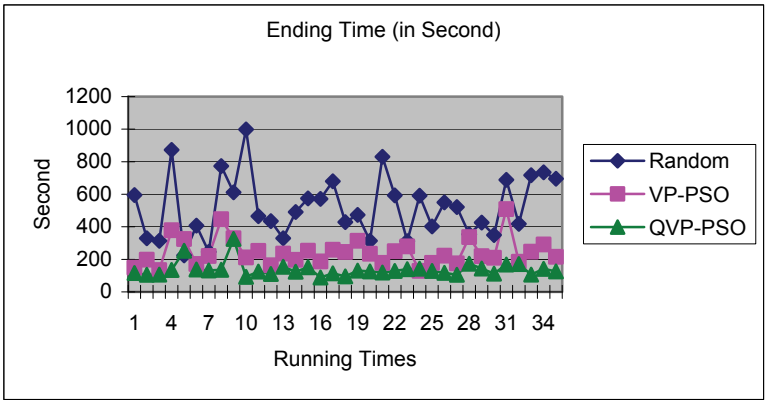


Figure 4. The search ending time vs. run times

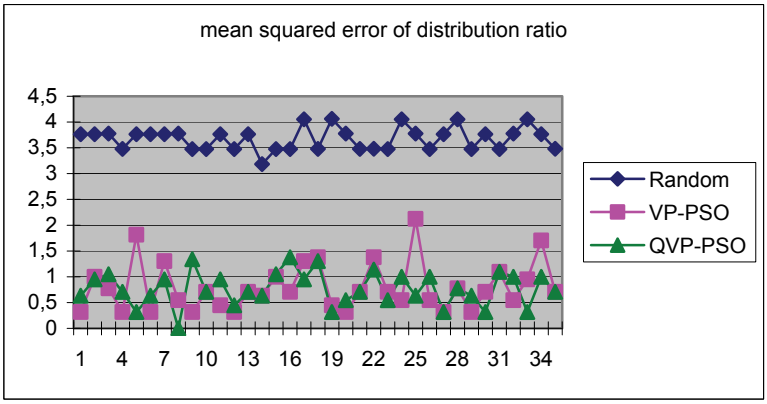


Figure 5. The mean squared error of the robot/target distribution ratio vs. run times

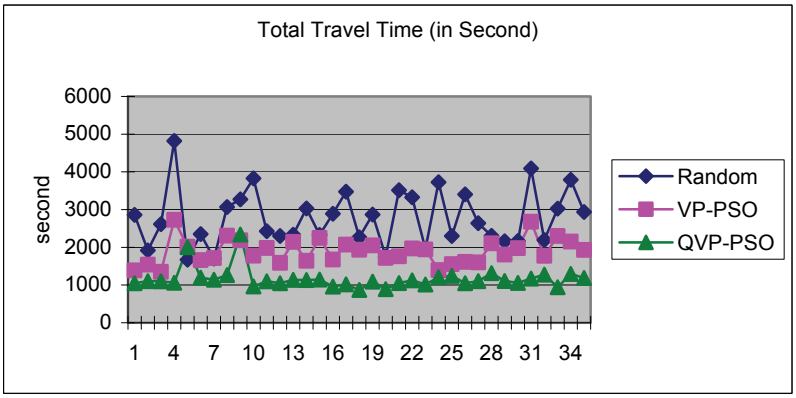


Figure 6. Total travel time of all robots vs. run times

It is obvious that the random approach takes much longer time than the other two approaches and obtains the bigger error in the robot/target distribution ratio. Although sometimes the QVP-PSO algorithm and VP-PSO algorithm have the similar performance under specific target configuration, overall the QVP-PSO algorithm outperforms the VP-PSO algorithm in the search ending time, the total travel time, and the robot/target distribution errors under different target configurations.

6. Conclusion

The proposed QVP-PSO algorithm has the following characteristics: (1) Robots act independently, asynchronously, and in parallel, without maintaining a global model; (2) Robots use a simple control algorithm regardless of the changes under a dynamic environment; and (3) Robots can only communicate with their neighbors to share information; (4) the randomness of the robot movement has been reduced to achieve more reasonable robot/target distribution. Compared to the VP-PSO method, extensive simulation results demonstrate the higher performance of the QVP-PSO algorithm in three different measurements, i.e., search ending time, total travel time of robots (which is directly related to the overall power consumption of the system), and the robot/target distribution ratio errors.

However, due to the dynamic characteristics of this distributed multi-robot system, it is difficult to estimate the target utility value with high accuracy because each individual robot makes its own decision independently based on its local view and some neighboring view. The robot sensor may also bring the noise for the target detection. Probability approaches tend to be more robust in the face of sensor limitation and model limitations. To improve the target detection rate and system robustness, the probability-based approaches will be investigated in the future.

7. References

Balch, T & Arkin, R. C. (1999). Behavior-based Formation Control for Multi-robot Teams, *IEEE Trans. on Robotics and Automation*.

Bersini, H. and Varela, F.J. (1991), The immune recruitment mechanism: a selective evolutionary strategy. In *Proc. Fourth Int. Conf. Genetic Algorithms*. San Mateo, CA: Morgan Kaufmann, 1991, pp.520-526.

Burgard, W., Moors, M., Stachniss, C., and Schneider, F. E. (2005). Coordinated Multi-Robot Exploration, *IEEE Trans. on Robotics, Vol. 21, No. 3.*

Chaimowicz, L.; Kumar, V., and Campos, M. F. (2004). A Paradigm for Dynamic Coordination of Multiple Robots, *Autonomous Robots*, Volume 17, Issue 1, July 2004: pp. 7-21.

Correll, N. and Martinoli, A. (2006). System Identification of Self-Organizing Robotic Swarms, in t*he 8th Int. Symp. on Distributed Autonomous Robotic Systems (DARS)*, Distributed Autonomous Robotic Systems, pp. 31-40, Springer Verlag, 2006.

Dias, M.; Zinck, M., Zlot, R. and Stentz, A. (2004). Robust Multirobot Coordinate in Dynamic Environments, in *Proceedings of IEEE International Conference on Robotics and Automation*, 2004, pp.3435 – 3442.

Dorigo, M.; Maniezzo, V., and Colorni, A. (1996). Ant System: Optimization by a Colony of Cooperating Robots, *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics*, Vol. 26, No. 1, February 1996.

Dorigo, M.; Trianni, V., Sahin, E. (2004). Evolving Self-Organizing Behaviors for a Swarm-bot. *Autonomous Robots*, Vol. 17, No.2-3, 2004. pp. 223-245.

Fernandez, F., Borrajo, D., and Parker, L. E. (2005). A Reinforcement Learning Algorithm in Cooperative Multi-Robot Domains, *Journal of Intelligent and Robotic Systems*, vol. 43, nos. 2-4, 2005: 161-174.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor.

Holland, O.E. and Melhuish, C. (1999). Pheromone, Self-Organization, and Sorting in Collective Robotics, *Artificial Life*, Vol. 5, pp. 173-202, 1999.

Kennedy J. and Eberhart, R. (1995). Particle Swarm Optimization, *IEEE Conference on Neural Networks*, Proceedings 1995.

Martinoli, A.; Ijspeert, A. J. and Mondada, F. (1999). Understanding Collective Aggregation Mechanisms: From Probabilistic Modeling to Experiments with Real Robots, *Robotics and Autonomous Systems*, Vol. 29, pp. 51-63.

Meng, Y.; Kazeem, O., and Muller, J. (2007). A Hybrid ACO/PSO Control Algorithm for Distributed Swarm Robots. *2007 IEEE Swarm Intelligence Symposium*, Hawaii, USA.

Meng, Y. and Gan, J. (2007). LIVS: Local Interaction via Virtual Pheromone Coordination in Distributed Search and Collective Cleanup. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2007)*, Oct. 29-Nov.2, San Diego, CA, USA.

Nolfi, S. and Floreano, D. (2000). *Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines*. MIT Press, Cambridge, MA.

Parker, C.A.C. and Zhang, H. (2006). Collective Robotic Site Preparation. *Adaptive Behavior*. Vol.14, No. 1, 2006, pp. 5-19.

Payton, D.; Daily, M., Estowski, R., Howard, M., and Lee, C. (2001). Pheromone Robotics. *Autonomous Robots*, Vol. 11, No. 3.

Pugh, J. and Martionli, A. (2006). Multi-robot learning with particle swarm optimization, *Proceedings of the Fifth International Joint Conference on Autonomous Robots and Multirobot Systems*", AAMAS, 2006, Hakodate, Japan, pp. 441-448.

Reynolds, C. W. (1987). Flocks, Herds, and Schools: A Distributed Behavioral Model. *Computer Graphics*, 21(4), July 1987, pp. 25-34.

Rechenberg, I.. (1973). *Evolutionsstrategie*. Stuttgart: Fromman-Holzbog.

Rumelhart, D. E. and McLelland, J. H. (1986). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. Cambridge, MA: MIT Press.

Spector, L.; Klein, J., Perry, C., and Feinstein, M. D. (2003). Emergence of collective behavior in evolving populations of flying robots. In *E. Cantu-Paz et. al. editor. Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2003)*, Berlin, Germany, 2003, pp.61-73.

Stewart, R. L. and R. A. Russell. (2006). A Distributed Feedback Mechanism to Regulate Wall Construction by a Robotic Swarm. *Adaptive Behavior*. 14(1):21-51.

Ward, C. R..; Gobet, F., and Kendall, G. (2001). Evolving collective behavior in an artificial ecology. *Artificial Life*, Vol. 7, No. 2, 2001, pp.191-209.

Weigel, T.; Gutmann, J. S., Dietl, M., Kleiner, A., and Nebel, B. (2002). CS Freiburg: Coordinating Robots for Successful Soccer Playing. Special Issue on Advances in Multi-Robot Systems, T. Arai, E. Pagello, and L. E. Parker, Editors, *IEEE Trans. on Robotics and Automation*, Vol. 18, No.5, pp. 685-699.

Werfel, J. and Nagpal, R. (2006). Extended Pheromone in Collective Construction. *IEEE Intelligence Systems*, Vol. 21, No.2, pp. 20-28.

9

Appearance-Based Processes in Multi-Robot Navigation

Luis Payá, Oscar Reinoso, José L. Aznar, Arturo Gil and José M. Marín
Miguel Hernández University
Spain

1. Introduction

When a mobile robot has to carry out an assigned task in a large environment, it has to take decisions about its localization and about the trajectory to follow to move from its current position to the destination point. To solve this problem a map or an internal representation of the environment is needed. This map should contain enough information to allow the robot to compute its current position and the necessary control action to lead it to its destination following, maybe, a chosen trajectory. In the last years, intensive research on this field, using SLAM techniques (Simultaneous Localization And Mapping) has been developed. This approach tries to build a global map of the environment while simultaneously determining the location of the robot. Usually, these approaches rely on the extraction of several landmarks or characteristic points of the environment either natural or artificial, as (Thrun, 2001) does.
Recent research in robotics has extended these concepts to applications that require the use of a team of robots. These applications require the coordination between the members of the team while they move within the environment. In some applications, the use of collaborative robotics clearly improves the performance, comparing to a single robot carrying out the same task. As an example, (Thrun, 2003) presents a probabilistic EKF algorithm where a team of robots builds a map online, while simultaneously they localize themselves. (Fenwick et al., 2002) take into account the problem of the concurrent mapping and localization with extra positional information available when multiple vehicles operate simultaneously. In (Ho & Newman, 2005), a map is built using visual appearance. From sequences of images, acquired by a team of robots, subsequences of visually similar images are detected and finally, the local maps are joined into a single map.
A typical problem in collaborative robotics implies a path following e.g. to perform a surveillance task in an office environment or an assembly or delivery task in an industrial environment. Also, the problem of formations, where a team of robots must navigate keeping a relative position in a structure of robots, can be seen as a problem of path following, where one or several robots must follow the path the leader is recording with an offset either in space or in time.
This problem of route following can be solved without necessity of creating complex maps of the environment. It is just needed a teaching step, where the route to follow is learned, and a navigation step, where the second robot follows the route just comparing its current sensory information with the data stored in the database. Classical approaches in this field

are model-based approaches. In these approaches, the extraction of several landmarks or feature points along the images permits computing the image Jacobian, which relates the change of the coordinates in the image with the changes in motion in the ground plane. Then, using the principles of visual servoing, the second robot can follow the route, as in (Burschka & Hager, 2001). Also, in the behaviour-based control (Balch & Arkin, 1998), some features of the images are extracted to carry out the localization and navigation of the members of a team in a formation problem.

However, other approaches suggest that these processes could be achieved just comparing the general visual information of the images, without necessity of extracting any feature. These appearance-based approaches are specially useful for complicated scenes in unstructured environments where appropriate models for recognition are difficult to create. As an example, (Matsumoto et al., 1996) and (Matsumoto et al., 1999) address a method consisting on the direct comparison of low-resolution images. This method may lead to errors when the size of the route is quite long so, other features must be added to make the method more robust, such as histogram, texture and density of edges, (Zhou et al., 2003). However, these features do not contain any geometric information so they are useful just for localization but not for navigation. To solve it, in some occasions, a feature selection is accomplished to determine the relationship between the images (Boorj et al., 2006).

When working with the whole images, the complexity of the problem can be reduced by means of the PCA (Principal Components Analysis) subspace as in (Kröse et al., 2004) or (Maeda et al., 1997), where through PCA techniques a database is created using a set of views with a probabilistic approach for the localization. Other works have shown how this subspace reduction can be applied to other tasks in robotics, such in loop closure detection for SLAM tasks (Newman et al., 2006). In classical PCA approaches, all the views along the route must be available before the database can be created so the navigation of the second robot cannot begin until the leader has finished learning the route. Actually, a new model must be built from the scratch when we want to include information about new locations in the map. These problems can be overcome using an incremental PCA method, as shown in (Payá et al., 2007), or other kind of techniques to describe the environment, like working in the Fourier domain (Menegatti et al., 2004).

In this paper, we present an appearance-based method for route following where incremental PCA and Fourier Transform have been used to build the database, and a probabilistic Markov process has been implemented for robot localization during the navigation. In the next section, the problem to solve is presented. Then, in section 3, the representation of the environment along the route is detailed. After this, in section 4, the implementation for the navigation is exposed. The results of the experiments carried out with a team of mobile robots are presented in section 5. To finish, the conclusions of the work are exposed.

2. Description of the problem

2.1 Phases to accomplish

The main goal of the work is to create an architecture to carry out multi-robot route following using just visual information and with an appearance-based approach. In this task, a leader robot goes through the desired route while a team of robots follow it. To accomplish this task, two phases must be implemented: the learning phase and the navigation phase.

In the first phase, the leader robot is teleoperated through the route to follow and it stores some general visual information along this route. In a general way, a new image is acquired when the correlation of the current one goes down a threshold respect the previously stored so, images are stored more frequently when the information changes quicker (e.g. turnings). In this step, it is important to define correctly the representation of the environment to allow that any robot can follow the route of the leader one with an offset either in space or/and in time in an efficient way.

The main disadvantage of the appearance-based techniques is that they require huge amounts of memory to store the necessary information of the environment and they suppose high computational cost to make the necessary comparisons during the autonomous navigation so, the key points of these approaches reside on the type and quantity of information to store and in how to make the comparison between the current view and the stored information to reduce the computation time.

Previous works solved the problem by means of low-resolution images (Payá et al., 2005). The main drawback arises when the scenes used to define the route are highly unstructured and varying. In this case, it is necessary to increase the resolution to get an acceptable correlation in navigation. However, it would suppose an increase on the computational cost, so the average speed of the robot would be limited. In section 3, different alternative methods are proposed which are able to deal with high resolution images without slowing down the performance of the classifier.

On the other hand, once the database is created or while it is being built, during the navigation phase, the second robot is situated in a point near the learned route. Then, it has to recognize which of the stored positions is the nearest to the current one and drive to tend to the route, following it till the end. It must be carried out just comparing its current visual information with the information stored in the database. Two processes must run successively: auto-location and control. During the auto-location, the current visual information is compared with the information in the database, and the most similar point is taken as the current position of the robot. This decision must be made taking into account the aperture problem in office environments. This way, the new position of the robot must be in a neighbourhood of the previous one, depending on the speed of the robot.

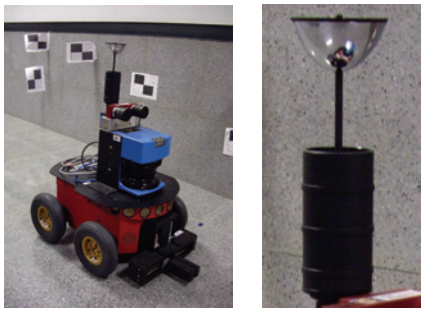


Figure 1. Pionner P3-AT robot and Catadioptric system that provides omnidirectional images

Once we have a new matching, in the control phase, the current visual information must be compared with the matched information of the database, and from this comparison, a control law must be deducted to lead the robot to the route.

In this application, we work with a team of Pioneer P3-AT robots that are equipped with a catadioptric system (fig. 1) consisting on a forward-looking camera and a parabolic mirror that provides omnidirectional images of the environment (fig. 2). To work with this information in an efficient way, the omnidirectional images are transformed to panoramic images, as shown on fig. 3. The size of these panoramic images is 256x64.

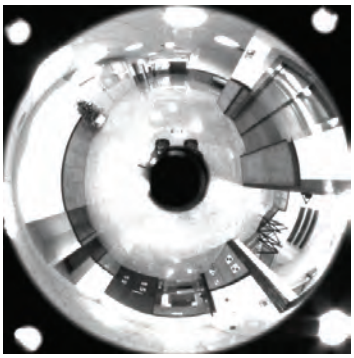


Figure 2. Omnidirectional view of the environment

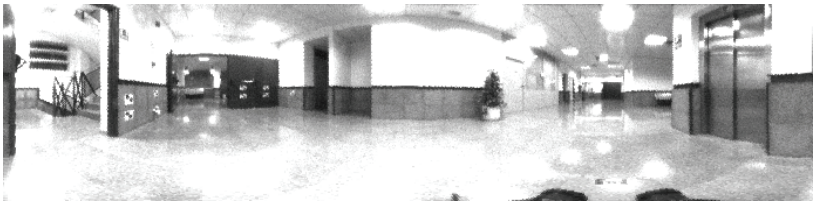


Figure 3. Panoramic view of the environment

3. Representation of the environment

An appearance-based approach for robot navigation implies using the information of the whole images, without extracting any kind of landmark. If the auto-location process described in the previous section was carried out with high resolution images, the computational cost of the classifier would be unbearable. However, it is possible to compress the information of the image with a feature extraction method.

In visual object recognition systems, different techniques have been used to reduce the dimensionality of the data, allowing the classifier to work on a smaller number of attributes. The goal is to discard the irrelevant or redundant data and to keep as much information as possible. The most popular technique for dimensionality reduction purposes is Principal Component Analysis (PCA) (Oja, 1983); (Kirby, 2000). This technique has been used both in general object recognition applied to robotic tasks (Murase & Nayar, 1995); (Nayar et al., 1996) and in face recognition applications (Kirby & Sirovich, 1990); (Turk & Pentland, 1991). Another widely used technique is Linear Discriminant Analysis (LDA) which takes into account class memberships in computing the subspace (Belhumeur et al., 1997); (Swets & Weng, 1996). Dimensionality reduction is in fact a feature extraction process, as new features are extracted from the original attributes.

In our approach, the original images defining the route are compressed using three different techniques: PCA, Incremental PCA and Fourier Transform. This compressed information is used both in the localization and in the autonomous navigation steps.

3.1 PCA

The philosophy of the appearance-based methods consists in working with the general visual information of the images, without extracting any interesting point. Thus, this family of methods presents the disadvantages of the size of the database necessary to retain all the information of the environment and the computational cost of the comparisons between the whole images.

When working with 64x256 images, the data vectors fall in a 16384 dimensional space. However, all these data are generated from a process with just three degrees of freedom (position and orientation of the robot). This way, before storing the images, a reduction of the dimensionality of the data can be performed with the goal of retaining the most relevant information of each scene. Since pixels tend to be very correlated data, a natural reduction step consists on performing Principal Components Analysis (PCA), as in (Kirby, 2000).

Each image $\vec{x}_j \in \Re^{Mx1}; \; j = 1 \ldots N$, being M the number of pixels and N the number of images, can be transformed in a feature vector (also named projection of the image) $\vec{p}_j \in \Re^{Kx1}; \; j = 1 \ldots N$, being K the PCA features containing the most relevant information of the image, $K \leq N$. In traditional PCA, first of all, the data matrix is built using the images of the environment. The PCA transformation is computed from the covariance of the data matrix using SVD and the Turk and Pentland's method (Turk & Pentland, 1991). After the process, a new data matrix with the most relevant information is obtained.

The input data to the PCA transformation is the matrix X:

$$\begin{aligned} \mathbf{X} &= \left[\tilde{x}^1 \ldots | \, \tilde{x}^j \, | \ldots | \, \tilde{x}^N \right] \\ \tilde{x}^j &= \vec{x}^j - \vec{m} \\ \vec{m} &= \frac{1}{N} \sum_{j=1}^{N} \vec{x}^j \end{aligned} \tag{1}$$

where $\vec{x}^j$ is each representative view of the route (arranged in 1-column vector) and $\vec{m}$ is the mean of the set of N views.

The PCA transformation is computed from the covariance of the data matrix using equation 1, e.g. using SVD and the Turk and Pentland's method (Turk & Pentland, 1991). If the size of the data matrix is $M \; x \; N$, where N is the number of images along the route and M is the number of pixels of each image, and also $M >> N$, a total amount of N eigenvectors are obtained. The dimensionality reduction is done by:

$$P = V^T \cdot X \tag{2}$$

where V is the matrix containing the K principal eigenvectors and P is the reduced data of size $K \; x \; N$, where $K \leq N$. After this process, we have reduced each of the original images with M pixels to one vector with K components. The database will consist on these N vectors. Fig. 4 shows the structure of the database for a sample route, taking K=3. In this

case, each image could be reduced to a point in the 3D space. The three coordinates of each point contain the most relevant information of the training images.

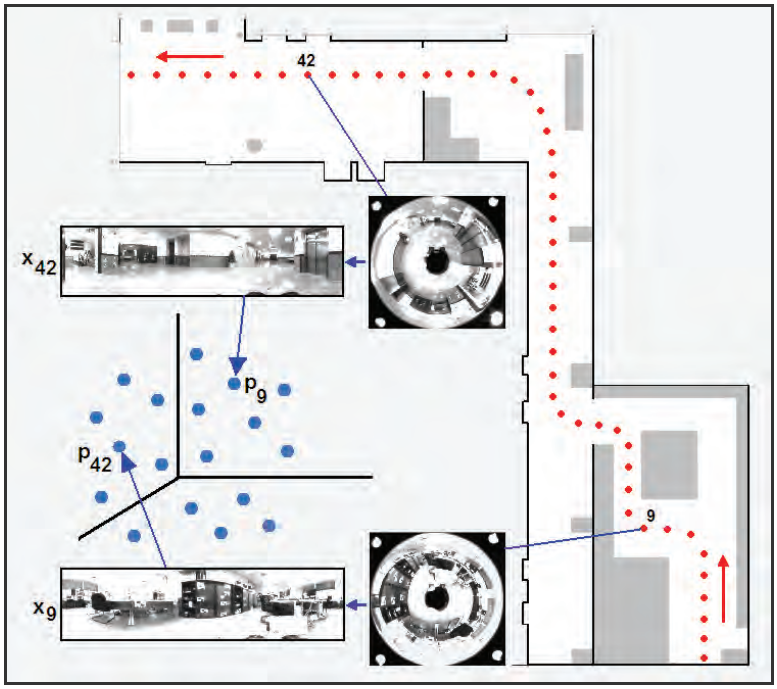


Figure 4. The route images are dimensionally reduced by means of PCA

The projection matrix P has part of the original image information but with lower size vectors, depending on the number of eigenvectors used in the transformation. If $K = N$, the information in P is the same as in X (the reconstruction error is 0 (Kirby, 2000)), but the dimension of the projected vectors is smaller: each projection, p_j, has just K components. Fig. 4 illustrates this approach.

3.2 Incremental PCA

In classical PCA approaches, all the images along the environment must be available before carrying out the compression. This way, the robots that follow the route should wait the leader one to run till the end. However, in collaborative tasks, usually it is necessary that some robots follow the first one while it is still recording the information.

With the PCA approach, the robot that is building the database should do it from the scratch when a new image along the route is captured, what is computationally very expensive. To overcome this disadvantage, a progressive construction of the database can be implemented. Several algorithms have been proposed to perform PCA in an incremental way (Artac et al., 2002).

As can be proved, when having a set of eigenvectors from a set of views, when a new image $\vec{x}_{N+1}$ is added to the database, these eigenvectors and the projections of the stored images can be updated following the next algorithm (Artac et al., 2002):

First, the mean must be updated:

$$\vec{m}' = \frac{1}{N+1}\left(N \cdot \vec{m} + \vec{x}_{N+1}\right) \tag{3}$$

Now, the set of eigenvectors must be updated so that they include the information of the new image $\vec{x}_{N+1}$. To do it, we compute the residual vector, that is the difference between the reconstruction and the original *N+1* image $\vec{h}_{N+1} = \left(V \cdot \vec{p}_{N+1} + \vec{m}\right) - \vec{x}_{N+1}$, that is orthogonal to the eigenvectors, and normalize it, obtaining $\hat{\vec{h}}_{N+1}$.

The new matrix of eigenvectors can be obtained by appending $\hat{\vec{h}}_{N+1}$ to *V* and rotating it:

$$V' = \left[V \cdot \hat{\vec{h}}_{n+1}\right] \cdot R \tag{4}$$

Being *R* the solution to the eigenproblem $D \cdot R = R \cdot \Lambda'$ and *D*:

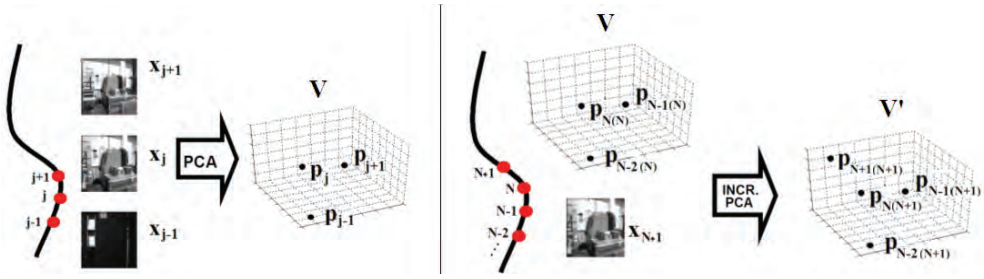
$$D = \frac{N}{N+1} \cdot \begin{bmatrix} \Lambda & \vec{0} \\ \vec{0}^T & 0 \end{bmatrix} + \frac{N}{(N+1)^2} \cdot \begin{bmatrix} \vec{p} \cdot \vec{p}^T & \delta \cdot \vec{p} \\ \delta \cdot \vec{p}^T & \delta^2 \end{bmatrix} \tag{5}$$

where $\delta = \vec{h}_{N+1} \cdot \left(\vec{x}_{N+1} - \vec{m}\right)$, $\vec{p} = V^T \cdot \left(\vec{x}_{N+1} - \vec{m}\right)$ and Λ is a diagonal matrix containing the original eigenvalues. This way, if $V \in \Re^{MxK}$, then $V' \in \Re^{Mx(K+1)}$.

The image representations can be updated with the next expression:

$$\vec{p}_{i(N+1)} = (R')^T \cdot \begin{bmatrix} \vec{p}_{i(N)} \\ 0 \end{bmatrix} + \left[V \cdot \vec{h}_{N+1}\right]^T \cdot \left(\vec{m} - \vec{m}'\right) \tag{6}$$

Fig. 5(b) shows the input data and the results of this approach in each iteration, comparing to the classical PCA (Fig. 5(a)).



(a) (b)

Figure 5. (a) Using PCA, the images of the route are dimensionally reduced. (b) Input and output data of the Incremental PCA process in each iteration. When a new image arrives. the projections in the database and the matrix *V* are updated

3.3 Fourier Transform

Another compact representation that we can use is a representation in the Fourier domain (Menegatti et al., 2004). To do it, each omnidirectional image is transformed into the

panoramic one. Then, this panoramic image is expanded row by row into its Fourier Transform.

The sequence of complex numbers $\{a_n\}=\{a_0, a_1, \ldots, a_{N-1}\}$ can be transformed into the sequence of complex numbers $\{A_n\}=\{A_0, A_1, \ldots, A_{N-1}\}$ using the Discrete Fourier Transform with the following expression:

$$A_k = \sum_{n=0}^{N-1} a_n \cdot e^{-j\cdot\frac{2\pi nk}{N}}; \qquad k = 0, ..., N-1 \tag{7}$$

The Fourier Transform computed in this way, row to row, presents two interesting properties for robot mapping and localization when omnidirectional images are used. First, the most relevant information in the Fourier domain concentrates in the low frequency components. These are the components whose magnitude is highest, as shown on fig. 7. This way, the information of the image is compressed on these components. Furthermore, removing high frequency information can lead to an improvement in localization because these components are more affected by noise.

The second interesting property is the rotational invariance. If two images are acquired at the same point of the environment but with different headings for the robot, then, the power spectrum of each row (the module of the Fourier Transform) is the same in both cases. This is because these two rotated omnidirectional images lead to two panoramic images THAT are the same but shifted along the horizontal axis (fig. 6). Each row of the first image can be represented with the sequence $\{a_n\}$ and each row of the second image will be the sequence $\{a_{n-q}\}$, being q the amount of shift, that is proportional to the relative rotation between images.

The rotational invariance is deducted from the shift theorem, which can be expressed as:

$$\Im\left[\{x_{n-q}\}\right] = X_k e^{-j\cdot\frac{2\pi qk}{N}}; \qquad k = 0, ..., N-1 \tag{8}$$

where $\Im\left[\{x_{n-q}\}\right]$ is the Fourier Transform of the shifted sequence, and X_k are the components of the Fourier Transform of the non-shifted sequence. According to this expression, the Fourier Transform of a shifted sequence of numbers is equal to the Fourier Transform of the original signal multiplied by a complex number whose magnitude is the unit. This means that the amplitude of the Fourier Transform of the shifted image is the same as the original transform and there is only a phase change, proportional to the amount of shift q.

This way, when a new omnidirectional image is captured, first it is transformed into the panoramic image and then, the Fourier transform of each row is computed. This way, each 64x256 image is compressed to a sequence of 64xF numbers that are the magnitudes of the Fourier Transform components.

Each image $[x]_j \in \Re^{RxC}; j = 1 \ldots N$, being R the number of rows and C the number of columns, is transformed into a new matrix $[t]_j \in \Re^{RxF}; j = 1 \ldots N$, being F the first Fourier components (corresponding to the lower frequencies).

It must be taken into account that the Fourier Transform is intrinsically, an incremental procedure, due to the fact that to compress each scene it is not necessary any kind of

information from the rest of scenes. This way, it also permits multi-robot navigation as Incremental PCA did and with the important property of rotational invariance.

Figure 6. Two panoramic images taken in the same point but with different heading for the robot

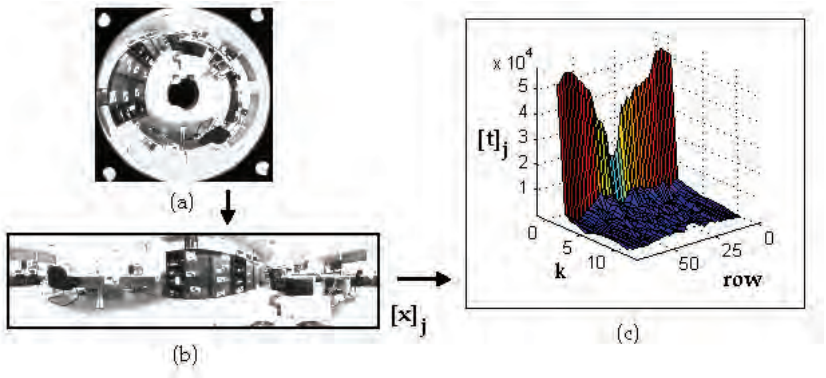


Figure 7. For each point on the route, the omnidirectional view (a) is transformed into the panoramic view (b). Then, the Fourier transform is computed row to row (c). The main information is in the module of the components of lower frequency so these components are retained in a matrix that identifies that point of the route

4. Robot navigation

After the Learning Step, the leader robot has created a database consisting on some feature vectors that are labelled as positions *1,..., N*. For each position, two feature vectors represent the visual information taken at that position: the PCA projection and the Fourier Transform. Now, a different robot can follow the same route carrying out two processes: Auto-location and Control.

4.1 Auto-Location

When the follower robot captures a new image $[x]_i$, using this information it must know which of the stored points is the closest one.

Using the incremental PCA approach, first, the projection $\vec{p}_i$ of the current image on the current eigenspace calculated by the leader is computed. Then, this vector has to be compared with those stored in the database $\vec{p}_j$. The one that offers the minimum distance is the matching one, which corresponds to the current position of the robot. The criterion used here is the Euclidean distance.

$$d_{ij} = \sqrt{\sum_{l=0}^{k} \left(p_{il} - p_{jl}\right)^2}; \quad j = 1 \ldots N \tag{9}$$

On the other hand, if we use the Fourier Transform approach, once the robot captures a new image, its Discrete Fourier Transform is computed row to row. This results in a new matrix $[t]_i$ that contains the magnitude of the low-frequency components of the transformed sequence. Then, this matrix is compared with the Fourier matrices in the database $[t]_j$ with the expression:

$$d_{ij} = \sum_{m=0}^{R-1} \sqrt{\sum_{l=0}^{F-1} \left(t_i^{ml} - t_j^{ml}\right)^2}; \quad j = 1 \ldots N \tag{10}$$

To normalize the values of the distances, a degree of similarity between projections is computed with eq. (11). The image of the database that offers the maximal value of γ_{ij} is the matching image, and then, the current position of the robot (the nearest one) is *j*.

$$\gamma_{ij} = \frac{\left(d_i^{\max} / d_{ij}\right) - 1}{\left(d_i^{\max} / d_i^{\min}\right) - 1}; \quad \gamma_{ij} \in [0,1]; \quad j = 1 \ldots N \tag{11}$$

where $d_i^{\max} = \max_{j=1}^{N}\left(d_{ij}\right)$ and $d_i^{\min} = \min_{j=1}^{N}\left(d_{ij}\right)$.

In office environments, that present a repetitive visual appearance, this simple localization method tends to fail often as a result of the aperture problem. This means that the visual information captured at two different locations that are far away can be very similar. To avoid these problems, a probabilistic approach, based on a Markov process, has been used. The current position of the robot can be estimated using the Bayes rule, with the next expression:

$$p(y|x;\theta) \propto p(x|y;\theta) \cdot p(y) \tag{12}$$

where *p(y)* denotes the probability that the robot is on the position *y* before observing the image *x*. This value is estimated using the previous information and the motion model. *p(x | y)* is the probability of observing *x* if the position of the robot is *y*. This way, a method to estimate the observation model must be deducted. In this work, the distribution *p(x | y)* is modelled through a sum of Gaussian kernels, centred on the *n* most similar points of the route:

$$p\left(x_i \mid y_j\right) = \frac{1}{n} \cdot \sum_{j=1}^{n} \left(\gamma_{ij} \cdot e^{-\left(\frac{y - y_j}{\sigma}\right)^2} \right); \quad j = 1 \ldots N \tag{13}$$

Each kernel is weighted by the value of confidence γ_{ij}. Then, these kernels are convolved with a Gaussian function that models the motion of the robot, and that depends on the previous position and velocity of the robot. At last, the new position is considered at the point with highest contribution probability $\max_{j=1}^{N}\left(p\left(x_i \mid y_j\right)\right)$.

Fig. 8 shows this process for *n=10* kernels. The previous position is the 28th. First, the ten most similar positions are selected. Then, a kernel function is assigned to each position. After that, the motion model is applied and at the end, the contribution of each kernel to each position is computed, selecting the point with the maximum contribution. In this case, despite the most similar location is the 10th, after the localization process the position selected would be the 28th.

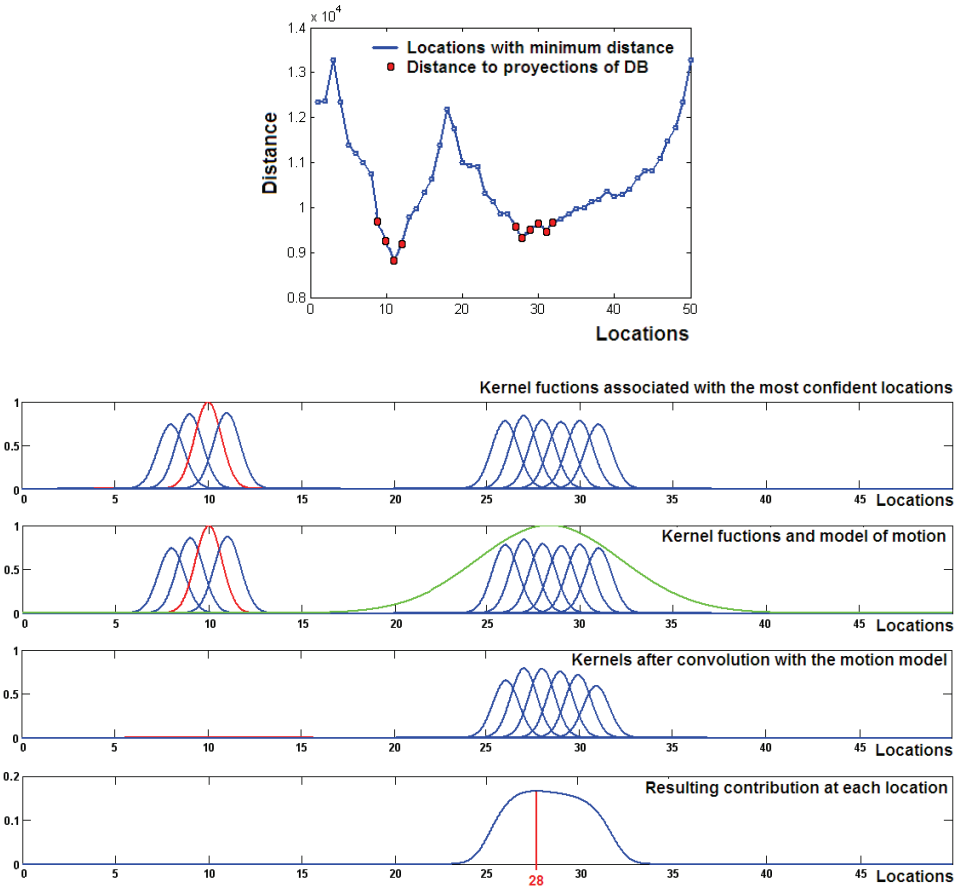


Figure 8. Improvement of the localization phase through a probabilistic-based approach. The new location depends on the previous one and the image acquired

After several experiments, the Fourier Transform approach has proved to be a more robust way to compute the current position of the robot. During a typical navigation problem, the

time needed to compute the feature vector, compare with all of the vectors in the database and extract the matching point with the probabilistic approach is about 25% lower, comparing to the incremental PCA approach.
Also, the rotational invariance of the Fourier Transform is an important property when the robot starts the navigation. If at first, the follower robot is near the route but its heading is very different to the heading of the leader, the PCA approach would not be able to retrieve the correct position, because it does not present rotational invariance. Opposite to it, with the Fourier approach, the position is computed correctly so the robot is able to correct its heading and tend to the route. In the final implementation, only the Fourier Transform has been used for Localization.

4.2 Control

Once the robot knows its position, it has to compute its heading, comparing to the original heading that the leader had when it captured the image at that position. Then, the steering speed of the robot should be proportional to this relative heading.
When working in the Fourier domain, eq. 8 allows computing the shift q and so, the orientation of the robot. However, after some localization experiments, this expression has showed a very unstable behaviour, and it has not been able to deal with changes in illumination and noise. It only works well under very controlled conditions and if the follower robot is exactly on the same point where the leader captured the image.

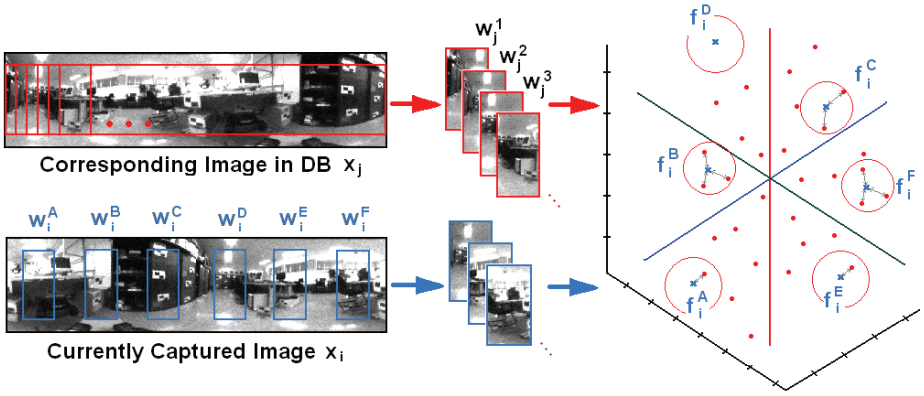
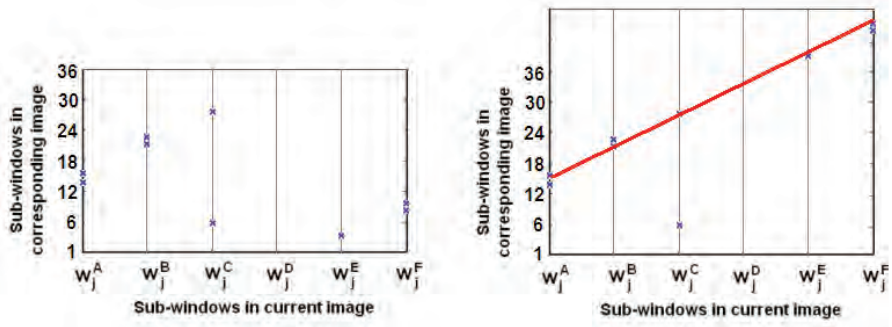


Figure 9. Calculating the linear and steering speeds of the follower robot. When it captures a new image, six sub-windows are extracted and tracked over the corresponding image

The procedure used here to retrieve the orientation is the following. From each image stored in the database, j, a set of N' sub-windows is obtained from the whole image where $\vec{w}_j^i \in \Re^{M'x1}$, is each sub-window. The sub-windows are obtained scanning the original scene with a step in the horizontal axis (fig. 9). Carrying out a process of PCA compression, the PCA components $\vec{f}_j^i \in \Re^{K'x1}$, of each sub-image are calculated, where $K' \leq N'$. Fig. 9 shows these projections as red dots in the case K'=3. During the autonomous navigation, six sub-windows $\left(\vec{w}_j^A, \vec{w}_j^B, \vec{w}_j^C, \vec{w}_j^D, \vec{w}_j^E, \vec{w}_j^F\right)$ are taken on the currently captured view (fig. 9) and tracked over the central band of the matching image. To do this, once the robot knows its

location, the PCA components of these six sub-windows are calculated. This operation returns six *K*'-components vectors $\left(\vec{f}_j^{\,A}, \vec{f}_j^{\,B}, \vec{f}_j^{\,C}, \vec{f}_j^{\,D}, \vec{f}_j^{\,E}, \vec{f}_j^{\,F}\right)$ that are shown as blue crosses on fig. 9. Then, the most similar projections to each of them are extracted. These most similar projections are those that fall in six spheres whose centres are $\left(\vec{f}_j^{\,A}, \vec{f}_j^{\,B}, \vec{f}_j^{\,C}, \vec{f}_j^{\,D}, \vec{f}_j^{\,E}, \vec{f}_j^{\,F}\right)$. The radius of these spheres is chosen so that a number of corresponding windows is extracted. In this work, a total number of ten sub-windows are extracted. The linear and steering velocities are inferred using a controller, whose inputs are the most similar projections to each of the six sub-windows. Analyzing these data and solving possible inconsistencies, the controller infers the linear and steering velocities of the robot to tend to the recorded route. To do it, once the most similar sub-windows are recognized, the controller tries to arrange them and look for a correlation that shows clearly if the robot has to turn left or to turn right to tend to the pre-recorded route.



(a) (b)

Figure 10. Computing the steering speed of the robot with the sub-windows approach. This speed is proportional to the ordinate in the origin of the fitted line

Fig. 10 shows an example of how the controller works. In fig 10(a), the blue crosses are the most similar sub-windows. On this figure, the most similar windows to $\vec{w}_j^{\,A}$ are the windows 15 and 16 $\left(\vec{w}_j^{15}, \vec{w}_j^{16}\right)$, the most similar to $\vec{w}_j^{\,\mathbf{B}}$ are the windows 22 and 23, $\left(\vec{w}_j^{22}, \vec{w}_j^{23}\right)$, the most similar to $\vec{w}_j^{\,C}$ are 6 and 28 $\left(\vec{w}_j^{6}, \vec{w}_j^{28}\right)$, $\vec{w}_j^{\,D}$ has no correspondences, the most similar to $\vec{w}_j^{\,E}$ is the 3rd window $\left(\vec{w}_j^{3}\right)$ and at last, the most similar to $\vec{w}_j^{\,F}$ are the windows 9 and 10, $\left(\vec{w}_j^{9}, \vec{w}_j^{10}\right)$. This distribution of correspondences shows that the robot has to turn left so that the sub-windows fit with those of the corresponding image. After doing a least-square fitting with these points, the change in the heading of the robot should be proportional to the ordinate in the origin of the adjusted line, so that the robot tends to the position which the image was originally captured with. However, some considerations must be taken into account.

1. Usually, the line presents a discontinuity, because of the periodicity of the panoramic images. Then, it is necessary to correct some of the correspondences so that the fitting is correct, as shown on fig. 10(b).
2. Some correspondences may be outliers due to partial occlusions or little changes in the environment (position of the objects or illumination). These points should not be taken into account while computing the fitting.

3. As the six sub-windows are taken with a constant offset, the slope of the fitted line has to be near to N'/6. It must be taken into account while fitting the line.
4. The ordinate at the origin of the chosen linear regression shows how the steering of the robot should be to tend to the route correctly.

It must be taken into account that the linear speed must depend on the steering. If the steering is high, it means that the robot heading is very different to the heading the leader had, so the linear speed has to be low to permit the correction in the heading.

5. Results

Several experiments have been carried out to validate the approach. Fig. 11 shows a typical route of around 50 meters, recorded in an office environment and the route of the follower when it starts from two different points around it. Typically, the follower robot tends to the route and follows it, showing a great performance on the straight lines and a relatively bigger error in the turnings. However, with this approach, the robot is able to find the route and tend to it, showing a very stable behaviour till the end. Fig. 12 shows the evolution of the localization during the navigation of the follower robot and the probability calculated, what can be a measure of the precision.

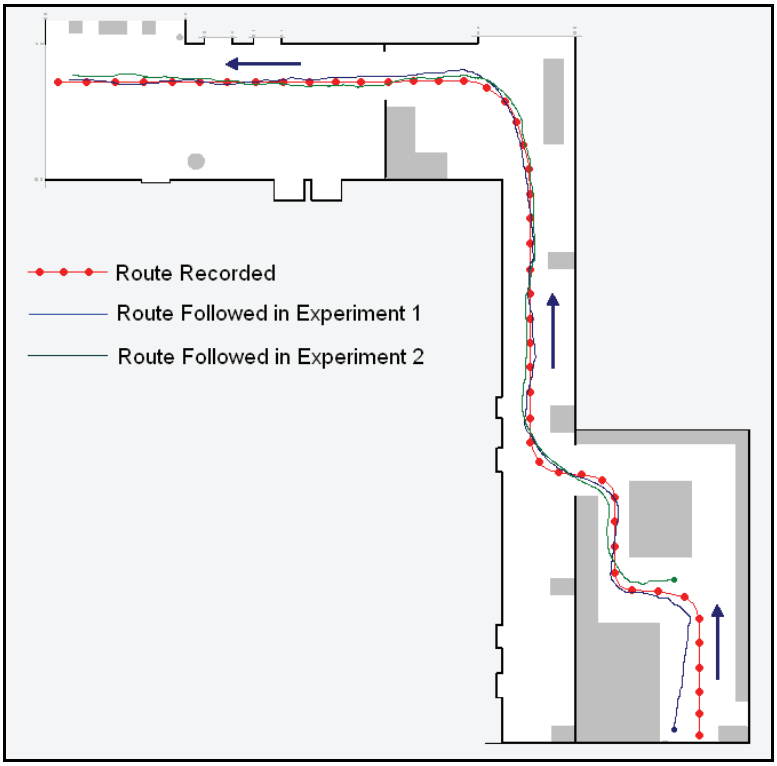


Figure 11. Route recorded by the leader robot and routes followed by a follower robot in two experiments with different initial point

To carry out these experiments, two Pioneer P3-AT robots have been used with two processors onboard that communicate using a CORBA-based architecture where they interchange the necessary information. It is important to design an application where the different robots can share the necessary information in an easy and quick way due to the fact that the follower robot has to use continuously the database that the leader one is computing. An additional processor has been added to the architecture to carry out some calculations to reduce the computational cost of the processes in the robots.
On fig. 12, the localization shows a correct evolution, despite the aperture effect in such office environments, and the robot recovers correctly of some errors in localization (such as those in images 100 and 170). Also, the probability begins with quite low value (the robot is far from the route). Then, it tends to increase when the robot approaches to the straight line and decreases again in the turnings.
In the second experiment, the robot was initially close to the route, but it has a 180º orientation respect the leader. Despite this fact, the robot is able to localize itself using the Fourier approach, correct its heading, tend to the route and follow it.
Also, the sub-windows approach has showed to be robust when partial occlusions appear or when the original environment suffers small changes.

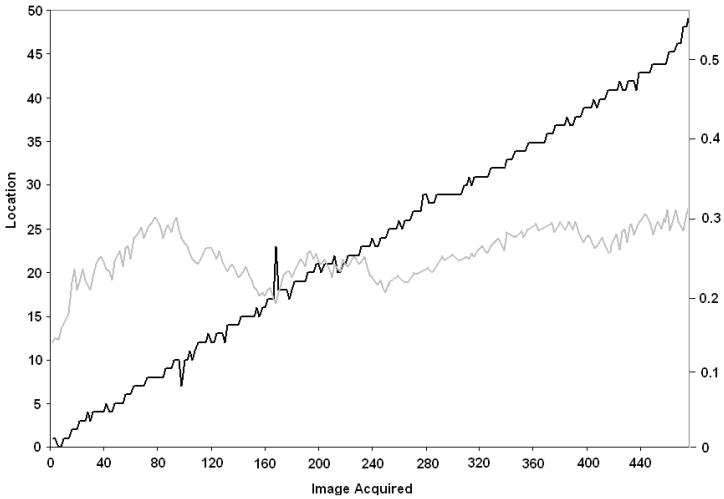


Figure 12. Evolution of the localization and the final probability $p(x \mid y)$ in the route followed during the experiment 1

6. Conclusions

In this paper, an appearance-based multi-robot following-route scheme is presented. The proposed solution uses low resolution panoramic images obtained through a catadioptric system, and techniques for dimensionality reduction to extract the most relevant information along the environment. To allow a new robot can follow the route that another robot is recording at the same time, an incremental PCA and a Fourier Domain algorithm are presented.

The objective of the work is that other robots can follow this route from a distance (as in space or in time). To do it, a probabilistic algorithm has been implemented to calculate their current position among those that the leader has stored, and a controller has been implemented, also based on the appearance of the scenes, to calculate the linear and turning speeds of the robot.
Some experiments have been carried out with two Pioneer 3-AT robots using a CORBA-based architecture for communication. These experiments show how the process employed is useful to carry out route following in an accurate and robust way.
We are now working in other control methods to reduce the error during the navigation, studying the effects of illumination changes and occlusions more accurately. Also, other techniques to compress the information are being analysed to achieve a higher speed of the follower robots.
At last, more sophisticated ways of building a map are being evaluated so that the robot can find the route and follow it although its initial position is far from this route. These more complex maps are expected to be useful in a number of applications in multi-robot navigation.

7. Acknowledgements

This work has been supported by MEC through project DPI2007-61197 "Sistemas de percepción visual móvil y cooperativo como soporte para la realización de tareas con redes de robots".

8. References

Artac, M.; Jogan, M. & Leonardis, A. (2002). Mobile Robot Localization Using an Incremental Eigenspace Model, *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 1205-1030, ISBN 0-7803-8232-3, Washington, USA, May 2002, IEEE

Balch, T. & Arkin, R. (1998). Behavior-based formation control for multi-robot teams. *IEEE Transactions on Robotics and Automation*, Vol. 14, No. 6, (Nov 1998) 926-938, ISSN 1042-296X

Belhumeur, P.N.; Hespanha, J.P. & Kriegman, D.J. (1997). Eigenfaces vs. fisherfaces: recognition using class specific linear projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 19, No. 7, (Jul 1997), 711-720, ISSN 0018-9340

Boorj, D.; Zivkovik, Z. & Kröse, B. (2006). Sparse appearance based modeling for robot localization, *Proceedings of IEEE International Conference on Intelligent Robots and Systems*, pp. 1510-1515, ISBN 1-4244-0259-X, Beijing, China, IEEE

Burschka, D. & Hager, G. (2001). Vision-Based Control of Mobile Robots, *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 1707-1713, ISBN 0-7803-6578-X, Seoul, Korea, May 2001, IEEE

Fenwick, J.; Newman, P. & Leonard, J. (2002). Cooperative Concurrent Mapping and Localization, *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 1810-1817, ISBN 0-7803-8232-3, Washington, USA, May 2002, IEEE

Ho, K. & Newman, P. (2005). Multiple Map Intersection Detection using Visual Appearance, *Proceedings of the Third International Conference on Computational Intelligence, Robotics and Autonomous Systems*, ISSN 0219-6131, Singapore, Dec 2005

Kirby, M. & Sirovich, L. (1990). Application of the Karhunen-Loeve procedure for the characterization of human faces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 12, No.1, (Jan 1990), 103-108, ISSN 0018-9340

Kirby, M. (2000). *Geometric data analysis. An empirical approach to dimensionality reduction and the study of patterns*, Wiley Interscience, ISBN 0-4712-3929-1

Kröse, B.; Bunschoten, R.; Hagen, S.; Terwijn, B. & Vlassis, N. (2004). Household robots: Look and learn. *IEEE Robotics & Automation magazine*, Vol. 11, No. 4, (Dec 2004) 45-52, ISSN 1070-9932

Maeda, S.; Kuno, Y. & Shirai, Y. (1997). Active navigation vision based on eigenspace analysis, *Proceedings of IEEE International Conference on Intelligent Robots and Systems*, pp. 1018-1023, ISBN 0-7823-0123-4, Grenoble, Francia, Sep 1997, IEEE

Matsumoto, Y.; Inaba, M. & Inoue, H. (1996). Visual navigation using view-sequenced route representation, *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 83-88, ISBN , Minneapolis, USA, IEEE

Matsumoto, Y.; Ikeda, K.; Inaba, M. & Inoue, H. (1999). Visual Navigation Using Omnidirectional View Sequence, *Proceedings of IEEE International Conference on Intelligent Robots and Systems*, pp. 317-322, ISBN 0-7823-5184-3, Korea, Oct 1999, IEEE

Menegatti, E.; Maeda, T. & Ishiguro, H. (2004). Image-based memory for robot navigation using properties of omnidirectional images. Robotics and Autonomous Systems. Vol. 47, No. 4, (Jul 2004), 251-276, ISSN 0921-8890

Murase, H. & Nayar, S.K. (1995). Visual learning and recognition of 3D objects from appearance. *International Journal on Computer Vision*, Vol. 14, No. 1, (Jan 1995), 5-24, ISSN 0920-5691

Nayar, S.K.; Nene, N.A. & Murase, H. (1996). Subspace methods for robot vision, *IEEE Transactions on Robotics and Automation*, Vol. 12, No. 5, (Oct 1996) 750-758 ISSN 0882-4967

Newman, P.; Cole, D. & Ho, K. (2006). Outdoor SLAM using Visual appearance and laser ranging, *Proceedings of the International Conference on Robotics and Automation*, pp. 1180-1187, ISBN 0-7803-9506-9, Orlando, USA, May 2006, IEEE

Oja, E. (1983). *Subspace methods of pattern recognition*, Research Studies Press and J. Wiley, ISBN 0-8638-0010-6.

Payá, L.; Vicente, M.A.; Navarro, L.; Reinoso, O.; Fernández, C. & Gil, A. (2005). Continuous navigation of a mobile robot with an appearance-based method, *Proceedings of the second International Conference on Informatics in Control, Automation and Robotics*, pp. 443-446, ISBN 972-8865-30-9, Barcelona, Spain, Sep 2005

Payá, L.; Reinoso, O.; Gil, A.; Pedrero, J. & Ballesta, M. (2007). Appearance-Based Multi-Robot Following Routes Using Incremental PCA, *Proceedings of Knowledge-Based Intelligent Information and Engineering Systems*, pp. 1170-1178, ISBN 3-540-74828-8, Vietri sul Mare, Italy, Sep 2007, Springer

Swets, D.L. & Weng, J.J. (1996). Using discriminant eigenfeatures for image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 18, No. 8, (Aug 1996), 831-836, ISSN 0018-9340

Thrun, S. (2001). A probabilistic online mapping algorithm for teams of mobile robots. *International Journal of Robotics Research*, Vol. 20, No. 5, (2001), 335-363, ISSN 0278-3649

Thrun, S. (2003). Robotic Mapping: A Survey, In: *Exploring Artificial Intelligence in the New Milenium*, 1-35, Morgan Kaufmann Publishers, ISBN 1-55860-811-7, San Francisco, USA

Turk, M. & Pentland, A. (1991). Eigenfaces for recognition. *Journal on Cognitive Neuroscience*, Vol. 3, No. 1, (Jan 1991) 71-86, ISSN 0898-929X

Zhou, C.; Wei, T. & Tan, T. (2003). Mobile robot self-localization based on global visual appearance features, *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 1271-1276, ISBN 0-7803-7737-0, Taipei, Taiwan, Sep 2003, IEEE

10

A User Multi-robot System Interaction Paradigm for a Multi-robot Mission Editor

Saidi Francois and Pradel Gilbert
IBISC Laboratory, FRE CNRS 2873, Université d'Evry-Val-d'Essonne
France

1. Introduction

Many applications such as space and underwater exploration, operations in dangerous environment, service robotics, military applications, etc. can call upon multi-robot systems. These systems, although far from achievement, can carry out difficult or even tasks impossible to achieve by a single robot. A team of robots provides a certain redundancy, contributes to the achievement of a task in a collaborative way and should be able to go beyond what could be done by a single robot.
According to (Parker, 2000) and (Arai & al., 2002), works in collective mobile robotics can be classified in three categories:

- Reconfigurable robots systems also called "Cellular Robots Systems". A cellular robot is an auto-organized robot-like system composed of a large number of units called cells. This idea is inspired by the organization of a living system. Various fields were studied in this domain, in particular the swarm intelligence (Bonabeau & Theraukaz, 2000) (Beni & Hackwood, 1992) on the "cyclic swarms". One will also note the work of Fukuda (Fukada & Nakagawa, 1987) on the system CEBOT.
- Trajectory planning: one can quote the works (Premvuti & Yuta, 1990) on the control of the aircraft traffic, of (Arai & al., 1989) and of (Wang, 1989) on the movement of groups of robots in formation.
- Architectures for multi-robot co-operation among which the system ACTRESS (Asama & al., 1989) (Sellem & Dalgalarrondo 1999). ACTRESS deals with maintenance in a known and structured environment. It resolves conflicts between robots and allocates tasks by creating staffs of robots.

In this paper, our interest is mostly focused on the human-system interactions. Studies on human-system interactions show that, to use efficiently the system, the user has to "accept" it. This acceptance implies that two concepts are to be taken into account: usability and appropriation (section 0 and 0). The user also needs to be permanently informed on the activity of the system, on the individual activities of its entities, on the state of the different components to compare the mental representation of the mission progress he/she has at the beginning of the mission and what is really executed by the system. This is peculiarly true when a user (e.g. a handicapped person, a person with specific needs) requests services to a multi robot system.

In the case of the teleoperation of a system, the user must compensate for sensory impoverishment, replace the sensorimotor mechanisms usually used, and learn new strategies to control the distant system. Initially, researchers sought to make the system the most autonomous possible. They realized that its complexification, in the design and in the realization, led to great difficulties of utilisation by the increase of what one could call the distance between the orders necessary to the machine and the natural reactions of the user. A second approach tends to design machines which interact with the user by studying the distribution of the tasks between the human and the system and how this user uses the system to conceive/improve this one. This use of the system by the operator is typical of the way in which he/she adapted to the machine. This is much more difficult when the system is a multi-robot system. These features led us to consider the system design as a participative system. These participative aspects will involve:

- the Human-System interface (HSI),
- the service request seen as a set of remote services carried out either in an individual or in a collective way by the robots,
- the user intervention in the construction of the solution,
- a way to act directly on the robots or on the mission scene in the event of modification, breakdown or execution failure,
- an always available information on the system state.

This paper presents an original approach of the user/multi-robot system interaction. After a review of the state of the art in collective robotics (section 0), we will introduce our participative model for human system interaction (section 0). This system design will call upon several levels of abstraction, the first level, the semantical and syntactical verificator, is discussed in section 0, the second level, the geometrical verificator, is analysed in section 0. This paper ends with a conclusion and some prospects.

2. Mobile collective robotics

The works of (Balch & Parker, 2002);(Schultz & Parker, 2002);(Parker & al., 2002) define 6 directions of study in multi-robot systems:

1. biologically inspired systems,
2. systems which study the communications,
3. systems which are interested in architectures, tasks allocation and control,
4. systems for objects transport and handling,
5. systems for displacements coordination,
6. systems dealing with the design of reconfigurable robots.

2.1 Biologically inspired systems

Most of multi-robot systems of biological inspiration follow upon the "behaviour-based" paradigm (behavioural robotics) of Brooks (Brooks, 1986). Researchers (Drogoul & Ferber, 1993) were interested in modelling insects or animals societies (ants, bees, birds, fish...) to reproduce their behaviours. Works in this direction (Goldberg & Mataric, 1999) showed the possibility for multi-robot systems to carry out the collective behaviours such as group displacement, gathering, dispersion, foraging. Studies modelling collective behaviour of more advanced animals such as wolves herd showed that the results for archaic animal did not successfully apply in that case. One will also quote the Animatlab approach which

conceives simulated or real artificial systems named "animats" whose behaviours exhibit some animal features (Hallam & al., 2002); (Hallam & al., 2004).

2.2 Systems dealing with the communications

The word "communication" can be understood in two different ways: communication between robots or communication between a user and the multi-robot system. Communications between the various entities of a multi-robot system are a crucial point. One often makes the difference between explicit communication, which is a relational operation between an entity and one or more others, and implicit communication ("through the world") in which an entity broadcasts a message which will be received by all others entities.

The problems involved in the user-system communication are tackled in (Jones & Rock, 2002). The aimed application is the use of robots team in the space construction industry. Dialogues are led by the user with a community of agents through a series of implicit and explicit questions on the robots and the environment state. The operator plays a significant role in the scheduling of the robots tasks. Authors underline the difficulties to:

- establish the structure and the range of the dialogue,
- create an infrastructure which allows for the system/robots to conduct a dialogue with the user,
- determine the methods which can take into account the subjacent social aspect in this kind of dialogue,
- develop an interface which allows the user to dialogue with the system.

In (Fong & al., 2001), authors recommend adapting the autonomy and the human-system interaction to the situation and to the user. According to these authors, part of the decision-making process, which is most of time not structured, must remain in the human domain, in particular because the robots remain very limited for the high level perceptive functions, such as objects recognition and "human-like" interpretation of the environment. Their approach tends to treat the robot not like a tool but like a partner.

2.3 Systems directed towards architectures design, tasks allocation and control

These systems deal with problems common to the decentralized systems: tasks allocation, tasks planning, communication system design, homogeneity or heterogeneity of the robots, delegation of authority, global coherence and local actions... In (Iocchi & al., 2001), multi-robot systems are initially shown like a particular case of multi-agents systems. The authors insist on the fact that a multi-robot system has specific constraints because of the immersion of the agents in a real environment.

Other works as those of Rybski (Rybski & al., 2002) present a software architecture intended for the control of a team of miniature robots. Their low on-board calculation capacity calls upon distant calculators. The used algorithm tries to dynamically allocate the resources to the robots according to their needs and to the evolution of the tasks they are carrying out. Tasks allocation is also discussed in the work of Gerkey and Mataric (Mataric & al., 2002) (Gerkey & Mataric, 2003). This article presents a strategy for tasks allocation by using a form of negotiation to optimize the use of the robot's resources. Experimental results are presented for a mission in which an object must be moved.

2.4 Systems for displacements coordination

In the field of displacements coordination of the various robots inside a formation, the main directions of study are the trajectories planning, the generation and the keeping of the formation as well as the traffic control such as they are defined in (Yu & al., 1995). (Das & al., 2002) describe a framework for the co-operative control of a robots group. Authors are interested in the applications of co-operative handling in which a semi-rigid formation is necessary to transport an object. (Tan & Xi, 2004) presents a distributed algorithm for the co-operation and the redeployment of a network of sensors embarked on mobile robots. (Yamashita & al., 2000, 2003) propose a method for movement planning of a robots team for the collective transport of an object in a 3D environment. This task raises various problems like the obstacles avoidance and the stability of the transported object.

2.5 Field of assistance supply for dependent people

If the field of assistance supply to dependent people using mobile robotics, this help is considered by a better ergonomics of the robot and an extended instrumentation of the robot (the robot can be, for example, the handicapped's wheelchair). Without being exhaustive, one can mention the works of the French multidisciplinary national group for the assistance to handicapped people (IFRATH) in which various problems on the man-machine co-operation and the co-operation between robots are studied.

In the ARPH project (Colle & al., 2002), the mobile robot equipped with an arm manipulator is intended to bring an assistance to the handicapped person. This system must help the disabled person to carry out functions of the everyday life: to grip, collect, carry and move. To achieve a task, the person cooperates with the assistance system, each one bringing its own competences and capacities. This collaboration has as main benefit the limitation of the system complexity. The system does not make "instead of" but implies the person at various degrees in the realization of the required service.

VAHM is a prototype of wheelchair mainly intended to help of the handicapped people for to control a conventional wheelchair (Bourhis & al., 2001). In this project, special efforts were made to analyse the user needs and thus to equip the wheelchair with an adapted graphical interface.

Projects proposing the implementation of a team of robots for people assistance are to our knowledge very few. Nevertheless, one will point out works that, without having for goal the assistance of handicapped people, are interested in the aspects bound to:

- the way in which a user can make a request to the system,
- a development environment of low-level behaviours in a multi-robot system,
- a system assigning tasks among a set of autonomous robots,
- several works on the interaction between a user and a multi-robot system.

We propose in the following section a design framework for a participative multi-robot system.

3. Participative multi-robots assistance system

The main objective of this work is to model a system allowing a user and more particularly a disabled person, to give a mission to a team of robots and to determine the whole process leading to its execution.

This work, in order to be realizable, has some limits and constraints:

- Experiments take place on a group of 5 compact, low cost, heterogeneous robots embarking only the minimum processing power and a fixed part managing the heavy computation and the information storage.
- The environment in which will move and operate the mobile robots is an indoor structured environment. Moreover the system has a model of the environment, i.e. a map of the places including all the obstacles and objects the robot will interact with.
- The environment is endowed with a house automation installation allowing the control of the domestic appliances (example: opening the door of the refrigerator).
- The user is part of the system; he/she may, to various degrees, act on it, accept or reject its decisions.

The user has at his/her disposal:

- the knowledge of the apartment, robots and an external view of their possibilities,
- a number of missions of displacement to be carried out like: go to, go towards, return, stop, take, put, gather, and of domestic missions like: bring closer, move away, move an object by pushing it or by pulling it, move an object by collecting it.
- a report on the state of the system.

We approach in this section various concepts which seem important for the study of the human multi-robot systems interaction. We discuss these concepts in order to define our approach and propose our participative, incremental, interactive model with an active help to the specification of a mission to a group of robots.

3.1 Usability

The concept of usability expresses the facility with which a user learns how to operate, to interpret the results of a system. It is carried out that a better adaptation of the machines to the users increases productivity, performances and reactivity.

A significant aspect of the concept of usability is the affordance. Affordance provides the user with a simple access to the actions allowed with a particular object. Controlling a mobile robot is done, for example, through an interface where one selects the zone to be reached by the robot. This possibility is, in general, coded in the interface at the time of its design. However obstacles or a breakdown of one of the robot can prevent it from reaching the indicated zone. An approach using the affordance concept would dynamically adapt according to the capacities and the state of the robot: at every moment, the user is informed of the exact list of the robot's allowed actions. Masking the impossible actions simplifies the work of the user by presenting only relevant information to him/her but can also harm the total comprehension of the state of the environment. This absence of information disturbs, according to us, the creation of a mental model of the progress of the action in the user's mind. For these reasons, a more didactical approach where all the possibilities are presented to the user will be adopted.

3.2 System appropriation

In (Rybarzcyk, 2004), the author explains that the nature of the adaptation of the human to the machine is strongly dependent on the degree of operating differences between these two entities (Piaget, 1936). Thus, if the difference is small, we are in the case of a familiar situation. Under this condition, the adaptation is carried out by a process with dominant of

assimilation[1], i.e. by generalization of the initial schemes relevant for the control of the situation or of the system. Conversely, in the event of a very different operation, the process of adaptation[2] becomes the dominating one (figure 1).
The process of accommodation leads to the transformation and the reorganization of the available action schemes which gradually produce new compositions of schemes allowing the renewed and reproducible control of new classes of situations. It comes out from these observations that the question of the difference between schemes and initial representations of the subjects and schemes and representations necessary to control the machine is crucial in the field of the ergonomic design of the system.
Within this framework, two opposite options are possible. The first one is to seek how to reduce the difference between the spontaneous operator's schemes and those appropriate to the control of the machine. This way consists in regarding the machine as a extension of the user's sensorimotor functions. When such a projection is relevant, the user tends to allot its own characteristics and properties to the system. According to this approach, we have to build machines whose appropriation will be made through a process with dominant of assimilation.

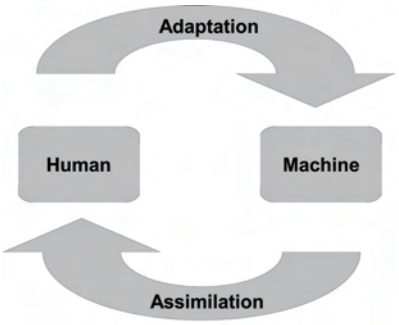


Figure 1. Application of the Piaget model to the man-system co-operation

In our case it does not appear that an identification of the user with the system is possible so much the difference is significant. Moreover, how to imagine a projection of the user in his/her individuality in the plurality of the controlled robots? This is why a second option which takes into account this difference appears much more adapted. Consequently, the design will seek to highlight this difference in order to ease its conceptualization by the subject through a mental model of the teleoperated operations.

3.3 Mental representation

A team leader carries out mental models of the course of the mission he supervises, of the expected results and of the capacities of each team member. This mental representation is used to efficiently allot the various sub-tasks of the on-going mission. In the case of a team

[1] An "action scheme" is the structured set of the generalizable properties of the action which make it possible to repeat the same action or to apply it to new contents. It is by the process of assimilation that the schemes of action will evolve.

[2] Adaptation corresponds to a modification of the organization to adapt to the external conditions. The process of adaptation is used for enriching or widening an action scheme and making it more flexible.

of robots, the operator will probably consider the robots at his disposal as agents allowing him to interact with the environment. The possibilities of each robot being limited to the actions this robot can do at every moment in its environment, we thus underline the need for maintaining a permanent correspondence between the mental representation and the real environment of the robots. However, the system must take into account low level requirements. These constraints lead to a solution differing from the user's mental representation. In this case, the system needs an interface allowing the user-system interaction for the modification of the expression the request (Jones & Synder, 2002).
We propose that the transformation of the mental representation is made in several stages according to a "top-down" process. Initially, the system will take into account the request, analyse it and, if required, propose modifications. If they are accepted, it will build a simulation of the execution and propose it to the user. This one will compare it with its mental representation. If the simulation is accepted, a second stage will transform the diagram into execution in the real environment. We are interested here only by the first stage. We propose to show in section 0 how we partly solved the problem by means of a participative approach with a multi-robot service system.

3.4 User's available information

What information the user has to know in order to have a good comprehension of what occurs during the mission? Several elements appear significant among which:

- in order to simultaneously operate several robots, a general view of the scene is used: a 2D top view (x,y) of the robots environment is adopted to simplify the handling of the interface by the user.
- a short summary of the characteristics of each robot and their current state must be available permanently.
- the user is "in the mission creation loop". Its construction is interactive thanks to a dialogue with the user in order to inform him of his errors or to suggest solutions. The system must provide a textual and graphical feedback to the user as soon as possible. This illustrates the participative aspect of the model.

3.5 Description level

The higher level of a robot control system contains mainly functions like training, reasoning and decision. Its role is to split up the user's requests into secondary tasks and to supervise their implementation. Abstract descriptions of the tasks provided by the user pass through the man-machine interface and are transmitted to a planner. This latter split up the abstract tasks in a comprehensible form by the intermediate control level. The intermediate level allows execution of the behaviours, the primitive actions and supervises the control of the robot subsystems. Low control levels contain the robot sensors and effectors controllers. On the highest level, descriptions and goals are expressed by the abstract terms of the language such as "brings me the blue cube". At the intermediate level, goals are split up into sub-goals but can still contain abstract terms. In lowest levels, descriptions are without any abstraction. The system must allow variable levels of description to allow the users to be more or less precise in their request according to their ease and to their degree of familiarisation with the system. Moreover, the user can delegate the secondary decisions to the system. Thus the choice of the robots involved in the mission can be considered as secondary. The system must also allow a degree of flexibility in the formulation of the

mission and allow omission of some intermediate stages. In our case the request is entered through a mission editor located in the highest level.

3.6 General model

Let us recall here the various points highlighted in the above sections, aspects that will guide the modelling:

- the provided service seen as a collaboration between the user and the group of robots,
- the use of a didactical version of the affordance,
- the research of a variable level description of the service request,
- the quantity and the type of information continuously returned to the user,
- the need to maintain a mental representation of the action in coherence with the mission.

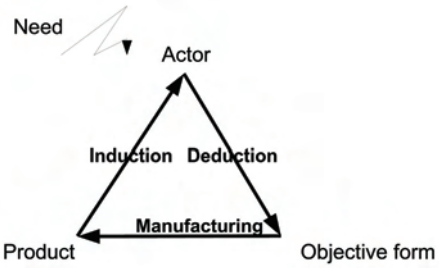


Figure 2. Modelling of the approach

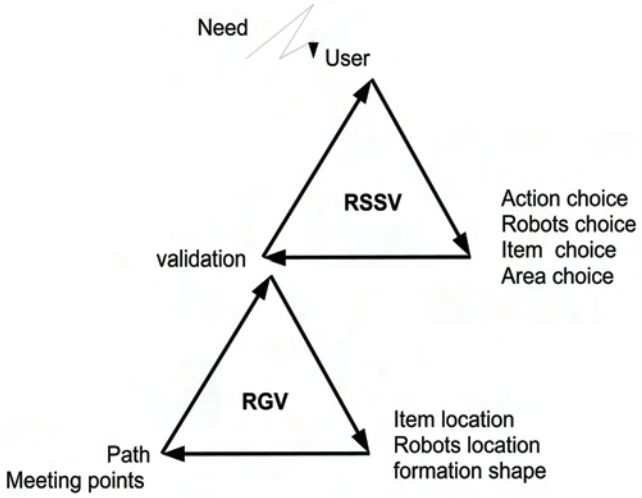


Figure 3. Processing levels

In agreement with these points, we propose in this section a model which calls upon several levels of abstraction. The participative model of this system is given in figure 2. We consider, in this diagram:

- the actor: active element submitted to the stimulus (the user and his request, for example),
- the objective form: object conceived (in the actor's mind, if this one is human, it can be described as a program if the actor is a compiler),
- the product: final state, element satisfying and corresponding to the stimulus (it can be a process whose execution will carry out the stimulus).

The deduction process creates the objective form, consequence drawn from a reasoning, which is calling upon a knowledge. The manufacturing transition transforms the objective form in its material form: the product. The return, which goes from the object to the actor, proceeds from the induction since it goes from the individual level toward a more general entity. We propose to apply recursively this general model to the mission editor.
Our approach is presented in figure 3. We separate two distinct levels:

- RSSV level, for Request Syntactic and Semantic Validation,
- RGV level, for Request Geometrical Validation.

At the RSSV level, the actor is the user; the objective form corresponds to the expression of the request. The deduction is carried out by the user following a service need. The objective form is a description of the robots missions, for example: "the robot *Its_Name* will make *Name_Action*", "the robots will go to *Place*". Manufacturing validates the choice of the robots, supplements and corrects the request by taking into account the constraints. The product, i.e. a new formulation of the mission, is presented to the user.
At the RGV level, the actor element is the RSSV, deduction extracts the relevant parameters for the geometrical checker from the formulation of the mission treated by the RSSV (the product). The manufacturing transition will set up the routes in the environment according to the formations that the team of robots must adopt for the transport of an object. The collective behaviours, such as they are defined in (Goldberg & Mataric, 1999), are directly associated to the various parts of the road to follow. These routes are then proposed to the user.
In the following sections we analyse the functionalities of the RSSV and RGV levels. We then propose solutions we implemented to create the functionalities required by the two levels of the participative model.

4. Request syntactic and semantic verificator

4.1 Object modelling

The first stage consists in modelling the various objects which will play a role in the system. Modelling emphasizes two main categories: the robots class and the objects class (figures 4 and 5).
Modelling of the robots is mainly guided by their features in terms of effectors and sensors. The modelling of the class ITEM makes it possible to describe the objects of the environment. There are two types of objects from which derive all the objects in the environment: the ITEM_ACTIVE and the ITEM_PASSIVE according to whether they are simple obstacles on which no action is possible or objects with which the robots can interact. The class AREA contains the various zones of the environment.
The number of rules evolves according to the complexity of the model of the environment. Currently, the rule base contains about sixty entries. In the following sections, we will describe the main rules classified in several groups. One will detail successively the rules that:

- check the number of arguments,

- solve conflicts between arguments,
- check the existence of ways between two rooms,
- generate the missing stages and check the state of the robots,
- check the compatibility between effector, action and item,
- check the number of robots for the transport of the object and choose the adapted formation,
- indicates the robots ready to achieve the task {*take*}.

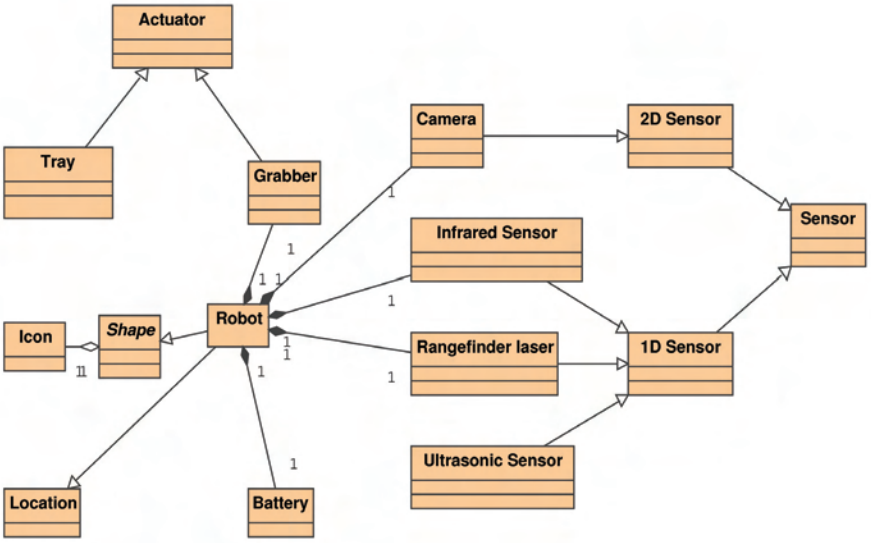
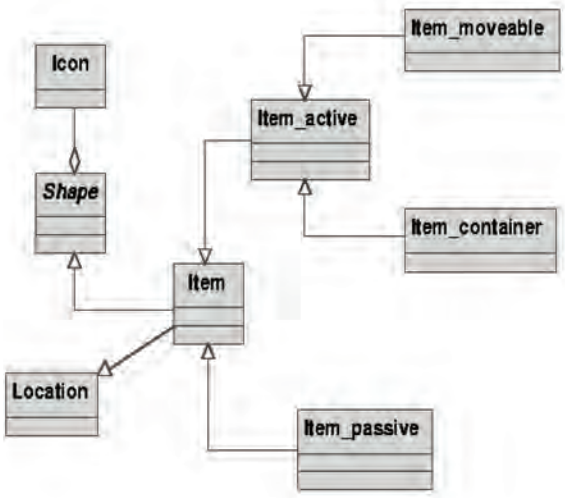


Figure 4. ROBOT class



Refrigerator

is_a_transportable = false
is_a_container = true
possible_action = open
needed_effectors = gripper
needed_sensor = camera
location = kitchen

Figure 5. ITEM class and item example

Before approaching the rules description, we briefly describe the format of the various elements specifying the mission. The user's request is represented by a succession of facts {chosen_X ?} in which X and ? can take various values:

- {chosen_action ?action} where ?action can have a value among {goto, gather, take, could, search}.
- {chosen_robot ?robot} where ?robot can have a value among {america, africa, asia, europe, oceania} (names of the five robots).
- {chosen_object ?object} where ?object can have any value of the type ITEM_ACTIVE.
- {chosen_area ?area} where ?area can have any value of the type AREA.
- the absence of a fact {chosen_X ?} systematically causes the insertion of the fact {missing_X} (X represents one of the parameters: robot, object or area).
- in addition, to each {chosen_X ?} describing a user's choice, corresponds {computed_X ?}, value of the parameter X calculated by the inference engine which will contain the arguments corrected or deduced by the system.

4.2 Request syntactic checking: Arguments number checking and arguments conflicts

When the various parameters are entered by the user, the inference engine starts by checking the number of received arguments. Each time a fact of type {chosen_X} is missing, the system inserts the fact {missing_X}. The following paragraphs describe the reaction of the system. Some examples of actions are given.

Case of action *goto*

- {missing_robot}: the system can choose by itself the robot which will carry out displacement. The choice of the robot will be done according to the mode in progress (parallel or sequential, section 0),
- {missing_area}: the presence of this fact blocks execution,
- {missing_object}: the presence of this fact is ignored,
- {chosen_object ?object}: the position of the object is extracted, starting from the data base, and inserted as a {computed_area}. In the event of conflict with a {chosen_area}, the system requires the user to solve the ambiguity on the choice of the robots destination.

Case of action *take*

- {missing_robot}: the system chooses one or more robots compatible with the action take on the object,
- {missing_area}: the presence of this fact is ignored,
- {missing_object}: the position of the object is extracted, starting from the data base, and inserted as a {computed_area}. In the event of conflict with a {chosen_area}, the system requires the user to solve ambiguity on the choice of the destination of the robots.

Case of action *put*

- {missing_robot}: if {chosen_object} exists, the robots which transport the object are used to create {computed_robot}. If not, the system requires the user to specify the request,
- {missing_area}: the current position of the robot is used to add {computed_area},
- {missing_object}: if {chosen_robot} exists, the object transported by the robot(s) is used to create {computed_object}. If not, the user is asked to be more precise in its request.

Table 1 recapitulates the links between the selected action and the type of required arguments. Any compulsory missing argument produces an error which results in a

message inviting the user to be more precise and to provide the missing argument. Certain actions as action *put* (table 2) are a little bit particular and use a group of specific rules: the user can either specify the robot that transports or the transported object.

Action	Goto	Gather	Take	Search
Optional arguments	robots_list	robots_list area	robot	robots_list area
Compulsory arguments	area		Transportable object	Object

Table 1. Actions/arguments table

Put	Optional arguments	Compulsory arguments
case 1	robot	carried object
case 2	carried object	Robot

Table 2. *Particular case: action put*

Arguments conflicts can occur in particular for action *put* where the robot and object-choice do not correspond (the robot does not transport the object indicated by the user). Some of these conflicts are solved in an automatic way: the conflict of argument is solved by the creation of an action *take* for the object indicated before carrying out the action *put*.

4.3 Environment topology checking

One of the roles of the RSSV is to check the accessibility of the activity area by the selected robots. The initial base of facts contains the list of the paths between two close parts in the form {path area1 area2}, {path area2 area3} (Paths such as {path area1 area3} are automatically deduced by chaining rules). It is the only representation of the environment available to CLIPS[3] (figure 6). CLIPS thus checks the existence of a path between the current position of each robot and the position to be reached. If the current position of the robot does allow reaching the final position, the system chooses another robot.

[3] CLIPS is an expert system development environment. See http://www.ghg.net/clips/CLIPS.html for details

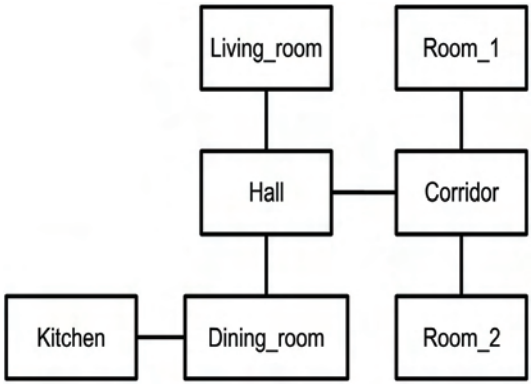


Figure 6. Topology of the environment known by the RSSV

4.4 Checking robots status and action/object/robot compatibility

The validation of the request checks the availability of the robots chosen by the user. Two variables manage the status of the robots, one indicates if the robot operates correctly, the other indicates if it is in use. The system will preferably choose robots which are not in use in order to be able to start the task immediately. Section 0 discusses some details regarding the possibilities of automatic robot allocation offered by the RSSV.

The modelling of the sensors and the effector includes the list of the achievable actions. Thus a position encoder is necessary to the action *goto*, an effector of the arm type is necessary to the action *take*. Moreover, the description of each object contains the list of its possible actions. The RSSV checks if the selected robots have the features necessary to achieve the required action. In the opposite case, a warning is sent to the user and the RSSV switches to the robot automatic allocation mode (section 0). In addition, for actions not possible with certain types of objects (for example, the action *take* must be mandatory carried out on object of TRANSPORTABLE type.) the system signals the inconsistency of the request to the user so that he may correct it.

4.5 Robots number checking and formation choice

The services offered by the group of robot are among others, the transport of objects and, in particular, collective transport of bulky objects. The number of robots necessary to act on an object is fixed during the modelling phase (section 0) in the description of the object. Thus there must be as many facts {chosen_robot} than the number of robots necessary to achieve the task. Specific rules manage the selection of the groups of robots. The following example illustrates these two RSSV functionalities: initially, the user makes the request {*take* chair America} the system then adds an additional robot (Asia for example) for this task because the action *take* chair requires two robots. In a second step, the user makes the request {*goto* America living_room}. To satisfy this request the system will create two actions *goto* with the robots America and Asia. The modelling of the object contains also the formation to be adopted during its transport. The geometry of the formation is not used by the RSSV and will be transmitted to the RGV to check the possibilities of passage of the formation.

4.6 Robots automatic choice

When the user does not specify a robot for a given task, or when a robot is missing in order to achieve a task, or when the selected robot cannot achieve the task (robot out of service, ...), the RSSV can select robots in an automatic way. To each robot is associated a coefficient called *usage_score*. This concept was introduced in order to be able to classify the robots according to a criterion of execution cost. The calculation of the *usage_score* takes into account the physical features of the robots in terms of embarked sensors and effectors. The value of each robot's feature was adjusted following several tests. The *usage_score* takes also into account the current state of the robot and will be higher if the robot is in use. When the user does not specify the robots that must take part in the action, the system chooses the robot that has the necessary functionalities and the lowest *usage_score*. Two operating modes can be used by the inference mechanism: the parallel and the sequential modes.

In the parallel mode, the system endeavours to parallelize the tasks carried out by the robots by choosing a new robot for each new task (with respect to the lowest *usage_score*) criterion. In the sequential mode, the system will help the user to build the mission step by step. In opposition to the parallel mode, the system tries to assign the maximum of tasks to the same robot in order to make it possible to the user to build a complex mission. In the current state of the project, the system cannot change the affected robot for the preceding tasks. The case occurs when the user inserts at stage N an action which cannot be achieved by the robot selected automatically for the N-1 first tasks. An evolution of the system will allow each task to have an influence on the choice of the robot of the preceding tasks. Lastly, it is planned that the system can switch from one mode to another during the edition of a mission, in order to benefit from both edition modes according to the context.

4.7 Example of operation

This section illustrates a realistic scenario in which the user wants a soda can from refrigerator. The request of the user thus takes the simple form: {*take* can}.

The figure 7 illustrates the operation completed by the RSSV. The following verifications are made:

- "can" item is known by the system,
- "can" item is compatible with the action *take*,
- "can" item is located the container "refrigerator",
- type and numbers of robots necessary for action *take* on item "can",
- availability of the robots,
- type and numbers of robots necessary for action *take* on item "can" from item "refrigerator",
- the container "refrigerator" is reachable.

Once all these constraints checked, a robot (America), compatible and free, is chosen, an intermediate action {*goto* America kitchen} is created and the result is displayed to the user. The following section presents the second level of the system.

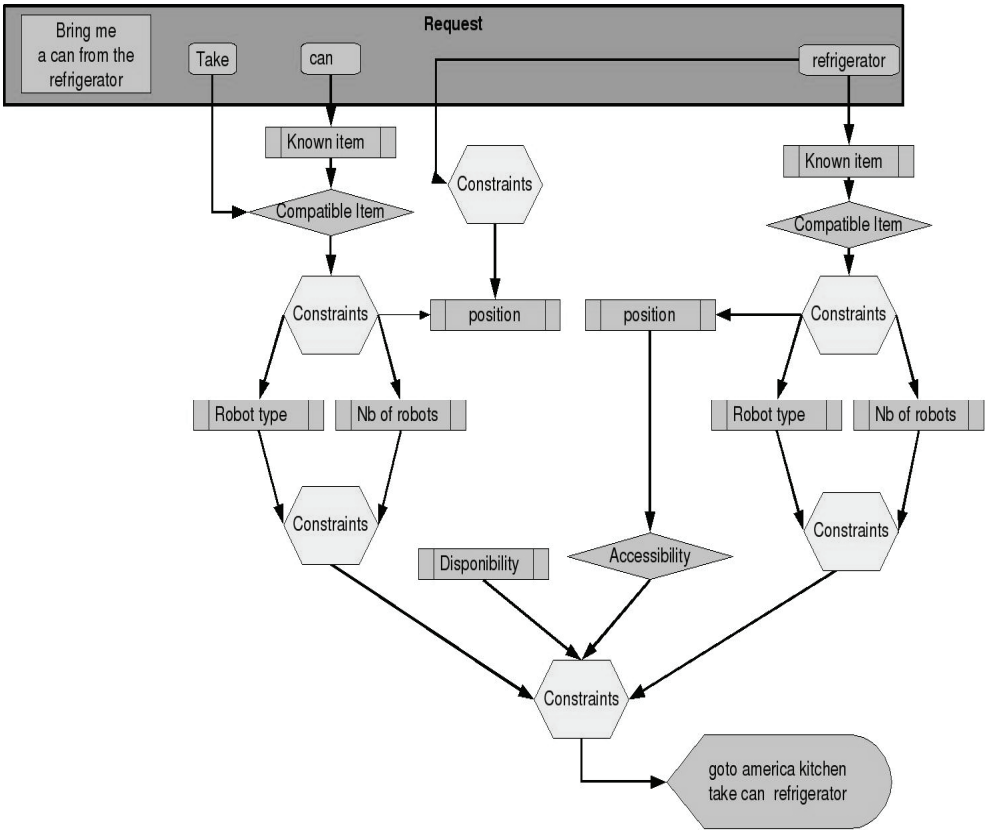


Figure 7. *Example of request*

5. Request Geometrical Validation

This section presents the Request Geometrical Validation (RGV): the constraints, the simplifying assumptions, the stages allowing the validation of the robots formation between the selected points.

First let's recall the functionalities the RGV module must provide:

- give an outline of the robots trajectories to the user,
- find the meeting points for group navigation,
- validate or reject the passages for the formations according to the meeting points.

In the discrete representation of the environment that was chosen, the environment is divided into cells of fixed size. A cell can be either in an obstacle or in a open space. The cells table representing the environment is dynamically created on each program start from information stored in a data base. A map representing the accessible points of the environment is illustrated in figure 8.

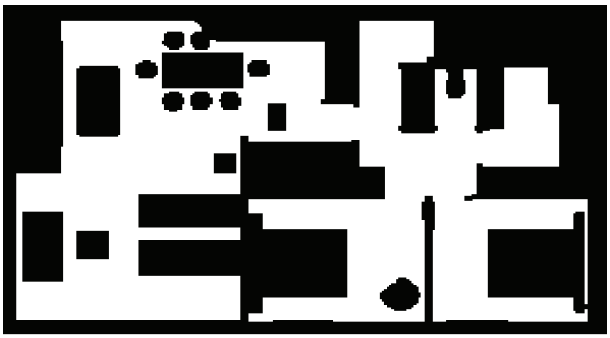


Figure 8. Environment occupation map

Figure 9 shows the various steps which lead to the geometrical checking of the robots formation path. Each step will be detailed in the following paragraphs.

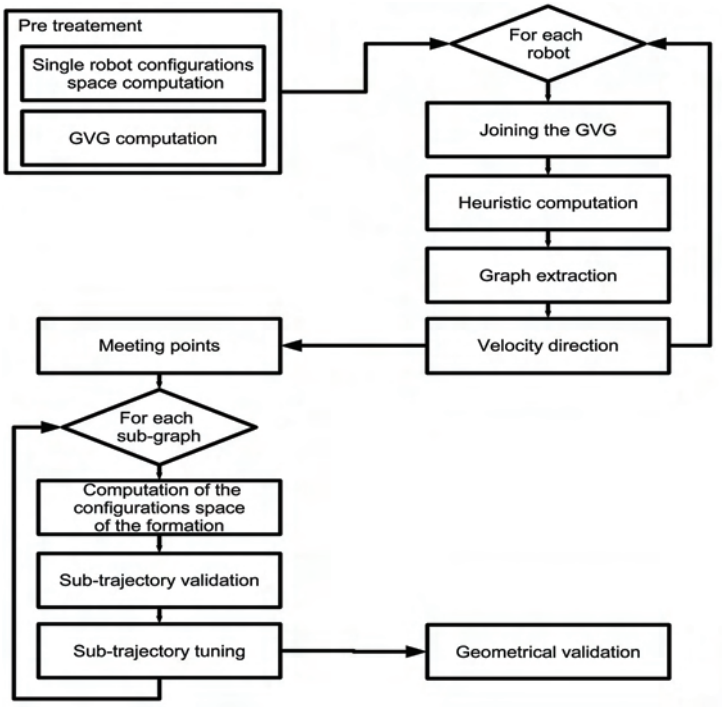


Figure 9. Processing steps leading to the routes geometrical checking

In the following sections, we will denote by:

- C the configurations space,
- C_{open} the set of valid configurations in C,
- $C_{obstacles}$ the set of invalid configurations in C,
- q_i a configuration in C,
- $R(q)$ the set of points occupied by the robot R in configuration q.

5.1 Environment preprocessing: Calculation of the robot configurations space and the GVG

A first off-line environment processing is carried out. It includes two stages: calculation of the configurations space of a robot and calculation of the generalized Voronoï graph in this space.

The robots are cylindrical, they have 3 parameters x,y and θ (x, y: position, θ: orientation). Thus $R(q)$ does not depend on θ (the points occupied by the robot do not depend on the parameter θ). The configurations space has only two dimensions isomorphous to the Euclidean plan. The configurations space is then approximated by a dilation algorithm using the d_8 distance. This distance approximation introduces a safety margin around the obstacles.

To connect the initial and final configurations in C_{open}, we use a generalized Voronoï graph (GVG) calculated on C_{open}. The GVG represents the set of the points of C_{open} equidistant of $C_{obstacles}$.

5.2 Individual trajectories processing

Each robot is individually considered to obtain a route in C_{open} between the final and starting configurations.

As the starting and final positions of the robots seldomly match a point on the GVG, Bezier splines join these positions to the nearest GVG point.

From the graph created in section 0, a directed sub-graph corresponding to the valid configurations between the starting and final positions is extracted.

To speed up the graph exploration, the distance between each node of the graph and the target point is evaluated. This evaluation allows the exploration function to choose the next node to be visited while approaching the target. A wavefront algorithm stamps each point of the graph with the distance separating it from the target. This distance takes account of the obstacles and avoids all local minima (figure 10).

Figure 10. Example of the wave front propagation in the environment

From the starting configuration q_d, the GVG nodes are visited following the potentials given by the wave front algorithm. Finally, the valid configurations set between q_d and q_f is obtained for each robot. This valid configurations set q_i is stored in a vector noted $T_{i \in R,j.}$

Each configuration contains only 2 parameters corresponding to x and y. The robot orientation, in each configuration q_i, is added to vector T. All the orientations θ being

allowed, the direction of the local tangent is calculated for each position on the trajectory which provides the missing parameter (figure 11).

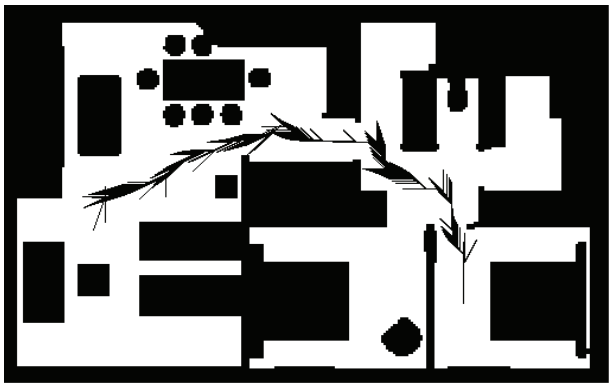


Figure 11. Representation of the speed vector

These stages lead to the construction of a vector set $t_{i\ e\ R,j}$ representing the series of the free configurations between the starting and final points of each robot involved in the mission. The following stages lead to the creation of robots groups for navigation in formation.

5.3 Superposition of individual trajectories}

By superposition of the various individual routes (figures 12 to 14), the meeting points as well as the trajectories segments along which the robots will navigate in formation are extracted.

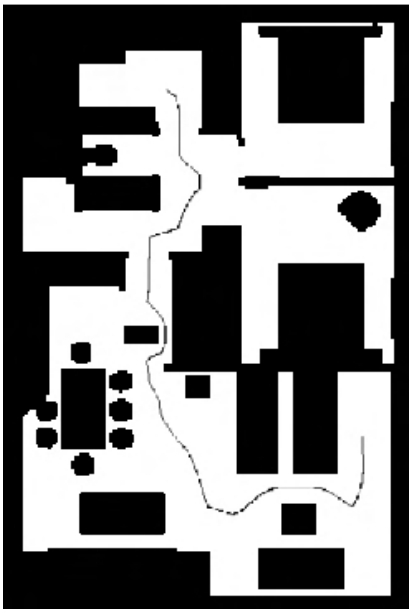


Figure 12. Robot 1 trajectory

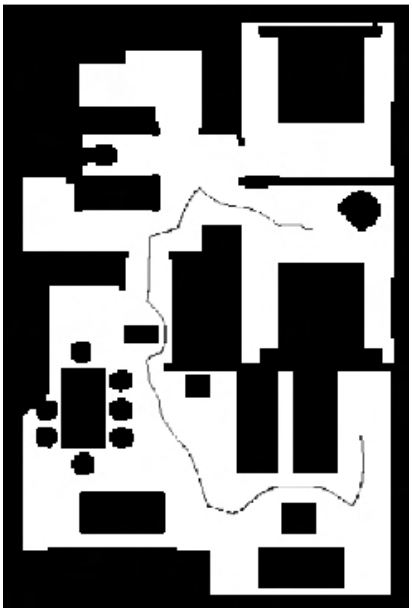


Figure 13. Robot 2 trajectory

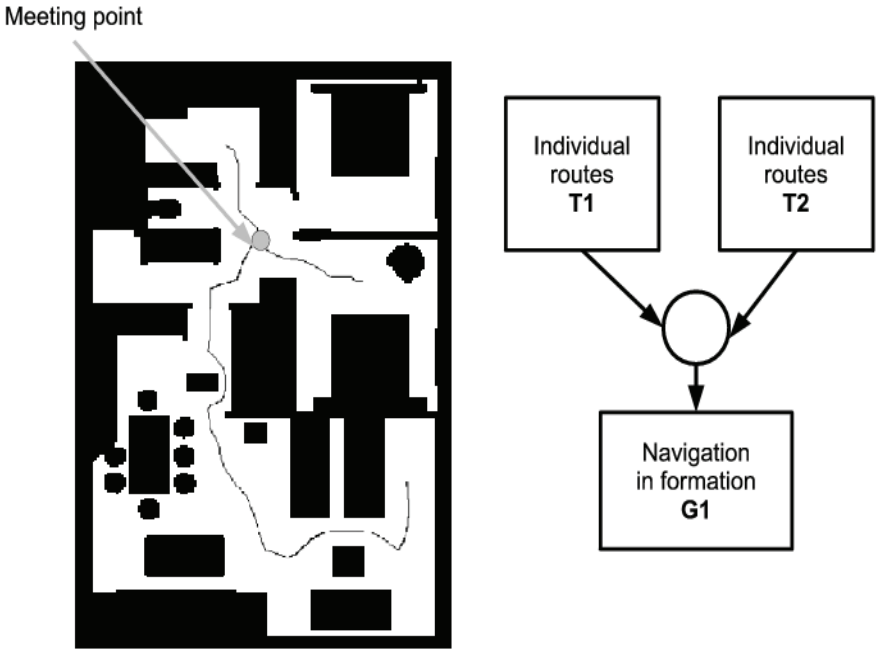


Figure 14. Trajectory of the two robots and navigation graph

Free configurations being taken on the GVG of the configurations space, the vectors of two robots having the same destination contain at least a common configuration. As soon as a common configuration is detected, a convoy formed by the two robots is created starting from this point. The process continues and other robots are added into the convoy. The vector containing the configurations of the formation resulting from the meeting point *i* is called G_i.

5.4 Adaptation of the sub-trajectories to the formations

The preceding stage created a set of G_i vectors containing the configurations of the robots between two meeting points. The configuration q ($q \in G_i$), initially defined as the configuration of a single robot, is now considered as the configuration of the robots formation whose reference point is its barycentre.

Because of this new status, the G_i vectors can contain invalid formation configurations as they were calculated in C_{open} of a single robot and not in C_{open} of the whole formation. The G_i vector must be modified in order to meet the formation constraint (each configuration must be in C_{open} of the formation) while maintaining the configurations connexity.

5.5 Computing the formation configurations space

Assume R_i is the set of the robots belonging to the formation, $R(q)$ is the set of points occupied by R_i while being in the configuration q and O the set of the points of the transported object. H is defined as $F = O \bigcup_i R_i$. One notes $\hat{H}$ the convex hull containing H.

C_{open} of $\hat{H}$ is included in the configuration space of the formation carrying the object, thus any configuration in $C_{open}/\hat{H}$ is also in C_{open}/H .

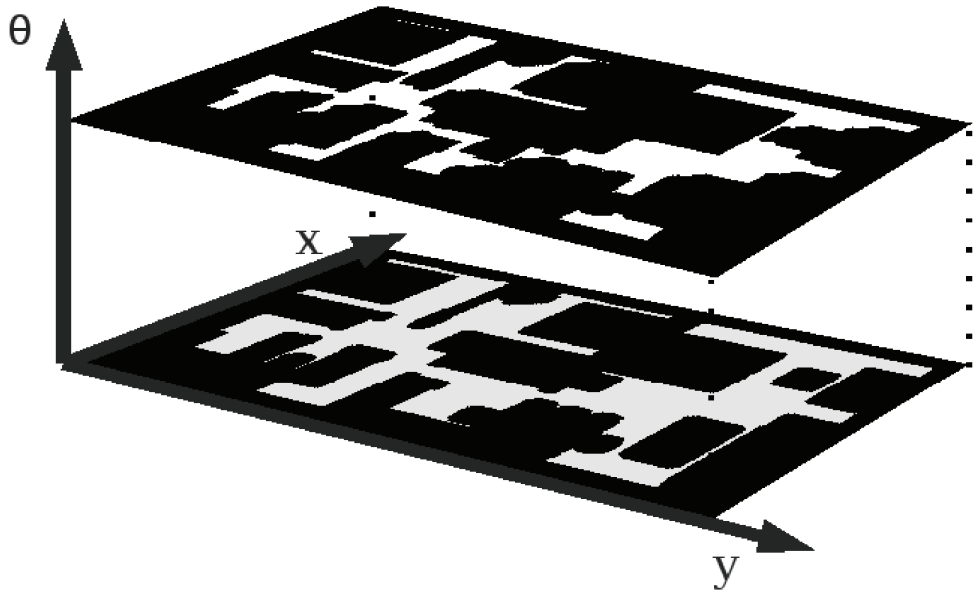


Figure 15. Configuration space of a robots formation. The configurations space is shown for two orientations of the robot formation

Two configurations q_1 and q_2 connected in C_{open}/R could not be connected by a free path in $C_{open}/\hat{H}$. To check the existence of a path between two consecutive meeting points in $C_{open}/\hat{H}$ a propagation algorithm (described in section 0), adapted to a 3 dimensional configurations space, is used. The distances are propagated from the final configuration q_2 until reaching the initial configuration q_1. If the propagation succeeds, q_1 and q_1 are connected in $C_{open}/\hat{H}$ (figure 15). If the propagation fails, the robots formation will not be able to move from q_1 to q_2.

Once the existence of a free path between the two ends of the group trajectory is established in the configurations space of the formation, it is still necessary to check the absence of collisions between the trajectory curve and the $C_{obstacles}$. If a collision occurs, on each collision zone, the trajectory is locally tuned in order to satisfy to the constraints. The tuning consists in a deformation of the original trajectory by selecting a clear path between the point preceding the clash with the obstacle and the point following that same clash. This clear path is also computed thanks to a local front wave algorithm (figure 16). This method ensures the least deformation between the original trajectory and the new one. This deformation according to the three dimensions of the C_{space} corresponds to translations and rotations of the formation in the Euclidean space.

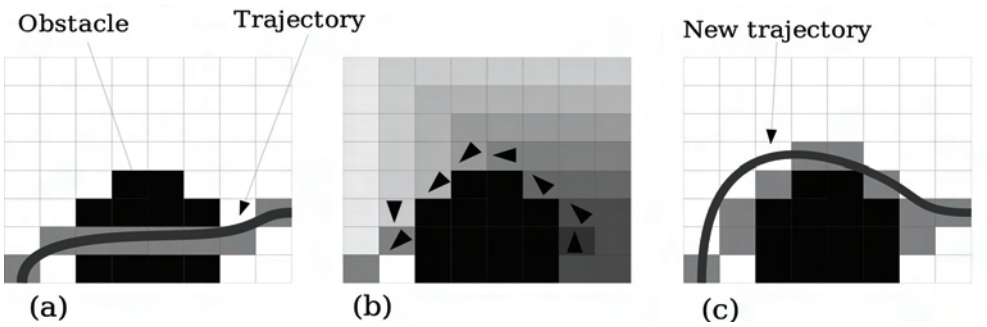


Figure 16. Avoidance of a $C_{obstacle}$

The operation is reiterated on each couple $(q_e, q_s)_i$ belonging to the vector G_i to create the vector $\hat{g}_i$ valid on C_{open}/F. Then, for each vector G_i, the vector $\hat{g}_i$ is calculated.

5.6 Final group trajectory and user interface

Lastly, a series of vectors $\hat{G}_i$ corresponding to the various configurations of the formations adopted by robots after every meeting point is displayed on the user interface (figure 17).
This interface is divided into several zones:

- the robots selection zone: each button displays the name and the state of the robot,
- the objects selection zone: the objects retrieved from a database are accessible via a drop-down menu.
- the map zone: represents the environment in which the robots evolve.
- the actions zone: a simple set of buttons attached to the possible actions.
- the information zone: general purpose message zone.
- the RGV zone: RGV control panel.
- the debug zone: access to useful functionalities during program development.

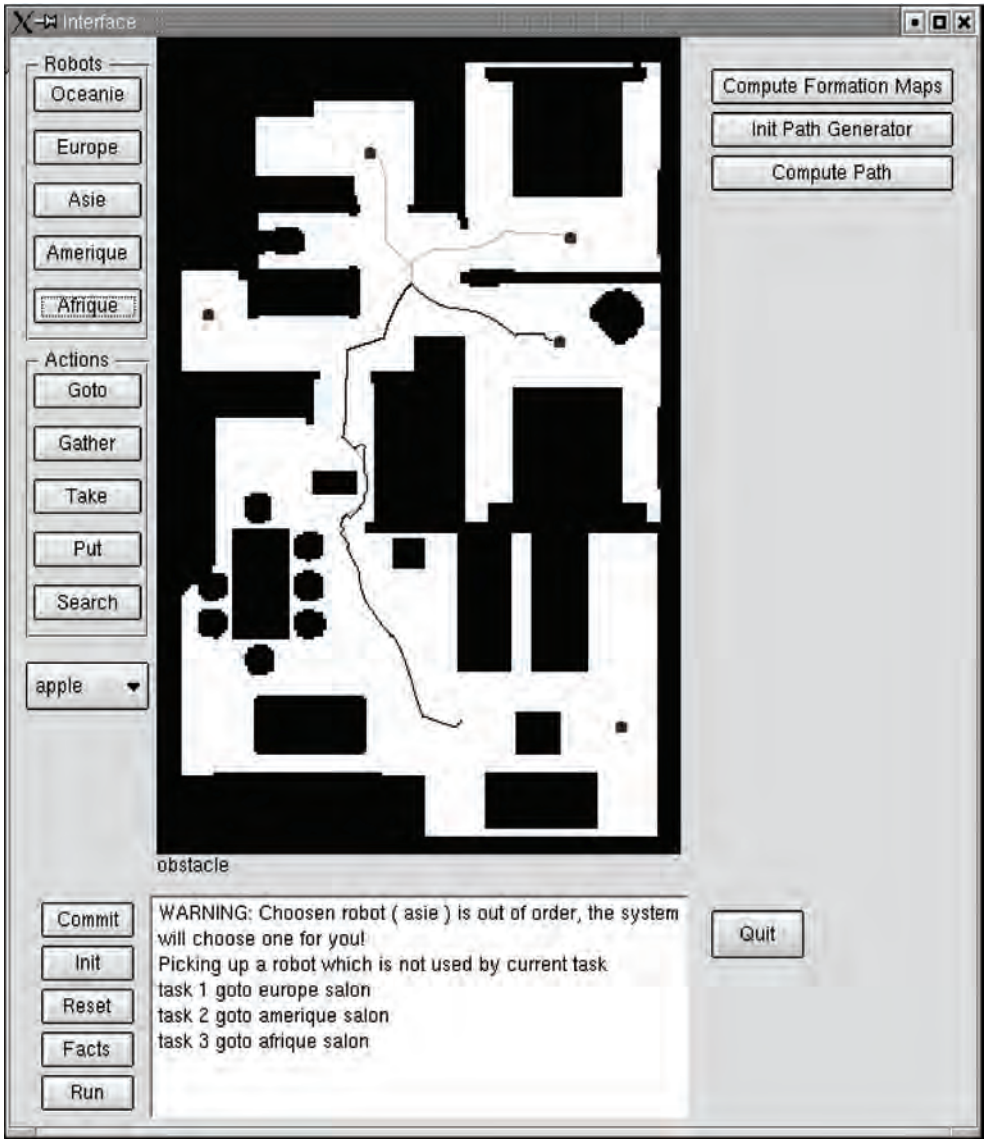


Figure 17. User's interface screenshot

The passages of the formations as well as the transported object must then be validated by the user before execution.

6. Conclusion and prospects

This work aims at modelling the interaction between a user and a multi-robot system in order to give a mission to a team of robots and to determine the set of the driving process leading to its execution.
According to some hypotheses among which:

- the indoor structured environment is known,
- the small number of robots,
- the cost which influences the system architecture,
- the user may act on the system decisions,

the user can request some missions the system is able to accomplish. We mainly insist on the system participative aspect in order to be accepted by the user in spite of its limitations. This participative aspect implies:

- a human-system interface taking care of the communication in a friendly, click and drag, icon based, high-level language,
- the expression of the request proposed as a set of remote services carried out by a reduced number of robots,
- the ability of act during the construction of the solution to the request,
- a way to act directly on the robots and on the mission scene in the event of modification, breakdown or execution failure,
- an information on the system state always available.

The originality of this work stands in our approach of the interaction between the user and the multi-robot system. The required services are seen as a man/machine co-operation where the user delegates some decisions to the system. We qualify our model of participative, incremental, interactive with an active help to the specification of the mission. Participative because the user is involved in the mission creation process. Incremental, because the request is analysed through two sequential levels. Interactive, because a constant dialogue between the user and the system is maintained throughout the construction of the request. We propose solutions for the realization of the functionalities of the two levels of our model. The realization of the first level uses of an inference engine and a rule base. The second level, the geometrical verificator uses trajectory planning methods. The two levels are assembled in a network architecture which includes a database and a graphical user interface to complete the multi-robot mission editor. Experiments allow us to validate the usability of this system. Tests on real robots are ongoing.
Several evolutions prospects and improvements of the system are possible. In one hand, the number of objects and actions must be extended to offer more services to the user. In addition, it is necessary to modify the human-system interface to take into account more complex missions and to allow interactivity during the mission execution. An improvement of the general ergonomics of the interface must also be considered after an evaluation of the system will be made by a handicapped person. Finally the lower levels of the robots control must be set up. Once these levels set up a second phase of study of the editor will be able to take place. In this phase the readjustments of the contents of the database according to the information collected by the robots will be done.

7. References

Ani:02: Hallam, B. & Floreano, D., Hallam, J. & Hayes, G. & Meyer, J.A. (2002). From Animals to Animats 7, *Proceedings of the Seventh International Conference on Simulation of Adaptive Behavior*.

Ani:04: Hallam, J. & Meyer, J.A. & Schaal, S. & Ijspeert, A.J. & Billard, A. & Vijayakumar, S. (eds) (2004). From Animals to Animats 8, *Proceedings of the Eighth International Conference on Simulation of Adaptive Behavior*.

Arai, T. & Ogata, H. & Susuki, T. (1989). Collision avoidance among multiple robots using virtual inpedance. *In Proceedings of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*. pp. 479–485.

Arai, T. & Pagello, E. & Parker, L. (2002). Editorial: Advances in multi-robot systems. *IEEE Trans. Robot. Autom*. 18(5), pp. 655–6617.

Asama, H. & Matsumoto, A. & Ishida, Y. (1989). Design of an autonomous and distributed robot system. *In Proceedings of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp. 283–290.

Balch, T. & Parker, L.: (2002). Robot teams: From polymorphism to diversity. Balch, T. & Parker, L. (eds.)

Beni, G. & Hackwood, S. (1992). Stationary waves in cyclic swarms. *In IEEE Int. Symp. on Intelligent Control*, pp. 234–242.

Bonabeau, E. & Theraulaz, G. (2000). *L'intelligence en essaim*.

Bourhis, G. & Horn, O. & Habert, O. & Pruski, A. (2001). An autonomous vehicle for people with motor disabilities. *IEEE Robot. Autom. Mag*. 7(1), pp. 20–28.

Brooks, R. (1986). A robust layered control system for a mobile robot. *J. Robot. Autom*. 2(1), pp. 14–237.

Colle, E. & Rybarczyk, Y. & Hoppenot, P. (2002). ARPH: An assistant robot for disabled people. *In IEEE International Conference on Systems, Man and Cybernetics.*

Das, A. & Fierro, R. & Kumar, V. & Ostrowski, J. & Spletzer, J. & Taylor, C. (2002). A vision-based formation control framework. *IEEE Trans. Robot. Autom*. 18(5), pp. 813–825.

Drogoul, A. & Ferber, J. (1993). From tom-thumb to the dockers: Some experiments with foraging robots. *In From Animals to Animats 2, Proceedings of the Second International Conference on Simulation of Adaptive Behavior*, pp. 451–459.

Fong, T. & Thorpe, C. & Baur, C. (2001). Collaboration, dialogue, and human–robot interaction. *In Proceedings of the International Symposium on Robotics Research*, Lorne, Victoria, Australia.

Fukuda, T. & Nakagawa, S. (1987). A dynamically reconfigurable robotic system (concept of a system and optimal configurations. *In Proceedings of IECON'87*, pp. 588–595.

Gerkey, B. & Mataric, M. (2003). A framework for studying multi-robot task allocation. I*n Proceedings of the Second International Naval Research Laboratory Workshop on Multi-Robot Systems*. Washington, District of Columbia, pp. 3367–3373.

Goldberg, D. & Mataric, M. (1999). Coordinating mobile robot group behavior using a model of interaction dynamics. *In Autonomous Agents*. Seattle, Washington, pp. 100–107.

Iocchi, L. & Nardi, D. & Salerno, M. (2001). Reactivity and deliberation: A survey on multi-robot systems.

Jones, H. & Rock, S. (2002). Dialogue-based human–robot interaction for space construction teams. *In Proceedings of the 2002 IEEE Aerospace Conference*, Big Sky, Montana.

Jones, H. & Synder, M. (2002). Supervisory control of multiple robots based on a real-time strategy game interaction paradigm. I*n ACM Computer Supported Cooperative Work.*

Lemay, M. (2003) Systèmes flous pour le contrôle d'une formation de robots.

Mataric, M. & Sukhatme, G. &Ostergaard, E. (2002). Sold!: Auction methods for multi-robot coordination. *IEEE Trans. Robot. Autom. special issue on Advances in Multi-Robot Systems*. 5(18), pp. 758–786 .

Parker, L. (1998*).* ALLIANCE: An architecture for fault-tolerant multi-robot cooperation. *IEEE Trans. Robot. Autom*. 14(2), pp. 220–240.

Parker, L. (2000). Current state of the art in distributed autonomous mobile robotics. *Distributed Autonomous Robotics Systems.* 18(4), pp. 3–12.

Parker, L. & Bekey, G. & Barhen, J. (2002). Distributed Autonomous Roboti systems, vol. 4. Springer, Berlin Heidelberg, New York.

Piaget, J. (1936). La Naissance de l'Intelligence chez l'Enfant.

Premvuti, S. & Yuta, S. (1989). Considerations on the cooperation of multiple autonomous mobile robots. *In Proceedings of the IEEE Int. Workshop on Intelligent Robots and Systems*, pp. 59–63.

Rybarczyk, Y. (2004). Etude de l'appropriation d'un système de tél´eopération dans l'optique d'une coopération homme-machine. *PhD thesis*, Université d'Evry-Val d'Essonne, Evry, France.

Rybski, P. & Stoeter, S. & Gini, M. & Hougen, D. & Papanikolopoulos, N. (2002). Performance of a distributed robotic system using shared communication channels. *IEEE Trans. Robot. Autom*. 18(5), pp. 211–225.

Schultz, A. & Parker, L. (2002). Multi-robot Systems: From Swarms to Intelligent Automata. Kluwer.

Sellem, P. & Dalgalarrondo, A. (1999). Extension d'une architecture de contrôle derobot mobile à unsystème distribué de robots.

Tan, J. & Xi, N. (2004). Peer-to-peer model for the area coverage and cooperative control of mobile sensor networks. *In Proceedings of IEEE ICRA'04 Workshop on Environmental Robotics*.

Tews, A. & Wyeth, G. (2002). MAPS: A system for multi-agents coordination. *Adv. Robot (VSP/RSJ)*. 14(1), pp. 37–50.

Wang, P. (1989). Navigation strategies for multiple autonomous mobile robots. *In Proceedings of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp. 486–493.

Yamashita, A. & Fukushi, M. & Ota, J. & Arai, T. & Asama, H. (2000). Motion planning for cooperative transportation of a large object by multiple mobile robots in a 3D environment. *In Proceedings of 2000 IEEE Int. Conf. on Robotics and Automation*, pp. 3144–3151.

Yamashita, A. & Ota, J. & Arai, T. & Asama, H. (2003). Motion planning of multiple mobile robots for cooperative manipulation and transportation. *IEEE Trans. Robot. Autom*. 19(2), pp. 223–237.

Yu, U. & Fukunaga, A. & Kahng, A. (1995). Cooperative mobile robotics: Antecedents and directions. *Technical Report* 950049.

11

Randomized Robot Trophallaxis

Trung Dung Ngo and Henrik Schiøler (equal)
Aalborg University
Denmark

1. Introduction

Energy is the critical resource of most living mechanisms. Recent research in robotics has been mostly considered in behavioural autonomy rather than in energy autonomy. This chapter presents our study in "randomized robot trophallaxis". The chapter consists of three main parts: modeling, simulation, and implementation.

In the first section, we model energy trophallaxis in multi-robot system through probabilistic modelling. Deterministic modelling of large groups of interacting mobile robots leads to highly complex nonlinear hybrid models, likely to be highly sensitive to pre-conditions, i.e. chaotic. Thus, any imprecision in pre-conditions would turn results from such a model useless even for moderate time horizons. However, chaotic systems often exhibit smooth ergodic properties, i.e. time averages have limit values independent of initial conditions and only smoothly dependent on model parameters, etc. Randomness and ergodic properties may exist naturally in such systems or even be intentionally enforced by introducing inherent uncertainty/randomization into the behaviour of individual robots in order to prevent non productive cyclic behaviour such as deadlock or livelocks. Ergodic properties and randomness calls for probabilistic modelling. We propose a combined probabilistic model covering energy exchange between robots, energy consumption in individual robots, charging at predefined charging stations and finally random mobility, where the latter comes in the shape of highly versatile Markovian mobility model. Stationary results furnish overall system performability analysis, such as the impact of individual behaviour on overall system survivability. The section presents the proposed model and comprises central parts of model development as well as illustrative numerical results.

In the second section, we simulate aspects of energy autonomy inspired by natural phenomena of animal behaviour. Trophallaxis is a natural phenomenon, biologically observed from social insects or vertebrate animals, to exchange food between colony members. This section describes the concept, "Randomized Robot Trophallaxis", based on a group of autonomous mobile robots with capabilities of self-refueling energy and self-sharing energy. We firstly clarify the concept "Randomized Robot Trophallaxis" by given examples of natural animal societies. Secondly, we examine the concept by simulation results in order to point out considerable advantages of trophallactic features when deploying multiple mobile robots. The section is concluded with discussion of "randomization" and its appearances in multi-robot system.

In the third section, we mainly present hardware implementation of our mobile robots capable of performing not only self-refueling energy but also self-sharing energy. We describe the mechanical and electrical design of a mobile robot, called the CISSbot[1]. The robots are designed towards truly autonomous robots in large populations through energy trophallaxis. Unlike present mobile robots, the CISSbots are energetically autonomous robots because they are able to not only autonomously refuel energy by picking batteries up at a charging station, but also share energy by exchanging batteries to other robots. The CISSbots basically consist of their own processing power, sensors, and actuators. However, to achieve the capability of battery exchange, the CISSbots need a special design of battery exchange mechanism. In this section, we present the realization of the design, both the mechanics and the electronics of the CISSbot. Details on battery exchange technique and power management are clarified. Finally, the section issues an outline of our future work on the CISSbots.

2. A Probabilistic Model of Randomized Robot Trophallaxis

2.1 An Introduction to Probabilistic Modelling

Various mathematical modelling paradigms exist for dynamical systems, which all aim to provide system predictability, i.e. answer questions regarding future state of the system evolving from some initial state or set of initial states. The appropriate model paradigm depends highly on the nature of the questions to be answered, i.e. the scope of required information as well as the form of the answer provided by the model. In the present case, we ask for distribution of energy resources throughout the population of mobile robots as well as the survival state of the population, i.e. how many robots have survived energy starvation over a certain time frame. Of particular interest is the impact, that individual robot behaviour may have on energy distribution and survival.

Any such model should include all aspects of robot behaviour suspected to impact population state. Here we suggest: mobility, energy sharing policy and recharging as well as energy consumption.

Deterministic models appear as differential equations, as discrete state transition systems or combinations of the two former when hybrid modelling is applied. Common to deterministic models is their ability to provide exact answers to exactly formulated questions. That is, when all pre-conditions are exactly stated the future may be exactly predicted. When pre-conditions are only partly known, non-deterministic modelling in the shape of differential inclusions or non-deterministic state transition systems, may be applied. The precision of non-deterministic models follows the precision by which pre-conditions are given.

From a deterministic modelling perspective, large groups of interacting mobile robots correspond to a highly complex nonlinear hybrid model, which is likely to be highly sensitive to pre-conditions, i.e. chaotic. Thus, any imprecision in pre-conditions would turn results from such a model useless even for moderate time horizons. On the other hand chaotic systems like large interacting robot populations are likely to possess so called *mixing* properties. Mixing implies, that from partly known pre-conditions, almost any future development is possible even within moderate time horizons. Thus precision of answers

[1] CISSbot is abbreviated from our center in Danish: Center for Indlejrede Software Systemer (CISS). http:// www.ciss.dk

from non-deterministic models of mixing systems rapidly deteriorates with time. Chaotic systems with mixing properties often exhibit smooth *ergodic* properties, i.e. time averages have limit value independent of initial conditions and only smoothly dependent on model parameters. In other words such systems exhibit predictable statistics. Mixing or ergodic properties may be enforced onto the system by introducing uncertainty/randomization into the behaviour of individual robots. This may prevent the overall system from getting stuck in non productive cyclic behaviour such as deadlock or live locks. As an example randomization is used for wireless access protocols and suggested for the so called *leader election* problem.
Based on the previous arguments we suggest a probabilistic model for trophallaxis among mobile robots including mobility, energy sharing policy and recharging as well as energy consumption. Such a model may serve as the basis for long term stochastic simulation, in which case ergodic properties become critical w.r.t. the meaningfulness of the obtained results. On the other hand, the model may be mathematically tractable, allowing for direct numerical assessment of the coupling between individual behavioural parameters and overall system state. Such a model is presented below.

2.2 Modelling

The developed model aims to combine the effects of all the influential mechanisms associated to trophallaxis in mobile robot populations: mobility, resource sharing, charging and resource consumption. Initially separate models are developed for each of the above effects, which are then assumed additive and conditionally independent given instant system state. System state is defined to be the position $x_i(t)$ of every robot i, its velocity $v_i(t)$ as well as its energy resource $b_i(t)$.
Since we aim for a probabilistic model, exact values of x_i, v_i and b_i are not tracked. Instead the developed model follows the evolution of the distribution of these random variables and in particular the distribution obtained in stationarity. Non parametric equations are developed for distributions of x_i and v_i, whereas the distribution for b_i is described in terms of 1 st. and 2 nd. moments. To ease exposition, velocities are assumed to take values within a discrete set.
Each separate model for resource sharing, charging and consumption are given as integro-differential equations governing the time evolution of the conditional expectation $b_i(t,x,v) = E(b_i(t)|x_i(t) = x, v_i(t) = v)$ of the resource $b_i(t)$ carried by robot i, given it is located at x at time t, with velocity v. Likewise a second moment model is developed for the time evolution of the conditional variance $V_i(t,x,v) = VAR(b_i(t)|x_i(t) = x, v_i(t) = v)$.

2.2.1 Mobility

Robot populations may be distributed randomly over some domain D, or they may be deployed according to some predefined plan. Additionally robots may be stationary or mobile, e.g. a subset of robots may be assigned highly stationary tasks, whereas a majority of units would be mobile.
A model of robot distribution should account for both deterministic deployment and random distribution. We assign to each robot i the time dependent probability measure L_i of location, i.e. $L_i(A,t)$ expresses the probability that node i is located within the subset A of D, at time t. Adding up for the entire set of nodes yields the additive positive measure L, i.e.

$$L(A,t) = \sum_i L_i(A,t) \tag{1}$$

where $L(A,t)$ expresses the expected number of robots within A at time t. Joint location and velocity measure is assumed to be expressed as

$$L_i(A \times W, t) = \int_A L_i(W|x,t) f_L^i(x,t) dx \tag{2}$$

i.e. position distribution is described by a density f_L^i, whereas velocity has a general conditional distribution $L_i(W|x)$. The notation L_i is consistently used for kinetic state (position and velocity) distribution of robot i. The corresponding argument list indicates the specific perspective in question.
Mobility affects resource distribution in two ways; it changes robot location distribution over time and secondly it changes the conditional resource distribution over time. The former effect is considered in this section whereas the latter is presented in the next section based on the stationary location distribution obtained in this section. Several mobility models exist, including deterministic as well as random movement. Among others we find the *Random waypoint*[Bettstetter et al, 2004], *Random direction* and *Random trip*[Le Boudec et al, 2005] an overview is presented in [Camp et al, 2002]. In this work we consider mobility models both suited for probabilistic modelling of which a special case is Brownian motion [Øksendal et al, 2003]. Brownian motion is characterized by its lack of velocity persistence, i.e. units move without memory of previous direction and speed. A more elaborate model is presented subsequently, which includes velocity persistence.

2.2.2 Less Drunk Model

The Less Drunk Model (LDM) is as Brownian motion a Markovian mobility model, where velocity remains constant between instants $..,t_{-1}t_0,t_1,..$ of velocity change. Time intervals of constant velocity have random exponentially distributed length, where the intensity parameter λ may depend on position, velocity and time. At an instant t_n of velocity change, a future velocity is selected randomly and independently from a probability distribution L_Q assumed to depend on position.
So equipped we may deduce the following location distribution dynamics

$$\begin{aligned} & \frac{d}{dt}(L_i(W|x,t) \cdot f_L^i(x,t)) \\ = \; & \lambda(x)(L_Q(W,x) - L_i(W|x,t)) f_L^i(x,t)) \\ - \; & \int_W < v, D_x[L_i(dv|\cdot,\cdot) f_L^i](x,t) > \end{aligned} \tag{3}$$

where <,> denotes inner product. The second term generalises continuous and discrete parts of L_i, i.e.

$$\begin{aligned} & \int_W < v, D_x[L_i(dv|\cdot,\cdot) f_L^i](x,t) > \\ = & \int_W < v, D_x[f_i(dv|\cdot,\cdot) f_L^i](x,t) > \\ + & \sum_{v_j \in W} < v, D_x[L_i(\{v_j\}|\cdot,\cdot) f_L^i](x,t) > \end{aligned}$$

As an example, L_Q may for all x concentrate probability on a discrete set of velocities $\{v_j\}$. Letting $W = \{v_j\}$ gives

$$\begin{aligned} & \frac{d}{dt}(L_i(\{v_j\}|x,t) \cdot f_L^i(x,t)) \\ = \; & \lambda(x)(L_Q(\{v_j\},x) - L_i(\{v_j\}|x,t))f_L^i(x,t) \\ - \; & < v_j, D_x[L_i(\{v_j\}|\cdot,\cdot)f_L^i](x,t) > \end{aligned} \tag{4}$$

Confinement of motion to a Consider a one-dimensional example, where velocities assume values 1 and -1.

$$\begin{aligned} L_Q(\{1\},x) &= 1/2 \;\text{ for }\; -1 \le x \le 1 \\ &= 0 \;\text{ for }\; x > 1 \\ &= 1 \;\text{ for }\; x < -1 \\ L_Q(\{-1\},x) &= 1/2 \;\text{ for }\; -1 \le x \le 1 \\ &= 0 \;\text{ for }\; x < -1 \\ &= 1 \;\text{ for }\; x > 1 \end{aligned} \tag{5}$$

for a constant value $\lambda(x) = \lambda$ for x outside $[-1,1]$ (4) possesses the following particular stationary solution

$$\begin{aligned} L_i(\{-1\}|x,t)f_L^i(x,t) &= L_i(\{-1\}|x,t)f_L^i(x,t) \\ &= C_0 \quad \text{for} \quad x \in [-1,1] \\ &= C_0 \; exp(\lambda(x-1)) \quad \text{for} \quad x \in (-\infty,-1) \\ &= C_0 \; exp(\lambda(1-x)) \quad \text{for} \quad x \in (1,\infty) \end{aligned} \tag{6}$$

where $C_0 = 1/4(1+1/\lambda)$ is found from normalization. Stationary solutions for various λ values are shown in figure (1). For large λ, C_0 is approximately $1/4$ and $f_L^i(x,t) = 1/2$ in side D.

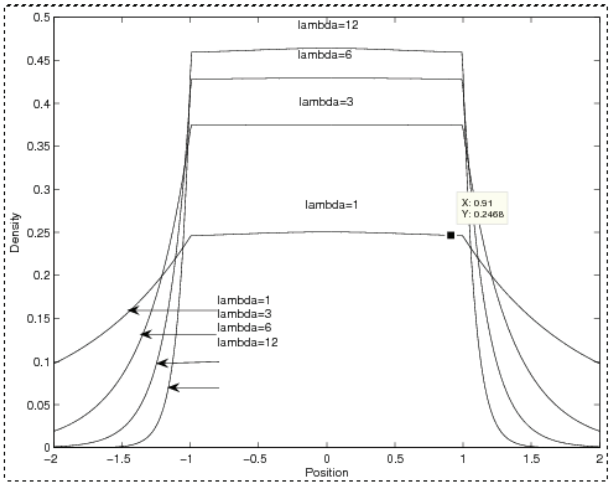


Figure 1. Stationary location densities for various λ-values

2.2.3 Mobility Impact on Energy Distribution

Consider a small sub domain A at times t and $t+\delta$. Not considering exchange nor consumption, the amount of energy resources carried by robot i in A at time $t+\delta$ is identical to the amount of resources moving (along with robot i) into $A \times \{v_i\}$ during $[t, t+\delta]$. The expected energy resource $E_{A\times\{v_j\}}(t+\delta)$ in $A \times \{v_i\}$ at time $t+\delta$, is expressed through the conditional resource given position and velocity $b_i(t,x,v)$, i.e.

$$E_{A\times\{v_j\}}(t+\delta) = \int_A b_i(t+\delta, x, v_j) L_i(\{v_j\}|x) f_L^i(x) dx \quad (2.3.1) \tag{7}$$

whereas the amount carried into $A \times W$ by i is found by conditioning on position and velocity x' and v' at time t and marginalizing, i.e.

$$\begin{aligned} & E_{A\times\{v_j\}}(t+\delta) \\ & +(1-\lambda\delta) \int_A b_i(t, x - v_j\delta, v_j) f_L^i(x - v_j\delta)\ L_i(\{v_j\}|x - v_j\delta)\ dx \\ & +\lambda\delta \int_A L_Q(\{v_j\}, x) \sum_k\ b_i(t, x - v_k\delta, v_k) f_L^i(x - v_k\delta)\ L_i(v_k|x - v_k\delta)\ dx \end{aligned} \tag{8}$$

Equating expressions (7) and (8) for every subset A and differentiating w.r.t. δ, we may show that $b_i(t,x,v)$ fulfils

$$\begin{aligned} & f_L^i(x) L_i(\{v_j\}|x)\ \frac{d}{dt} b_i(t,x,v_j) \\ & \quad = - < v_j, D_x[b_i\ f_L^i\ L_i](t,x,v_j) > \\ & \qquad + \lambda\ (L_Q(\{v_j\}, x)\ b_i(t,x) - \ L_i(\{v_j\}|x)\ b_i(t,x,v_j)) f_L^i(x) \end{aligned} \tag{9}$$

where

$$b_i(t,x) = \sum_k b_i(t,x,v_k) L_i(\{v_k\}|x) \tag{10}$$

Equivalent results can be found for then conditional second moment $b2_i$

$$\begin{aligned} & f_L^i(x) L_i(\{v_j\}|x)\ \frac{d}{dt} b2_i(t,x,v_j) \\ & \quad = - < v_j, D_x[b2_i\ f_L^i\ L_i](t,x,v_j) > \\ & \qquad + \lambda\ (L_Q(v_j, x)\ b2_i(t,x) - L_i(\{v_j\}|x)\ b2_i(t,x,v_j)) f_L^i(x) \end{aligned} \tag{11}$$

2.2.4 Numerical Example

We continue the numerical example from above and extend it with expressions and results for energy distribution. We assume the stationary solution (6) for $f_L^i\ L_i$ for each robot i. Likewise we assume λ to be high outside D and thereby neglecting movement outside D. Thus we have $f_L^i\ L_i = 1/4$ inside D and zero outside. Likewise

$$\begin{aligned} & < v_j, D_x[b_i\ f_L^i\ L_i](t,x,v_j) > = 1/4 < v_j, D_x[b_i](t,x,v_j) > \\ & < v_j, D_x[b2_i\ f_L^i\ L_i](t,x,v_j) > = 1/4 < v_j, D_x[b2_i](t,x,v_j) > \end{aligned} \tag{12}$$

altogether we have from (9) and (11)

$$\begin{aligned}\frac{d}{dt}b_i(t,x,v_j) &= - < v_j, D_x[b_i](t,x,v_j) > +\lambda\ (b_i(t,x) - b_i(t,x,v_j))\\ \frac{d}{dt}b2_i(t,x,v_j) &= - < v_j, D_x[b2_i](t,x,v_j) > +\lambda\ (b2_i(t,x) - b2_i(t,x,v_j))\end{aligned} \tag{13}$$

Inserting discrete velocities $\{1\}$ and $\{-1\}$ gives

$$\begin{aligned}\frac{d}{dt}b_i(t,x,-1) &= b_i'(t,x,-1) + \lambda/2(b_i(t,x,1) - b_i(t,x,-1))\\ \frac{d}{dt}b_i(t,x,1) &= -b_i'(t,x,1) + \lambda/2(b_i(t,x,-1) - b_i(t,x,1))\\ \frac{d}{dt}b2_i(t,x,-1) &= b2_i'(t,x,-1) + \lambda/2(b2_i(t,x,1) - b2_i(t,x,-1))\\ \frac{d}{dt}b2_i(t,x,1) &= -b2_i'(t,x,1) + \lambda/2(b2_i(t,x,-1) - b2_i(t,x,1))\end{aligned} \tag{14}$$

revealing that, when mobility is studied in isolation, stationary solutions for expected battery resources as well as second moments are constant over D, which coheres well with intuition.

2.2.5 Energy Transfer

Energy exchange is in this work considered to be an unplanned epidemic process, i.e. transfer of energy between robots take place during accidental rendezvous. Epidemic propagation is previously studied in other contexts, such as disease spread [Medlock et al, 2003] and information spread [Schiøler et al, 2005],[Moreno et al, 2004]. All mobile units are assumed to move randomly in patterns generated by a Less Drunk mobility process as described above. When two robots come within a suitable (not too large) distance to each other, conditions promote energy exchange as illustrated in figure (2).

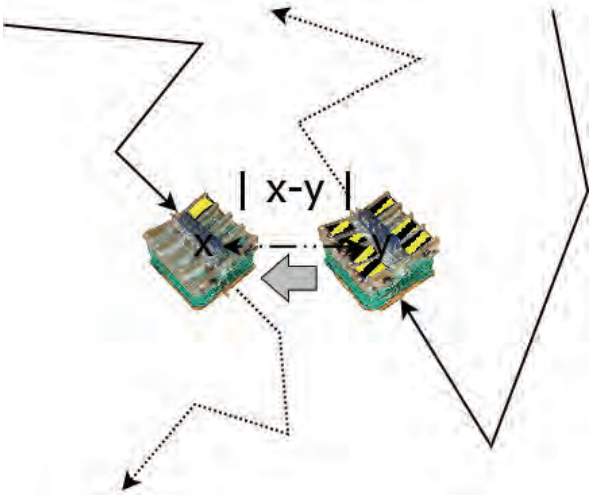


Figure 2. Two robots in accidental rendezvouz, candidating for energy exchange

More precisely two robots i and j positioned at positions $x = x_i$ and $y = x_j$ respectively are assumed to engage in a battery exchange within the time interval $[t, t+dt]$ with a probability $\alpha\ dt\ K(x_i, x_j)$, where α is a rate parameter and K is a *neighbourhood kernel* modelling the dependence of relative/absolute positions on exchange probability. The decision to engage in battery exchange is taken randomly and represented by the random Boolean selector Ce_{ij}, where $P(Ce_{ij}) = \alpha\ dt\ K(x_i, x_j)$. At time t robots i and j mutually communicate remaining battery resources $b_i(t)$ and $b_j(t)$ respectively. The final choice of battery exchange is taken randomly and represented by the random Boolean selector Cf, where

$$P(Cf_{ij}) = C|b_i(t) - b_j(t)| \tag{15}$$

where C is chosen, so that $P(C_f) \leq 1$ always. If exchange is decided, a fixed size quantity Q is exchanged, where $Q = |Q|\ sign(b_j(t) - b_i(t))$. Altogether the exchange dynamics for two robots can be written as

$$b_i(t+dt) = b_i(t) + Ce_{ij}\ Cf_{ij}\ |Q|\ sign(b_j(t) - b_i(t)) \tag{16}$$

Potentially i may exchange batteries with every other robot j in the entire population, so the overall exchange dynamics can be written like

$$b_i(t+dt) = b_i(t) + \sum_{j \neq i} Ce_{ij}\ Cf_{ij}\ |Q|\ sign(b_j(t) - b_i(t)) \tag{17}$$

When robot positions are unknown, a location measure L_i is associated to each robot i. Likewise we define $b_i(t, x, v)$ to be the conditional expectation of $b_i(t)$ given i is positioned at x with velocity v at time t. Thus from (16)

$$\begin{aligned} b_i(t+dt, x, v) &= b_i(t, x, v) + |Q| \sum_{j \neq i} E[Ce_{ij} Cf_{ij}\ sign(b_j(t) - b_i(t))] \\ &= b_i(t, x, v) + |Q| C\alpha dt \sum_{j \neq i} E[K(x, x_j)|b_j(t) - b_i(t)| sign(b_j(t) - b_i(t))] \\ &= b_i(t, x, v) + |Q| C\alpha dt \sum_j \int_D K(x, y^{x_j}) b_j(t, y) - b_i(t, x, v)) L_j(dy) \end{aligned} \tag{18}$$

Where velocity is marginalized away in b_j, i.e.

$$b_j(t, y) = \int b_j(t, y, v) L_j(dv|x) \tag{19}$$

Adding location measures ($L = \sum_j L_j$), leads from (18) to

$$\begin{aligned} \frac{d}{dt} b_i(t, x, v) = |Q| C\alpha \sum_j \int_D K(x, y) b_j(t, y) L_j(dy) \\ - |Q| C\alpha\ b_i(t, x, v) \int_D K(x, y) L(dy) \end{aligned} \tag{20}$$

and for the conditional second moment $b2_i(t,x,v)$ of $b_i(t)$ given position x and velocity v at time t.

$$\begin{aligned}&\frac{d}{dt}b2_i(t,x,v) = \\ &2|Q|C\lambda\ (b_i(t,x,v)\sum_j \int_\Omega K(x,y)b_j(t,y)L_j(dy) - b2_i(t,x,v)\int_\Omega K(x,y)L(dy)) \\ &+|Q|C\lambda\ \sum_j \int_\Omega K(x,y)|b_j(t,y) - b_i(t,x,v)|L_j(dy)\end{aligned} \tag{21}$$

2.2.6 Charging Station

Charging stations may be considered as only robot units serving special objectives. Formally we define a robot i to be a charging station, when $i \in CS$, where CS is the index subset for charging stations.

Specific to charging stations is the fact, that batteries should never be received by these, and additionally that they may move according to a specific mobility patterns. With respect to the former exception we exclude from the model the resource level of charging stations and simply assume resource levels always to assume an upper bound, i.e. $b_i(t,x) = \bar{b}\ \ \forall\ t,x$. This excludes the possibility of battery units to be handed over to charging stations. Likewise it may be desirable to have separate control of the exchange rate from the charger. Thus we set the exchange rate parameter for the charger by $\alpha_C = r_C\alpha$, where r_C is a positive real typically > 1.

Regarding mobility of charging stations, they may as a first suggestion be stationary at known locations. Location measure L_i for a charging station is in, this case, concentrated at a particular point x_C, i.e. $L_i(\{x_C\}) = 1$. Even for stationary charging stations, locations may be unknown, in which case locations are specified according to some a priori measure L_i. For non stationary charging stations some mobility model may be assumed and L_i may be time dependent converging to a stationary measure as for robot units.

2.2.7 Example

Continuing the above example we have for mobile units $L_j(A) = |A|/|D|$. Furthermore we assume $b_i(t,x,v) = b(t,x,v)$ for all mobile units i, whereas $L_i(A) = I_A(x_c)$ and $b_i(t,x,v) = \bar{b}$ for a single charging station located at a fixed position x_C. Assuming N robots equations (20) and (21) yield

$$\begin{aligned}&\frac{d}{dt}b(t,x,v) \\ &= |Q|C\alpha N/|D| \\ &\cdot \int_D K(x,y)\ (\frac{b(t,y,\{-1\}) + b(t,y,\{1\})}{2} - b(t,x,v))\ dy \\ &+|Q|C\alpha\ r_C\ K(x,x_C)(\bar{b} - b(t,x,v))\end{aligned} \tag{22}$$

and for the conditional second moment $b2(t,x,v)$

$$\begin{aligned}
&\frac{d}{dt}b2(t,x,v) = \\
&-2|Q|C\alpha\ b2(t,x,v)(N/|D|\int_D K(x,y)dy + r_C\ K(x,x_C)) \\
&+2|Q|C\alpha\ 2b(t,x,v) \\
&\cdot(N/|D|\int_D K(x,y)\frac{b(t,y,\{-1\})+b(t,y,\{1\})}{2}\ dy + r_C\bar{b}K(x,x_C)) \\
&+|Q|C\alpha \\
&\cdot(N/|D|\int_D K(x,y)|b_j(t,y)-b_i(t,x,v)|dy \\
&+r_C\ K(x,x_C)|\bar{b}-b_i(t,x,v)|)
\end{aligned} \tag{23}$$

2.2.8 Energy Consumption

Various models for energy consumption in mobile robotics are suggested in literature [Mei et al, 2006a, 2006b]. In this case choosing a suitable model involves a trade-off between precision and mathematical tractability. The rate of energy consumption may depend on various parts of the system state, i.e. on aspects of the state of the entire population as well as the state of the individual robot. Since robots may be equipped with energy preserving activity policies, their individual activity may depend on their remaining energy resources. Taking such behaviour into account may be achieved by letting consumption rate depend on remaining resources. In this case we suggest a Poisson modulated model, i.e.

$$b_i(t) = b_i(0)\cdot exp^{-n}(-r/\gamma) \quad \text{for} \quad t\in[t_n,t_{n+1}] \tag{24}$$

where $\{t_n\}$ is an increasing Poisson generated sequence of time instants, where remaining battery resources are discounted through multiplication by $exp(-r/\gamma)<1$ so that (2.8.1) exhibits an expected exponential consumption profile, i.e.

$$\begin{aligned}
&E(b_i(t)|x_i(t)=x,v_i(t)=v) = b_i(t,x,v) \\
&= b_i(0,x,v)\cdot exp(-\gamma t(1-exp(-r/\gamma))
\end{aligned} \tag{25}$$

which, for large values of γ can be approximated by

$$b_i(t,x,v) = b_i(0,x,v)\cdot exp(-rt) \tag{26}$$

For our Poisson modulated consumption model (23), we may deduce

$$\begin{aligned}
&\frac{d}{dt}b_i(t,x,v) = \gamma\ (exp(-r/\gamma)-1)\ b_i(t,x,v) \\
&\frac{d}{dt}b2_i(t,x,v) = \gamma\ (exp(-2r/\gamma)-1)\ b2_i(t,x,v)
\end{aligned} \tag{27}$$

which for large values of γ can be approximated by

$$\begin{aligned}
&\frac{d}{dt}b_i(t,x,v) = -r\ b_i(t,x,v) \\
&\frac{d}{dt}b2_i(t,x,v) = -2r\ b2_i(t,x,v)
\end{aligned} \tag{28}$$

2.2.9 Complete Model

A complete model is presented which combines the effects of mobility, energy exchange and energy consumption. The developed model assumes the shape of integro-differential equations governing the time evolution of the conditional expectation $b_i(t,x,v)$ of the battery resource $b_i(t)$ of robot i given this robot is located at position x at time t, with velocity v. Likewise integro-differential equations for the conditional variance $b2_i(t,x,v) = E(b_i^2(t))$ are given. The model is developed for stationary location distributions.

All individual model parts (mobility, exchange, consumption) are developed from elementary dynamics giving $b_i(t+\delta)$ from $b_i(t)$ for an infinitesimal time step δ, i.e. $b_i(t+\delta) = b_i(t) + D_M(\delta)$, $b_i(t+\delta) = b_i(t) + D_E(\delta)$, $b_i(t+\delta) = b_i(t) + D_C(\delta)$, where D_M, D_E and D_C are random variables modelling randomized mobility, energy exchange and energy consumption respectively. Thus the complete integro-differential equation for conditional expectation is found as

$$\frac{d}{dt}(b_i(t,x,v)) = \frac{d}{d\delta}E[D_M(\delta)] + \frac{d}{d\delta}E[D_E(\delta)] + \frac{d}{d\delta}E[D_C(\delta)] \tag{29}$$

D_M, D_E and D_C are assumed independent, being continuous at $\delta = 0$ and having 1st. and 2nd. moments with finite non-zero 1st. derivatives at $\delta = 0$. This allows aggregation of separate model components for conditional 2nd. moments by addition i.e.

$$\frac{d}{dt}b2_i(t,x,v) = \frac{d}{d\delta}E[D_M^2(\delta)] + \frac{d}{d\delta}E[D_E^2(\delta)] + \frac{d}{d\delta}E[D_C^2(\delta)] \tag{30}$$

2.2.10 Example

The complete model is illustrated by examples combining the previous examples in this chapter. It is not possible to provide an overview of results for the entire parameter space, so therefore only a few illustrative examples are shown. Parameter settings for the provided examples are selected below to mimic a realistic situation. It is basically assumed that all robots inhabit a one-dimensional domain of operation $D = [-1,1]$ and move with two possible speeds $\{-1,1\}$. Thus crossing the entire domain without speed changes lasts 2 time units.

For the mobility parameter λ we assume robots to change velocity 10 times for each such 2 time units, i.e. $\lambda = 5$.

In order for an energy propagation mechanism to be worthwhile, a significant power loss should be associated with travelling from the peripheral of the domain of operation to the charger. Thus we assume, that a direct travel half way across D discounts the energy resources by 2/3, i.e. $exp(-r) = 0.3$ or $r \approx 1$.

Regarding energy exchange, we normalize the charger resource by $\bar{b} = 1$ defining an upper bound for b_i. In accordance we set $C = 1$ and $Q = 1/10$, that is, the energy quantum exchanged is far lower than the upper bound for remaining resource. The neighbourhood kernel K is assumed to allow energy exchange within a fixed distance $d = 0.1$, i.e. $K(x,y) = I_{|x-y|<0.1}$. The charging process is assumed to be faster than the energy consumption process. Thus we set $\gamma < \alpha_C = 40$. The mutual robot exchange rate α is varied to illustrate its effect on energy distribution. A charger placed at a fixed location $x_C = 0$ serves $N = 50$ robots.

2.11 Survivability

Energy resources at each robot needs to be above a certain critical lower level L to maintain robot functionality. Below this level robots are no longer capable of moving, communicating or exchanging energy. Thus energy levels below L implies irreversible entrance to a *death* state. The suggested consumption model above prescribes consumption to take place at discrete moments $\{t_n\}$ in time, where energy resources are discounted by a factor $exp(-r/\gamma)$. Every robot holding an energy level less than $Lexp(r/\gamma)$ is therefore a candidate for entering the death state at the next discrete consumption instant t_n. Since $\{t_n\}$ is assumed to be a homogeneous Poisson process with intensity γ the death rate associated to such a robot is γ. Likewise we may find the overall expected death rate ρ_D of the population by

$$\rho_D = P(b(t) \leq L\ exp(r/\gamma))\ N\ \gamma \tag{31}$$

Approximating the conditional stationary distribution of $b_i(t)$ by a normal distribution we get

$$\rho_D = \int_D erf(L\ exp(r/\gamma) \cdot \sigma_i(x,v) - b_i(x,v))L_i(dv|x)f_i(x)dx\ N\ \gamma \tag{32}$$

where $\sigma_i = \sqrt{b2_i - b_i^2}$ is the conditional standard deviation and $erf()$ is the error function. Figures (3) and (4) show stationary energy distributions for values of α 0.5 and 10. Corresponding death rate values are $\rho_D = 268$ and $\rho_D < 2E-16$, where the latter indicates a result below machine precision. Thus the effect of the mutual exchange rate is rather dramatic.

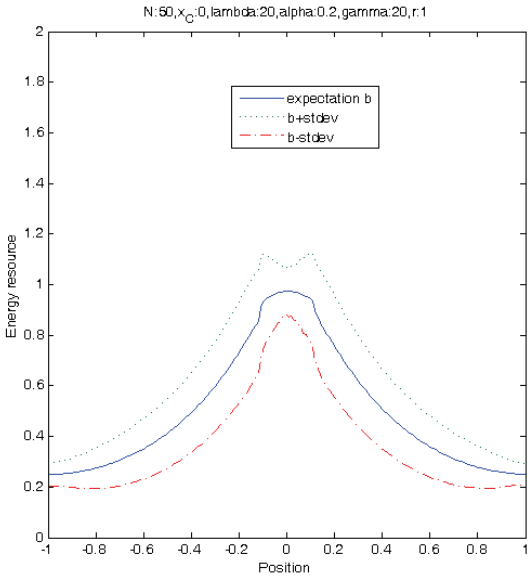


Figure 3. Energy distributions for low level of $\alpha = 0.2$

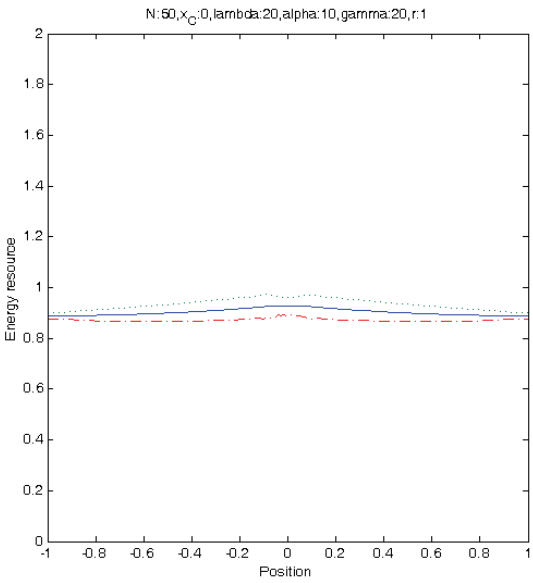


Figure 4. Energy distributions for high level of $\alpha = 10$

An increased mutual exchange rate increases the flow of energy away from the neighbourhood of the charger, which in turn allows more flow from the charger to its neighbourhood. Additionally, mutual exchange transports energy resources to the peripheral of D increasing survival far away from the charger. As seen from figures (3) and (4) mutual exchange levels energy resources among robots and in turn reduces variance and improves survival.

3. Biologically Inspired Robot Trophallaxis Simulation

3.1 An introduction to Biologically Inspired Robot Trophallaxis

The term "trophallaxis" is simply defined as mutual exchange of food between adults and larvae of certain social insects or between parents and offspring of vertebrate animals [Camazine, 1998]. In other words, trophallaxis is the regurgitation of food by one animal for the other in a colony. This phenomenon is mostly observed from social insects e.g., ants, fireants, bees, or wasps. For instance, *food* is exchanged among adults and larvae in the ants' trophallaxis process. The ant workers carry baits back to the colony's nursery. Because adult ants cannot actually digest *solid foods*, the bait is fed to the larvae which digest the material and regurgitate the baits in a *liquid form* back to adult ants. In turn, these ants feed other members of the ant colony. In this manner, ant baits are spread throughout the targeted ant colony. Without trophallaxis the ant bait would not penetrate the *gigantic organism* constituted by the ant colony. The phenomenon is also seen from vertebrate animals e.g., birds or wild dog. For example, bird parents looks for food to store it in their crops when *far away* from the nest. To feed their offspring, they fly back to the nest and *regurgitate* foods to transfer to their young. Trophallaxis is also performed by members of the dog family. In the wild, a hunting dog will regurgitate food *gorged* when far from its lair in order to feed its

puppies. To *trigger* trophallaxis, these puppies *lick* the face of their parents. For domestic dogs, they are tame because of arrested development, and will treat with certain humans, in particular their *owner,* as their "parents". Therefore, a dog may manifest a vestigial feeding instinct when it licks human face.
Besides trophallaxis, *pheromones* [Sumpter et al, 2003],[Payton et al, 2005], act as agents to keep all members within the group. For example, the ant queen produces a special pheromone without which the workers will begin raising a new queen.
In short, "trophallaxis" obtains the meanings of *food reproduction* and *food exchange* while "pheromones" is implicitly recognized as means of communication, *global agents* and *local agents*. In details, 1) ant larvae digesting solid food into liquid form and bee pupa digesting nectar into honey are good examples of the *foods reproduction* phenomenon, 2) bird parents feeding their offspring, hunting dogs regurgitating foods for their puppies, ant larvae returning liquid baits to ants, and ants feeding the others typically manifest the phenomenon of *foods exchange*, 3) ants or bees also lay down their pheromones along their trails as *global agent*s to group all colony members together, 4) puppies lick their parents to trigger the trophallaxis of regurgitated foods or nestlings rub their beak to their parents' one as *local agents* for the trophallaxis.
Inspired from the natural phenomena, we have created a system of multiple autonomous mobile robots that is capable of performing energy trophallaxis to sustain robots' life without human intervention. This immediately rises a central question: what are the minimal requirements to achieve energy trophallaxis in multiple mobile robots? Some answers can be found the following section where the meaning of "Randomized Robot Trophallaxis" is clarified.

3.2 The "Randomized Robot Trophallaxis" Concept

The term "autonomous robot" is widely used to define robotic systems to function without human intervention. In fact, people have attempted to build systems, which could operate without human control. However, the term "autonomy" [Ieropoulos et al, 2004] is difficult to assess due to policy of inventors, which are leading to ambiguous meaning in use. In our opinion, a truly autonomous robot is a robot that must obtain two policies: *behavioral autonomy* and *energetic autonomy* in which behavior and energy are closely related. Until now, the term "autonomy" in robotics has mostly been addressed in the sense of "behavioral autonomy" only, not including "energetic autonomy".
In the further perspective, we have paid interest especially to large populations of mobile robots in which each robot is a truly autonomous agent. But, like animal societies, a potential method to achieve entire autonomy is that robots must demonstrate the capabilities of *energy trophallaxis* obtaining two functionalities: the self-refueling energy and the self-sharing energy. However, due to the randomized robot behaviors in large populations, obviously based on assigned tasks, the energy trophallaxis could be *randomized*. That is, the desired robots have to independently perform not only individual behaviors but also cooperative behaviors to achieve energy trophallaxis randomly.
Next we attempt an answer to the question of minimal requirements appearing in the previous section:
Foods reproduction:
Most electronic vehicles are nowadays equipped with rechargeable batteries to power their executions. In particular, for mobile robots, rechargeable batteries seem presently to be the

best solution. Thereby, rechargeable batteries are considered as "foods" and "foods reproduction" is the process of refueling battery stored energy. A few previous systems e.g., Roomba vacuuming[2] robots, mentioned "foods reproduction" as a docking station where a robot can move back to dock with the station for battery recharging. Unlike the recharging process of Roomba robots, animal trophallaxis includes the exchange of "foods" from one to another other. Inspired from the foods reproduction of animals e.g., *solid foods* digested into *liquid foods*, we create a charging station where hundreds of rechargeable batteries are automatically recharged and available to mobile robots.

Foods exchange:

Like the phenomenon where bird parents feed their offspring, hunting dogs regurgitate foods for their puppies, or ant larvae returns liquid baits to ants, and ants shares baits to the others, "foods exchange" through direct "mouth-to-mouth" contact is the key to achieve *energetic autonomy*. It requires a robot to have a battery exchange mechanism that allows batteries to be exchanged to other robots. Comparing with the method of battery charging, this approach holds the potential for saving much time of electrical energy transfer. However, ants, bees or dogs can exchange/feed its foods to the other if and only if they can find heir colony/family members. Similarly, the self-sharing energy process of mobile robots is completely successful if and only if a robot is capable of searching the other and establishing a "mouth-to-mouth" contact with the other. A battery exchange mechanism is purely required to perform the energy trophallaxis through "mouth-to-mouth contacts". Indeed, the former is global agents in a colony while the latter is local agents between two colony members. Features of the agents will be explained in details next sections

Global agents:

Natural stigmergy is a concept to describe a method of *indirect communication* [Payton et al, 2005] in a self-organizing emergent system where its individual parts communicate with one another by *modifying their local environment*. In particular, ants communicate to one another by laying down pheromones along their trails, i.e. where ants go within and around their ant colony is a stigmergic system. However, stigmergy is not restricted to eusocial creatures in growth. For examples, in passive way, birds rely on the earth magnetic field to emigrate in the winter. In active way, a pole-cat marks its own areas by spreading out its feces while another pole-cat enlarges their own area by moving the poops. Inspired from the natural behaviors, we define "global agents" as "agents" that are able to keep communication of all colony members together or to manage their own behaviors in relation with other members in the colony. In our experimental setup, a pre-built grid map on which mobile robots can follow lines is the "classical stigmergy" inspired solution. For the "evolved stigmergy", using external sensors e.g., compass to estimate related orientation among robots, infrared array to detect lines are methods to enable robots being aware of their locations. However, to overcome the limit of "stigmergy", global radio frequency communication may be a good choice to complement *indirect communication.*

Local agents:

Trophallaxis between two colony members is successfully completed if and only if they are able to communicate or activate the trophallatic state in each other simultaneously. For examples, puppies will lick their parents to start the foods regurgitation when they are hungry. Thereby, licking or rubbing are local agents between two individuals engaged in trophallaxis. Similar to the dialogue of animals, a line of sight infrared local communication

[2] See www.irobot.com

complemented by contact detection systems within each robot is typically required for trophallaxis process to be successful.
In particular, we have developed a new prototype of robots, named CISSbot capable of performing energy trophallaxis in three forms: robots with mother-ship, robots with robots, and robots with their child. In other words, the robots are capable of carrying out not only energetic autonomy but also behavioural autonomy. The realization of the robots is on the one hand expected to redefine the definition of "autonomy" in robotics. On the other hand, the unique design can suggest a new method to generate truly autonomous robots in large populations.

3.1 Simulation of Randomized Robot Trophallaxis

In this section we address simulated results of energy trophallaxis in terms of self-refuelling energy and self-sharing energy. Like animal life, we assume that a group of mobile robots share a nest, that is, a charging station where they can come back to refuel energy. A simulation setup can be seen in figure 5. The simulation state is shown in four windows (from left to right): Motion, Energy Distribution, States of Energy, and Tasks.
We firstly establish an energy model for single robots. Obviously, battery measure is the best way to estimate the remaining energy of a robot at an instant. However, because the energy consumption model is not uncertain to every robot due to its own mechanism, control, assigned tasks, etc., it is hard to model battery measure for a robot. Therefore, we temporally choose Peurket's discharging function $C = I^{k}t$ where k is supported by the battery manufacturer since the function is close to the linear equation of experimental power consumption of a mobile robot

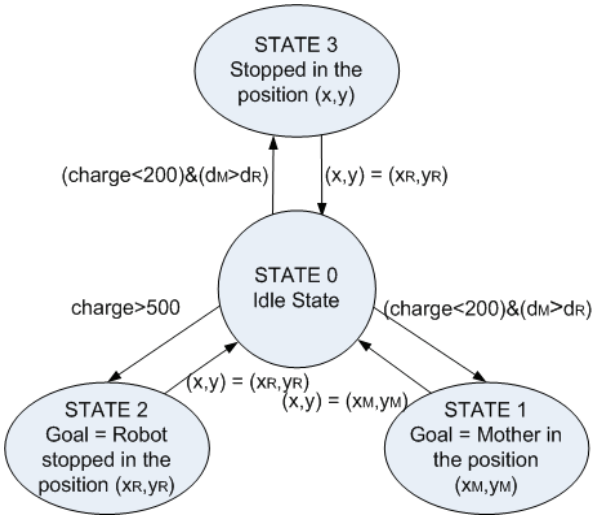


Figure 5. Model of single robot

Basically a robot is initialized with 800 energy units (**eu**) corresponding to the 8 battery holder of every robot. The robot consumes a specific amount of energy, using Peukert's equation, for each step. We propose 4 energy states of robot corresponding to behaviours and energy states:

- State 1 is an interaction between a robot and the mother charging station in the organization. Arobot has to go to the mother charging station to refill energy if its energy is less than 200 **eu**, and by default, it has a higher priority to go to the mother charging station.
- State 2 is an interaction between two robots on demand in a organization. A robot is able to exchange 100 **eu** with anoother robot demand if its energy amount is more than 500 **eu.**
- State 3 is also an interaction between two robots in organization and their interaction with the environment (for example, due to an assigned task), but it is different from the State 2. A robotic agent will stop to wait for another robot coming to share 100 **eu** if its energy is less than 100 **eu** and it is impossible to go to the mother charging station due to its estimation of the relative distance and remaining energy.
- State 0 is an interaction between a robot and its environment (for example, obstacle avoidance among robots, and between robots and lateral walls). A robotic agent is autonomously free to explore in order to consume energy.

To approach a solution for battery exchange quickly, we suppose a coordination algorithm for the multi-robot system based on two phases: path planning and battery exchange. The algorithm is proposed to emphasize the interaction of agents irrespective of their surrounding environment which should be taken into account in practice.

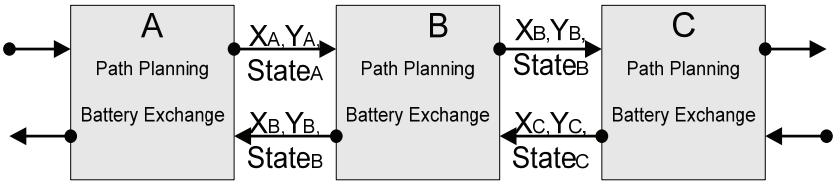


Figure 6. Model of multi-robot system coordination

Briefly, each robot has its own battery exchange supervisor. The supervisor collects input data from the robots, e.g., the current coordinate (X,Y) and the current energy state STATE; deals with this updated data; and issues output commands, e.g., NEXT STATE of energy, goal coordinate (Xgoal, Ygoal). A more detailed algorithm of the battery exchange executes infinite loops of comparisons of energy states and current positions among the robots as well as the robot with the mother, in order to give commands about what the robot should do next (the goal of the robot). Meanwhile, the path planner guides the robot to reach the directed goal and update the next position, which is used as feedback for the battery exchange algorithm to compute the next states (fig.6) Detailed information of the simulation setup can be found in [Ngo et al, 2007].

Firstly, inspired from the instinct of self-preservation in ant colonies where worker ants return to the nest to eat a liquid foods produced by the larvae, a simulation of self-refuelling energy is performed to demonstrate the capability of self-refuelling energy. Secondly, like feeding of bird or dog parents to their offspring, we establish a simulation to demonstrate the capability of self-sharing energy among robots. Thirdly, we examine a combination of self-refueling energy and self-sharing energy to point out an efficient solution for energetic

autonomy in mobile robots. Finally, we discuss problems related to meaning of "randomization" in terms of initializations, motion, and energy distribution.

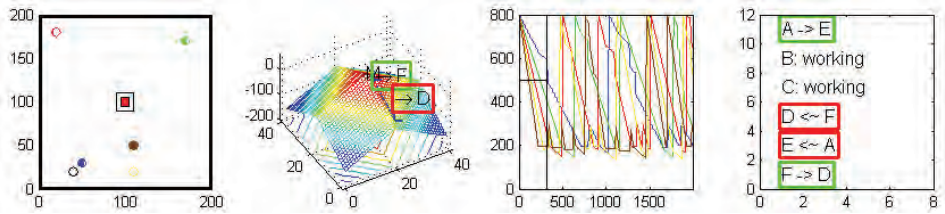


Figure 7. Simulation screen: Motion (over left), Potential of Energy (left), States of Energy (right), Tasks (over right)

Firstly, inspired from the instinct of self-preservation in ant colonies where worker ants return to the nest to eat a liquid foods produced by the larvae, a simulation of self-refuelling energy is performed to demonstrate the capability of self-refuelling energy. Secondly, like feeding of bird or dog parents to their offspring, we establish a simulation to demonstrate the capability of self-sharing energy among robots. Thirdly, we examine a combination of self-refueling energy and self-sharing energy to point out an efficient solution for energetic autonomy in mobile robots. Finally, we discuss problems related to meaning of "randomization" in terms of initializations, motion, and energy distribution.

3.3.1 Self-refueling Energy Based on "Food Reproductions"

A simulation of 2000 running steps is set up to examine how many time robots need return to the charging station to refuel their energy. To model energy consumption we assigned different energy cost functions for every robot as the Peukert's equation. Virtual pheromones based on Euclidean distance are used to guide robots to the charging station placed at the center of scenario. Every robot is continuously managing its resources by estimating remaining energy, but not estimating the distance from its current position to the charging station.

Initially, robots are equipped with fully charged batteries and randomly deployed in the scenario. The robots are freely moving around to spend energy based on energy cost functions and returning to the charging station when energy is low. Table 1 shows the record of number of energy refuelling events in 2000 steps, which can correspondingly calculate the total energy consumption of each robot with respect to energy state. Truly, the result corresponds to the energy cost functions (workloads) assigned to individual robots, which are increased from A to E. In the scenario of 200x200 unit (shown in figure 7 over left), a robot equipped with 800 **eu** maximum has possibility to come back because it is in the energy potential itself. However, some robots die when the scenario is enlarged. The death sometimes happens once robots far way from charging station do not have sufficient energy to return to the charging station. Death rate reduction of robots when their energy is expired while working far way from the charging station was partly discussed in the modelling and will be clarified further next section.

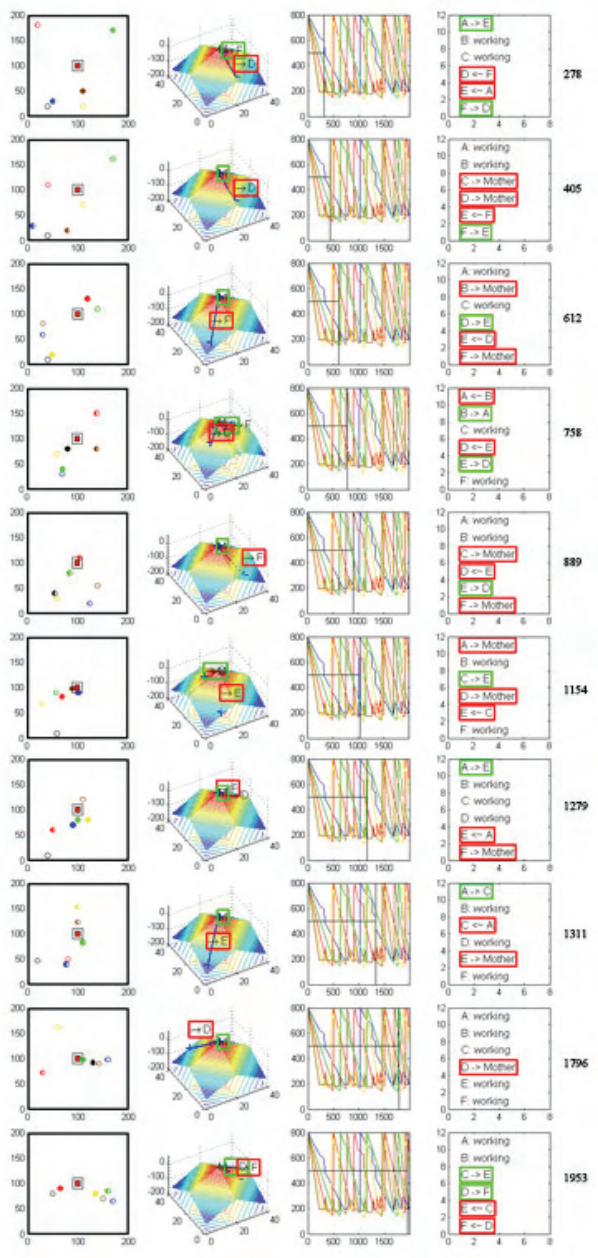


Figure 8. Snapshots of simulation on time scale

	A	B	C	D	E
Back to CS	**3**	**5**	**5**	**6**	**7**

Table 1. Simulated result of 5 robots in 2000 steps: Self-refueling Energy

	A	B	C	D	E
Survival time	5.4	4.2	1.1	3.4	**5.4**
Sharing Battery	0	0	1	0	**0**
Receiving Battery	**0**	**0**	**0**	**0**	**1**

Table 2. Simulated result of 5 robots in 2000 steps: Self-sharing Energy

	A	B	C	D	E	F
Back to CS	2	3	4	5	6	**8**
Sharing	1	3	0	1	0	**2**
Receiving	**0**	**0**	**2**	**1**	**2**	**2**

Table 3. Experiment of 6 robots in 2000 steps: Combination

	A	B	C	D	E	F
Back to CS	3	4	5	7	10	**13**
Sharing	2	1	2	3	3	**3**
Receiving	**1**	**1**	**3**	**3**	**2**	**4**

Table 4. Experiment of 6 robots in 4000 steps: Combination

3.3.2 Food Exchange: Self-refueling Energy & Self-sharing Energy

Observed from of the experiments of self-refueling energy described in the last section insufficient energy to return to the charging station causes robots to die. This enables robots to limit their activity domain since the further the robot is away from the charging station the hard the robot is able to survive. On the other hand, if every robot must go back to the charging station when energy is exhausted, they spend too much time and energy by travelling back. Moreover, density of traffic is accordingly increased causing traffic jam on the road as well as at the charging station, leading to prevention of energy refuelling.

To compensate drawbacks of self-refuelling energy, a solution of self-sharing energy is proposed. The solution enables each robot to become a mobile charging station to rescue other robots through battery exchange.

To test capability of self-sharing energy, we apply the policy of self-sharing energy only to 5 robots with different energy cost functions and deploy them randomly in the scenario without a charging station. From simulation result, we have had a statistic number of energy exchange and survival time as shown in table 5. It is observed that energy cost function of robot C is less than the one of robot A or robot B, but robot C dies earlier than robot A or robot B since it has shared energy with robot E. Thanks to energy aid of C, E survives longer than the other while its energy cost function is the most heavy. Although robot E is energetically rescued to prolong the life, no robot survives after an interval since external power resource is not provided. The example illustrates that the capability of self-sharing energy is aware as a short-term energy while the capability of self-refueling energy is understood as a long-term energy.

A combination of short-term and long-term energy solution enables robots to prolong their life, save energy and time of traveling back to charging station, avoid traffic on the road and

collision at the charging station. An example of 6 robots with both capabilities is examined to evaluate survivability of mobile robots.
For the first trial, we executed the simulation in 2000 steps illustrated in figure 8. Astonishingly, there is no dead robot in 2000 steps. In fact, robots all use short-term and long-term energy to support or refuel energy cooperatively. A statistical table of simulated results can be seen in table 3. Given example of robot D, at instant 278, D is shared energy by F, and then D has enough energy to go back the charging station to refuel energy at instant 405. It is surprising that D turns into a mobile charging station at instant 612 when it is going to share energy to E. But, after energy of E is refreshed at the charging station, E is going to share energy to D again at instant 758 and 889. Likewise, D still survives at step 1796 and will go back to the charging station. About 200 steps latter, D with full energy capacity is going to rescue E and so on.
Similarly, the simulation was executed 4000 steps again. Simulated results in table 4 demonstrate that the combination of self-sharing energy and self-refueling energy is a novel promising solution for groups of mobile robots towards energy autonomy.

4. Hardware Design for Energy Trophallactic Robot

4.1 An Introduction

The term "autonomous robot" is nowadays widely used to define robotic systems which function without human intervention. In fact, people have attempted to build systems which could operate without human control. However, the term "autonomy" is rather hard to assess due to the policies of inventors, which are leading to ambiguous meanings. In our opinion, a fully autonomous robot is a robot that must poses two qualities: *behavioral autonomy* and *energetic autonomy*. Behavioral autonomy can be defined as the ability to determine and execute actions which could be affected by the obtainment of energy. Energetic autonomy can be seen as the ability to maintain its energy to prolong its lifetime. However this behavior can be used to yield energy in the case of an energy self-rechargeable robot. To date, the term "autonomy" in robotics has mostly been used in the sense of "behavioral autonomy" only, not including "energetic autonomy". The example given is an intelligent battery-operated robot that can carry out a task without human intervention. e.g., iRobot Roomba vacuuming robot[3]. However, when working on an assigned task, the energy of the robots must previously be estimated to complete the task over some predefined period. In this period, the behavior of the robot may be considered as an autonomous operation. Otherwise, without human assistance to complete a job, the robot must autonomously return to a charging station, if possible, to refuel when the battery becomes low. In short, the behavior of the robot is always under the constraint of energy.
In the longer perspective, we are interested in large populations of mobile robots in which a robot is a truly autonomous agent. However, to achieve full autonomy, the robot must demonstrate the ability of energy trophallaxis obtained from two functionalities: self-recharging energy and self-distributing energy, accompanying the ability of behavioral autonomy. Conversely, energy trophallaxis powers essential elements of the behavior, including sensing, motion, and computation in order to maintain the robots' action.
In particular, we have developed new robots, named CISSbot, that can perform not only energetic autonomy, but also behavioral autonomy, concurrently. The realization of the

[3] See www.irobot.com

robot is on the one hand to redefine the definition of "autonomy" in robotics, and on the other hand, the unique design can suggest a new method to create truly autonomous robots in large populations.

4.2 Related Work

In the section, parts of related work of behavioral autonomy and energetic autonomy will be addressed. On the one hand, the overview is to emphasize the importance of energy trophallaxis in large robot populations since there does not yet exist a truly autonomous robotic system. On the other hand, we will explain the technical details that we have taken into consideration to design our robots. From this viewpoint, we have chosen some types of robots as examples since their designs bear the closest similarity to our design specification.

For behavioral autonomy, it is easy to search for a list in terms of self-reconfigurable robots and behaviorally autonomous robots. The former is defined as a machine built from several identical modules, and by reforming their module connections to change the shape and functionality of the entire machines or organism. The M-TRAN [Murata et al, 2000],[Yoshida et al, 2001] and ATRON [Lund at al, 2005][Ostergaard et al, 2004] reconfigurable robots are typical representatives of the class. The M-TRAN uses magnets to attach surfaces of modules together and Shape Memory Alloys (SMAs) to detach those connections. The ATRON sets up a point-to-point physical connection by using the hooks of the active connector to grab into the passive connector. M-TRAN modules are battery-powered units but they are not able to share or self-recharge energy automatically, while the ATRON modules are also battery-powered units that are able to share energy by inter-connections. However, M-TRAN and ATRON both still meet the limit of energetic autonomy, and thus they are only self-reconfigurable in morphology, but not self-rechargeable. The second requirement of behaviorally autonomous robots is an autonomous robotic system operating autonomously without human intervention in behavior. Although this exists in many single robots, and in cooperative or coordinating robots in a wide range of applications for underwater, ground, or aerial environments, we have chosen Swarm-bots [Groß et al, 2006][Mondada et al, 2005] as a good example since the robotic system covers numerous features similar to, or more advanced than, the other systems. Swarm-bots (s-bots) is a European project to create self-assembling and self-organizing robots inspired by from social insects or other animal societies. Unlike M-TRAN or ATRON, s-bot is a completely independent mobile robot, that is, s-bot is able to move without the need to inter-connect with other s-bots. For capabilities of self-assembling and self-organizing, s-bots are equipped with a flexible gripper that is capable of grasping another s-bot to lift it when it needs to pass over large holes or pass through narrow passages in complex environments. They also have two short rigid grippers to provide easy connectivity to the other robots. In combination with two kinds of gripper, several types of sensors are used to play very different roles in swarm-robot configurations, e.g., infrared proximity sensors, infrared ground proximity sensors, color sensors, inclinometer sensors, humidity and temperature sensors, etc. In fact, Swarm-bots are a type of battery-powered robot. Unfortunately, the swarm-bot has no capability for sharing energy among robots, or refueling its energy by exchanging batteries with a charging station, eventually they would be able to perform such capabilities thanks to very high processing power, excellent sensing capability, and smart grippers. In short M-TRAN, ATRON, and Swarm-bots are not yet truly autonomous robots.

However, features of physical connections, neighbour communications, and processing power are very useful examples when designing the CISSbots.
For energetic autonomy, there exist only a few robots that are able to act as energy rechargeable robots or self-power robots. A very good example of the first class is the vacuum cleaning robot of which Roomba and CleanMate are two typical representatives. Typically, the robots move around freely to clean carpets and automatically return to the docking station to recharge the battery when it is low. Although the robots have partly solved the problem of self-recharging energy, they still lack the capability of quickly recharging by exchanging batteries instead of waiting at the charger for a long time. Also, the robots can only work alone without capability of sharing energy among moving robots as our CISSbots do, so they can not act in large populations. A well-known robot of the second class is a series of ecological robots named EcoBot. These robots are referred to as a class of energetically autonomous robots that are self-sustainable by collecting their energy from waste in the environment. EcoBot-I [Ieropoulos et al, 2004] was developed to utilize sugar as the fuel by onboard Microbial Fuel Cells (MFC), while EcoBot-II [Ieropoulos et al, 2005] was created to consume dead flies, rotten fruit, and crustancean shells. The application domain of EcoBot is places which are very hard to access by humans, e.g., under water or in a poisoned area. In fact, the EcoBot series has partly overcome the problem of self-powering, but they can only work in a waste environment where biomass is available. Moreover, like the vacuum clearing robots, the EcoBot usually acts alone, so it can not be able to benefit from colony activities.

4.3 The CISSBOT Design Idea

As a part of our research into sociable robots, we are currently developing a prototype of a truly autonomous robot, called CISSbot (shown in figure 9). The CISSbot is a differential two wheeled robot with a special mechanism of battery exchange. The architecture of the CISSbot is divided into two main parts: the lower layer and the upper layer. The lower layer is the behavioral control to perform the behavioral autonomy while the upper layer supports battery exchange to refuel the robot's energy or to share energy with the others.
Like mobile robots working in 2D, the lower layer of the CISSbot also has a full specification of sensors, actuators and a central processor to control the robot's behavior automatically. However, because of our target to demonstrate energy distribution in large populations, we simplified the experimental scenario to a grid map. Thereby, the behavioral control enables the CISSbots to move freely on the grid map, meeting each other at a point, or moving back to a charging station.
However, unlike mobile robots seen previously, we have been developing a battery exchange mechanism for the CISSbot to load and unload batteries. The mechanism is designed in an H-shaped set of battery holding boxes that include the electrical contacts and the battery pushing systems at both sides. However, in order to make a decision about the battery exchange, the electronics of energy management have to be taken into consideration. More details are described in the next sections.
This design is a compromise between several mechanical, electronic and control considerations. To cause a robot to exchange batteries with the charging station, or to distribute batteries to other robots, the behavioral control and the battery exchange mechanism must collaborate in a synchronized mode.
In order to realize energy trophallactic robots in large populations, we require a robot to:

- sense the current state of its energy;
- give a decision about energy replenishment;
- search and talk with other robots;
- have neighbor to neighbor communication;
- be able to contact or bump the charging station or other robots precisely;
- have a mechanism to exchange the battery successfully.

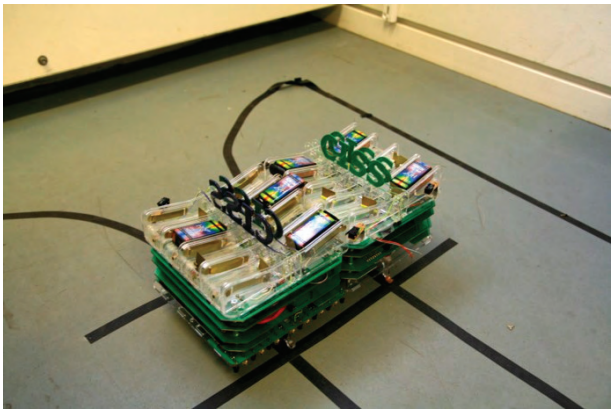


Figure 9. Two robots in a battery exchange process

4.4 Hardware Realization

The goal in designing the CISSbot is an experimental platform of an energy trophallactic robot, so we decided to develop the robot in layers and modularity. That is, the system and its functionality can be extended by just adding new layers onto the base layer, or plugging new modules onto the other electronic boards. The rest of the section will realize several aspects of the mechanical design in multilayer architecture, modular functions of electronics, and how to assemble parts into a compact platform.

4.4.1 Mechanical Design

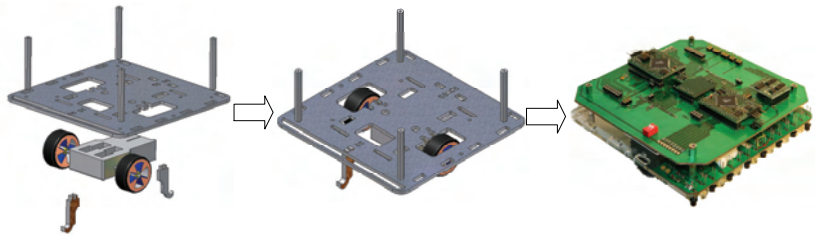


Figure 10. Mechanics of the behavioral control layer: CAD disassembled (left), CAD assembled (middle), completed mechatronics (right)

Mechanics of Behavioural Control

Towards an easy process of battery exchange, the CISSbot is physically formed in a 15 x 15cm square shape. Like most mobile robots, the chassis of the CISSbot is a base to hold the motion mechanism and electronic system to control the robot. The movement of the CISSbot

is driven by a differential two wheeled system and two castors to keep the robot balanced. Initially, we selected the gear-box of two DC motors from Tamiya toys, but the test showed that the gear-box can not be used since its power consumption is over 800 mA, while we are using 250 mA batteries to power the robot. Therefore, we have modified servomotors into continuous servo motors at a gear rate of 218:1 to drive the two wheels because such motors allow the CISSbot to move quickly in a narrow space, to turn slowly around its central point, and to approach easily to a target at a stable current of 100 mA approximately.
The mainboard to control the CISSbot's behavior is put on the top of the base. At two sides of the base chassis, an odometer of the two wheel encoders is mounted to observe the movement of the wheels. The base is made as compact and robust as possible, with the main emphasis on the low-cost production.

Mechanics of Battery Exchange

The mechanism of battery exchange is a complicated architecture. To protect against short-circuits, flexible plastic is the material chosen to compose the architecture. In particular, the skeleton of the battery exchange mechanism is assembled from several flexible plastic parts to form 8 battery boxes on the base. At the two sides of each box, there are two brass plates to physically conduct electricity from the batteries to the entire system. To hold a battery in the battery box, as well as to protect against mechanical shocks when the robot is moving, a plastic pad is tilted 12.5 deg over the central base (see figure 11). Each battery box involves a linear motion mechanism fixed at the two ends of the box in order to allow the mechanism to push the battery in the slide-way. As expected, initially, it was very hard to find a compact linear motor mechanism since there is no geared micro motor available. Fortunately, we found a geared micromotor, rate 25:1 with excellent specification from www.solarbotics.com. After being technically modified, the microlinear mechanism was made manually to ensure that it is powerful enough to force the battery out.

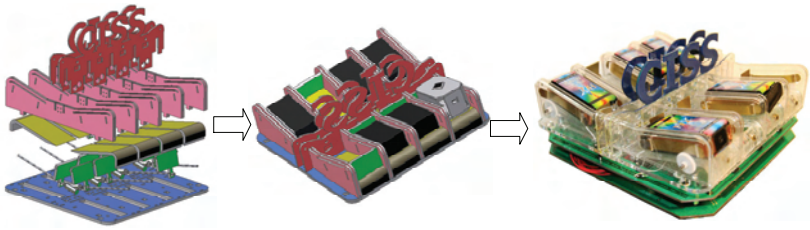


Figure 11. Mechanics of the battery exchange layer: CAD disassembled (left), CAD assembled (middle), completed mechatronics (right)

Under the base of the battery exchange, the entire battery management, control of 8 battery boxes and local communication for the battery exchange process is physically hung (see figure 10, right). In the future, we will redesign the battery layer to hold a laser-sensing system on the top for the robot's further sensing capability.

4.4.2 Modular Electronics

Power Management

The goal of the CISSbot is to lead to a new methodology for a truly autonomous robot, so power management is the key to our considerations. The main requirements of the power management system are to continuously sense the current state of the batteries carried by

the robots, protect against occurrence of the short-circuits, regulate the power of the source of usable energy for the robot's actions, and select the number of batteries in use.
As shown in figure 12, because the batteries used are of rectangular parallelepiped form with two brass connectors modified at each side, the battery exchange process could randomly lead to the wrong sign connection when the battery is shifted another robot or from the charging station. To solve this problem, we added a two way rectifier to ensure the right polarity, no matter how the connector side of the battery is electrically conducted on the electronics board of the power management system.
In moving towards a smart energy management software program for the CISSbot, the input data of the current state of the battery is very important, and inquiries about the updating speed, the current and the reliability of the measured data are strongly emphasized. However, depending on the chemical characteristics of the rechargeable batteries, e.g., Li-Po, Li-Ion, Ni-CD, Ni-MH, etc., their capacity is changed differentially, and thus their discharging functions are not identical. This makes battery monitoring more difficult and less precise if only the voltage of the battery is measured, which is what many battery monitors have done previously. Moreover, load of the system powered by the battery probably affects the measurements, and the limit of the deliverable current. The question is, how to monitor the current state of different types of the battery precisely in real-time. To improve the accuracy of energy management, the methodology of monitoring both the current and the voltage of the battery has been selected. Fortunately, our method has also been the considered by Maxim electronics offering a commercially available smart battery monitor chip. We needed to connect only one wire of the chip to our processor in order to evenly monitor the elements: voltage, current, and temperature.

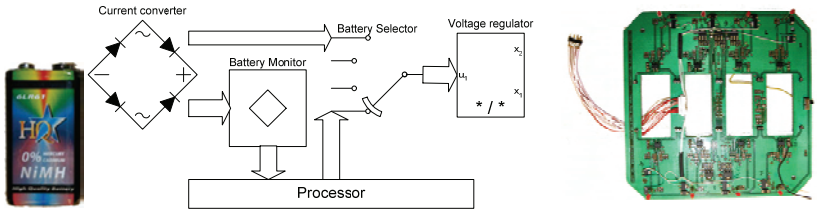


Figure 12. Power management: Block diagram (left), completed PCB (right)

The voltage regulator of the power source is also very significant for the modular electronics of the CISSbot since there are many different modules powered by a single source. Because the CISSbot's electronic modules are operating in a large voltage range from 3.0 V to 9.0 V, a series of voltage regulators has been implemented. Unusually, we had to specially select two high-current regulators for the two drive motors of the wheels owning to the heavy load.
To make the power management more flexible and powerful, an electronic circuit of battery selection controlled by the processor was added on. The selector allows the power management system to decide the number of batteries in use with respect to the power consumption of the CISSbot in an execution mode, as well as the current state of the batteries on the energy base. For example, in free motion mode, the CISSbot needs to be powered by two fully charged batteries, but there are currently only five partly used batteries, which might be exchanged by the other robots on the base so that four of those are

fully activated at once. This helps the CISSbot to use the remaining energy from all batteries used.

The elements of power management described above is entirely mounted on a single printed circuit board (PCB) that is held under the mechanics of the battery exchange layer (see figure 12)

Processing Power

Since the CISSbot architecture is structured on two layers, the behavioural control layer and the battery exchange layer, we decided that each layer should be powered by a processing unit instead of having a central unit for the entire system. This division aims to increase computational power in each unit, as well as to reduce unnecessary signals passing through two processing units. If the processing power was put in a single unit, several electrical signals would have to be processed, filtered, and transferred to related devices simultaneously, leading to overloading of the processing unit. In addition, information propagated between two processing units must be preprocessed and filtered to avoid a traffic-jam on the common bus, leading to transmission delay.

As illustrated in figure 13, the behavioural control layer is controlled by an ATMega128 microcontroller, named the main processor, and the battery exchange layer is driven by an identical micro-controller, named the secondary processor. ATMega128 from Atmel is a 8-bit micro-controller with 128 Kb flash memory, 4 Kb RAM and 4Kb EEPROM. The microcontroller is set up to run at 14.7456 MHz.

The main processor is on the one hand responsible for several sensing devices to control the CISSbot's behaviour: an infrared array (line-following sensing), a compass (orientation sensing), an odometer with two wheels (distance estimation), and wireless communication (global signal propagation). On the other hand, the processing unit also drives the two differential motors of the motion mechanism.

The secondary processor is mainly responsible for monitoring the state of the batteries. In addition, the local infrared communication (neighbour communication) of the battery exchange process and the control of the microlinear motors to push the batteries are also managed by this processor. Assisted by the main processor to control the CISSbot's behaviour, proximity based on the same local infrared sensors can be detected by the secondary processor in order to build a local map. In association, the two processors are able to intercommunicate on a TWI interface based on I²C protocol.

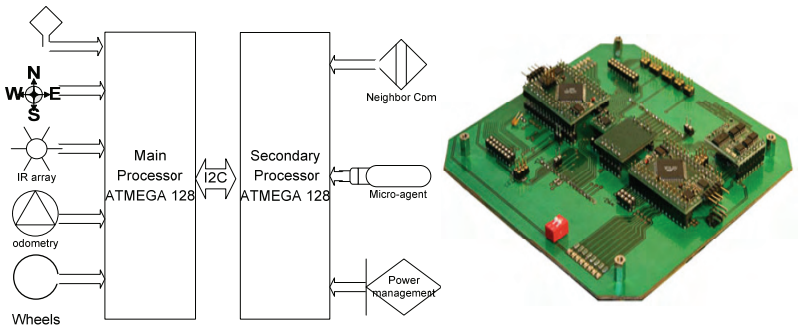


Figure 13. Processing power: block diagram (left), completed board (right)

Sensor

Line following sensor:

As the CISSbot is an experimental platform of a truly autonomous robot, we set up a grid map scenario of white crossing lines on a black carpet. To approach quickly to the goal of battery exchange, a highly accurate infrared array has been made to control the motion mechanism of the CISSbot adaptively. However, unlike most of the infrared arrays available on the market, we applied high computational power using the Lagrange interpolating polynomial in order to find the distance measured from a white line to the central point of the robots precisely. The measured values are converted to numerical values by the on-board ADC port of the microcontroller. The infrared array is also able to calculate the varying distance from the center to 10^{-2} mm accuracy. The numerical value of the varying distance is used to apply the proportional – integral – differential (PID) controller for CISSbot motion.

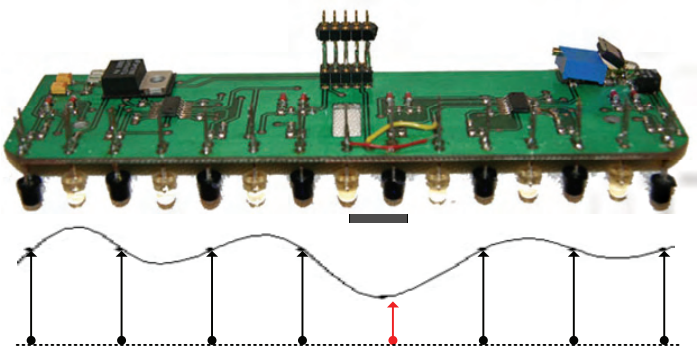


Figure 14.. Top. Line following infrared array. Bottom: Correlative Lagrange interpolating polynomial

Orientation sensor:

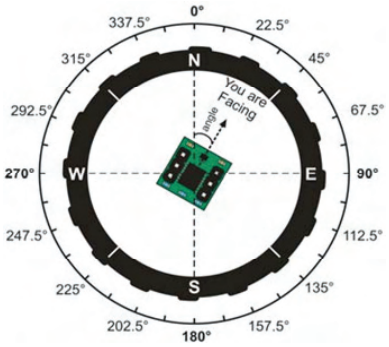


Figure 15. Digital compass

In addition to sensing information, a CISSbot is equipped with a digital compass: a dual axis magnetic filed sensor built round the Hitachi HM55B IC. The external device is used to determine the orientation of the robot upon the Earth's magnetic field with sensitivity up to

microtesla. This information is essential to calculate the related orientation among the moving robots in order to compensate for the inadequacy of their orientational sensory information. In particular, it is very useful for the battery exchange process when two robots are in preparing to exchange batteries. Although we are currently using the compass to sense bidirectional axes only, it will be more useful in the future, with high sensitivity, to guide robots approaching each other when we remove the grid map.

Odometer sensor:

For further sensing, a CISSbot is also set up with a couple of incremental encoders for both wheels. By using Hamashuta (very short-range infrared), the encoder will increasingly accumulate the travelling distance by observing a barcode, which is fixed on the wheels, when the wheels are rotating (see figure 16). In the current design, the sensor is able to sense 2 mm a long the distance of the robot's movement. The distance information captured by the width of on-off pulses by the microcontroller is used to guide one robot to approach another one for battery exchange very precisely, or to estimate energy consumption, and to propose a schedule for energy management.

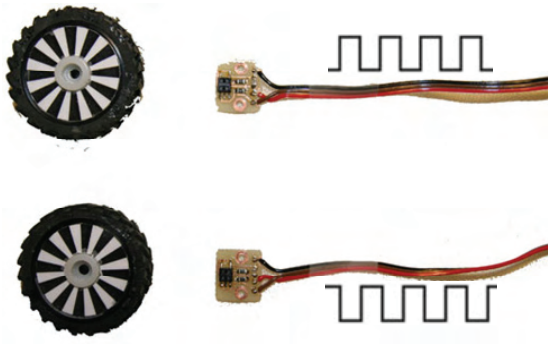


Figure 16. Odometer: barcode (left), short-range infrared sensors (right)

Proximity distance sensor:

To perceive the surrounding environment, distance sensors are indispensable. However, in a large population of homogeneous mobile robots where interference with signals always happens, long-range sensors are not absolutely appropriate, and primitive short-range sensors are better. Further, the signal interference is reduced if a high-speed sensor is chosen. Based on the required specifications, infrared is the best choice in this case. At the each side of the CISSbot, four pairs of proximity infrared sensors are disposed as illustrated in figure 17. The current design allows sensing up to 16 cm in front and on both two sides of the CISSbot approximately. The purpose of this sensing is to report if the robot meets obstacles, or recognizes other robots.

Neighbourhood Communication

In order to have a successful battery exchange process, every robot must be able to communicate with its neighbouring robot. To obtain this communication, each side of the CISSbot has four sets of infrared diodes and phototransistors, corresponding to the location of the four battery boxes on the sensor board. A diode for transmitting messages and a phototransistor for receiving data are placed according to the center of each battery box. With the infrared couple placed in this pattern a robot can communicate with a neighbouring robot if they are closely approaching side by side. Based on the number of

infrared couples in communication, the two robots negotiate to decide in which battery box of the sharing robot the battery is handed on, and which battery box receives such a battery on the receiving robot.

For the possibility of high-speed communication, we initially designed the communication with four infrared couples transmitting data from one robot at the same time, and then the other robot should be able to receive four independent signals at any given time. These couples all were directly connected to the processor in order to be driven by four identical set of UART software. Unfortunately, the test showed that the speed for a single channel would be always less than 735 bps. Thus, one robot must take a longer time to negotiate with another one in order to complete the battery exchange process. However, the need for high-speed neighbour communication to avoid traffic-jams in large populations is highly urgent, so $(\sum_{1}^{n} +1)$[4] times 730bps is not a promising result. We changed our minds, and used the built-in UART of the processor in combination with a multiplexer. In fact, the four pairs of infrared sensors were multiplexed by an analog multiplexer, shown in figure 17. Although use of the multiplexer for neighbour communication could limit the number of communications to a neighbour at the same time, it increases the transmission bandwidth to 9.6 kbps. As a result, neighbour communication is raised to 2.4 kbps approximately.

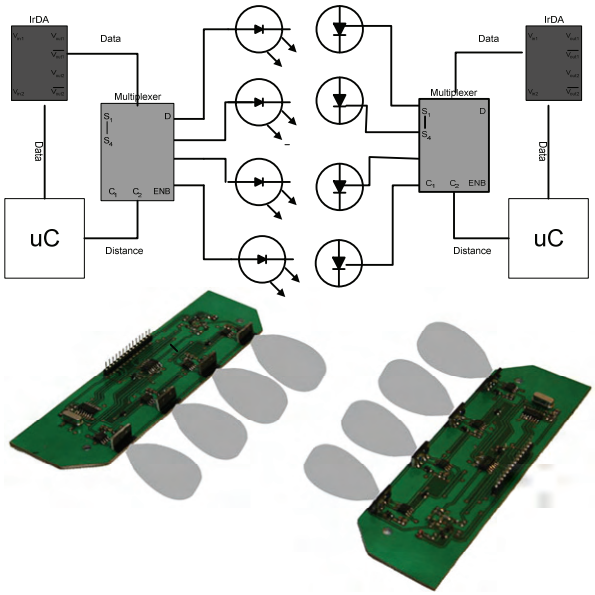


Figure 17. Neighbor Communication: block diagram (Top), completed boards (Bottom)

To ensure reliable propagation of data between two neighbors, an IrDA physical layer is placed between the UART and the infrared sensors for encoding and error checking. The data received from the standard UART is encoded, and output as electrical pulses to an

[4]n is the number of neighbor communication units in communication, maximum 4. 1 plus is to send a command of starting the battery exchange process.

infrared diode transmitter. If infrared phototransistor receivers also capture data which are output as electrical pulses, the physical layer demodulates these electrical pulses and transmits to the UART of the processor. Because the modulation and demodulation method is performed according to IrDA standard specifications, it allows a user to interact with the CISSbot using a portable device, e.g., a cellular phone, a laptop, or any other device with an IrDA interface.

Global Communication

As described above, neighbor communication can assist a robot to communicate with other robots at very short range only. Nevertheless, the CISSbot is a energy trophallactic robot in large populations, and thus global communication at long range is certainly required to search for and talk with other robots. In this request, wireless communication is the best choice for low-cost, wide range, and ultra-low power consumption. Initially, we selected CC1000 from Chipcon for our robots. However, a highly complicated circuit board is needed for this device, and its frequency is not in the EU licensed range. Hence, we have finally chosen CYWUSB6935 from Cypress for the global communication. The wireless chip is easily integrated into the processor, with few extra components. This utilizes the worldwide unlicensed 2.4 GHz ISM frequency band and uses a robust Direct Sequence Spread Spectrum (DSSS) transfer method with a data rate of 62.5 kbps and a 50 m range. The low standby power consumption of <1μA is ideally suited for battery-powered robots. Fortunately, the wireless board is available from the Chip45[5] company.

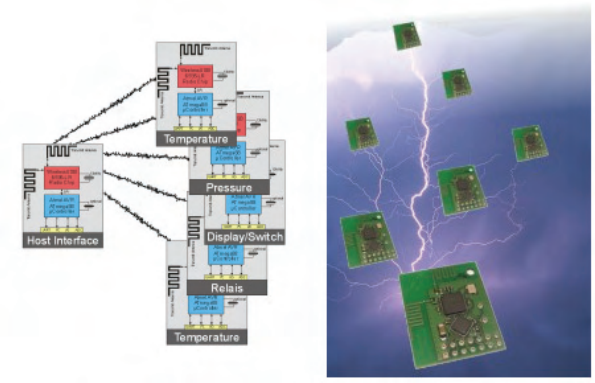


Figure 18. Global communication: (N:1) Wireless Network

To ensue reliability in bidirectional communication between one CISSbot and another, a set-up of wireless multipoint-to-point (N:1) networks was implemented. This protocol aims beyond the classical solution of point-to-point (1:1) wireless communication. Error-detection, correction, and automatic channel selection are additionally programmed into the processor. The propagation protocol guarantees that a robot will receive necessary information on the battery state, and commands about the communal jobs from other robots, as well as broadcast its own information to the others.

Driver of Behavioural Control

[5] The boards are bought from chip45.com. (N:1) protocol was initially a firmware of the website. However, this protocol is mostly modified in order to apply for our robots.

Basically, two DC motors of the motion mechanism are driven by Pulse Width Modulation (PWM) signal sent from the main processor. As the most popular control algorithm used to control a mobile robot is a PID controller, we also apply this method to the CISSbot with an external feedback of the varying distance of the infrared array and odometer information from the wheel. The power electronics used to control two wheeled motors are a dual H-bridge driven by a micro-controller. This allows the CISSbot's motion mechanism to be controlled by the UART of the main processor.

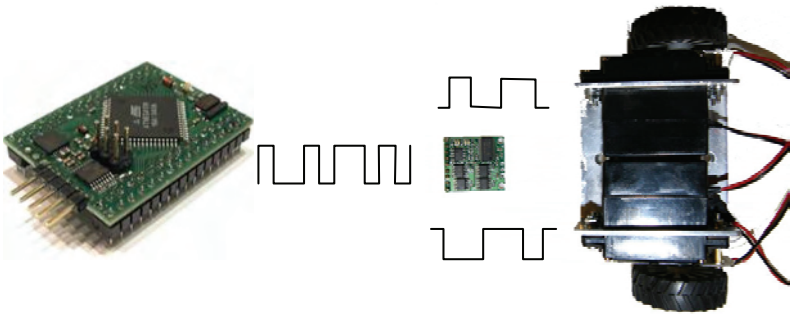


Figure 19. Driver of two wheeled motors

However, the CISSbot executes a lot of action modes, e.g., free mode, collision detection mode, job mode, battery exchange mode, and so forth, and a basic PID controller might not respond well to these differential behaviors. To adapt the control to each mode, the controller must be upgraded to an adaptive PID whose setpoint and gains are adaptively changed according to each mode. For example, the gain might be large in the free motion mode to allow the robot to run quickly, but it should be minimal value in the battery exchange mode to make the process successfully.

Driver of the Battery Exchange

The battery exchange mechanism is a set of eight linear DC micro-motors systems, manually constructed in a combination of linear micro-motors and screw thread rods. This allows the rotating force of the DC micromotor to transform the pushing force of the screw thead rods in order that a battery is physically pushed from one robot's battery box to the other if the two robots are completely alighted side by side. However, it is too wasteful to use eight H-bridges to drive eight motors, instead of multiplexing them in order that they can be controlled by a single H-bridge motor controller. The solution confronted the barrier of the capability of a multiplexer to deliver a high current to the DC micromotors. Referring to the specifications of the micromotor (3 V, 128 mA), no fully suitable multiplexer is available on the market. Therefore, to overcome the problem of high current conductivity, we used a digital multiplexer to switch the mode of the micro-motor. However, the current to control the micromotors goes through high power FET transistors under control of the H-bridge. Because we used a single H-bridge, a UART and 3 address lines of the secondary processor are needed to drive the pushing mechanism of the eight battery boxes. In fact, a battery exchange process takes 5.7 s to perform, and it consumes 0.7 joule to deliver a force of 80 N approximately, when the micromotor is powered by 3.3 V at a current of 120 mA.

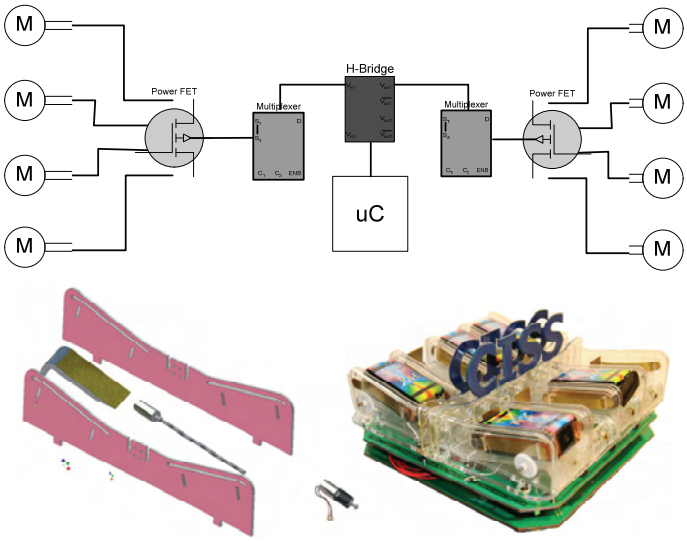


Figure 20. Battery pushing system: block diagram (Top), diassembled mechanical part (bottom left), geared micro-motor (bottom middle), completed system (bottom right)

4.4 Drawbacks and Improvements

The CISSbot hardware has now been implemented. Although the design is promising and it is able to function as we expect, there are some drawbacks in the technical details which we wish to improve in the future.

Mechanical stability:

To establish experiments quickly, plastics may feature as flexible materials. Although the CISSbot is assembled with several plastic pieces, the mechanical properties have not shown any unexpected errors to date. However, so far, there have been a few simple established experiments so we may need to conduct more complicated experiments to investigate the mechanical stability. In the near future, we may change plastics to aluminum materials to extend more functions to the CISSbot, e.g. a self-forming chain of robots, attaching each other to exchange batteries.

Electronics functions:

The compass provided unreliable angle measurement due to motion friction when the robot is moving. Actually, the compass acts as a passive sensor, and thus the magnetic field generated by two wheeled DC motors may effect it. Fortunately, we are currently using the sensor to sense only four bidirectional axes so that this is not serious problem. But it should be considered when the pre-built map is removed.

In the proximity mode, the infrared signal may interfere if two robots are too close. Moreover, the infrared signal sometimes jumps to an unexpected infrared receiver. As predicted, this may be a problem of signal strength, so we must pay attention to solving these shortcomings in the near future.

Control complexity:

As explained, the main goal of the CISSbot is to create truly autonomous robots through self-distributed energy. This is done by two methods, either coming back to a charging

station to refuel energy or exchanging energy with the other robots. However, these methods emphasize the techniques of motion planning in large populations. Given this assumption, the execution phases of a battery exchange process are a set of states (invisible state, visible state, close state, etc.), and every state must be successfully solved in order to ensure that the set of states ends completely. As a result, the success of the set leads to the success of a battery exchange. Although the current design obtains the full basic elements to control a battery exchange process between two CISSbots on a pre-built map, the new control algorithm still has to be improved in order to increase the percentage of successful battery exchange processes and save travel time.

Neighbor communications:

To perform a battery exchange process, a robot must first be driven to approach another robot side by side. Second, the two robots will have to use neighbor communication to negotiate with each other to find out the right battery boxes to be exchanged. Normally, a number of $\sum_{1}^{n}$ neighbor communications has to be carried out to find the right battery boxes, and one more talk is needed to complete the process. However, wasteful communication time should not happen in large populations. Therefore, we have to pay more attention to this aspect in the future. A new method of battery exchange through direct attachment will soon be implemented to reduce, or even remove the number of neighbor communications soon.

General improvements:

The development of the CISSbots is long whiled with several constrained circumstances. For example, the battery choice would be able to lead to the size of the battery box, and the size of the battery box might decide the size of the robots and so on. Further, we might find it hard to produce aluminum pieces to make the robot, and thus the robot would essentially be assembled from numerous plastic pieces, which can make the robot slightly unstable if some hard tasks are required. The battery pushing mechanism could give us a trouble since there are many opinions about using a micro pneumatic air pump, shape memory alloys, or static-magnetic force but none have proved feasible after experimentation. Finally, the linear micromotor mechanism is an acceptable solution, although it is not perfect, due to some home-made parts.

In the near future, we have to pay more attention to interference in both neighbor communications and global broadcasting. This does not always happen, but it has sometimes made our robots to behave incorrectly. More investigation onto distance sensing capabilities, based on a smart beacon, would be preferable in order to remove the current grid-map.

5. Conclusion

We presented our study of randomized robot trophallaxis through the main aspects of modeling, simulation and real implementation. The modeling is kept at an abstract level in order to provide general analytical results of intergro-differential equation governing the evolution of expected energy levels and variance as function of position velocity and time. The model is partly used to evaluate the performance associated to embedded environments such as the position and density of charging station and robots. Ultimately, survivability is the final target of the developed model. Besides, the concept “randomized robot trophallaxis” is further clarified through simulation results. The developed simulation

showed that a group of mobile robots is able to be truly autonomous if and only if they are capable of "trophallaxis" containing self-refueling energy and self-sharing energy. To shed light on "randomized", several simulations were setup to examine in terms of density of robots, control algorithms, task allocation, and location of charging station. Finally, to increase the reality of our study, we consequently presented hardware implementation of the trophallatic robot. Details of mechanical architecture and modular electronics were described. Based on the implementation, we also pointed out the advantage and drawbacks of the current design and direction for future work.

6. Reference

Bettstetter. C, Hartenstein. H, and Perez-Costa.X (2002). Stochastic Properties of the Random Waypoint Mobility Model: Epoch Length, Direction Distribution, and Cell Change Rate. In *Proc. ACM Intern. Workshop on Modeling, Analysis, and Simulation of Wireless and Mobile Systems* (MSWiM), Atlanta, GA, USA, pp. 7-14, Sept 2002.

Le Boudec, Jean-Yves ; Vojnovic, Milan (2005). Perfect Simulation and Stationarity of a Class of Mobility Models, *IEEE INFOCOM* , Miami, 2005.

Øksendal, Bernt K. (2003). Stochastic Differential Equations: *An Introduction with Applications*. Springer, Berlin. ISBN 3-540-04758-1.

Camp. T, Boleng. J, and Davies. D, (2002) A survey of mobility models for ad hoc network research,*Wireless Communications and Mobile Computing,* vol. 2, no. 5, pp. 483–502, 2002.

Schiøler. S ; Martin Bøgsted. H.P. Schwefel (2005). Probabilistic Modelling of Information Propagation in Wireless Mobile Ad-Hoc Network. In *Proceedings of the International Symposium on Wireless Personal Multimedia Communications* (WPMC 2005). Aalborg : Center for TeleInFrastruktur (CTIF), Aalborg Universitet, 2005.

Medlock.J, Kot. K (2003). Spreading disease: Integro-differential equations old and new, *Mathematical Biosciences*, 184(2):201–222, 2003.

Moreno. Y, Nekovee. M, and Pacheco A.F. (2004) Dynamics of rumor spreading in complex networks, *Phys. Rev. E. 69*, 066130.

Camazine. S (1998), Protein Trophallaxis and the Regulation of Pollen Foraging by Honeybees, *Apidologie*, Vol.29, pp.113-126, 1998.

Sumpter. D.J.T and Beekman. M (2003), From Nonlinearity to Optimality: Pheromone Trail Forageing by Ants, *Animal Behaviour*, Vol. 66, pp.273-280.2003.

Payton. D, Estkowski. R, and Howard. H (2005), Robotics and The Logic of Virtual Pheromones, in *Swarm robotics, SAB 2004*. LNCS 3342, Springer Verlag, 2005, pp.45-57, 2005.

Ngo. T.D, Schiøler. H (2006), An approach to sociable robots through self-distributed energy, *In proceedings of the International Conferences on Intelligent Robots and Systems (IROS)",* Beijing, 2006.

Ieropoulos. I, Melhuish. C, Greenman. J (2004), Energetically autonomous robots, in *Proceedings of the 8th Intelligent Autonomous System Conferences (IAS-8)*, Amsterdam, The Netherlands, pages 128-135, 2004.

Ngo. T.D, Raposo. H, H. Schiøler (2007), Multi-agent robotics: towards energy autonomy, in *Proceedings of International Conference in Artificial Life and Robotics (AROB'12th)*, Beppu, Oita, Japan, 2007.

Ieropoulos. I, Melhuish. C, Greenman. J, Horsfield. I (2005), EcoBotII: An artificial agent with a natural metabolish, *International Journal of Advanced Robotic Systems*, Vol2, 2005.

Mei. Y, Lu. Y.H, Charlie Hu. Y, and George Lee C. S. (2006a), Deployment of Mobile Robots with Energy and Timing Constraints, *IEEE Transactions on Robotics*, 2006.

Mei. Y, Lu. Y.H, Charlie Hu. Y, and George Lee C. S. (2006b), Energy-Efficient Mobile Robot Exploration, *IEEE International Conference on Robotics and Automation*, USA, 2006.

Murata. H, Yoshida. E, Tomita. K, Kurokawa. H, Kamimura. A, and Kokaji. S (2000), Hardware design for modular robotic system, *In proceedings of the International Conferences on Intelligent Robots and Systems (IROS)*, pages 2210-22127, 2000

Lund. H. H., Beck. H. H., Dalgaard. L, Self-Reconfigurable Robots with ATRON Modules (2005), In *Proceedings of 3rd International Symposium on Autonomous Minirobots for Research and Edutainment (AMiRE 2005)*, Springer-Verlag, Fukui, 2005.

Yoshida. E, Murata. H, Mamimura. A, Tomita. K, Kurokawa. H, and Kokaji. S (2001), A motion planning method for a self-reconfigurable modular robot, In *proceedings of the International Conferences on Intelligent Robots and Systems (IROS),* pages 1049-1054, 2001.

Groß. R, Bonani. M., Mondada F., Dorigo M. In K. Murase, K. Sekiyama, N. Kubota, T. Naniwa (2005), and J. Sitte, editors, Autonomous Self-assembly in a Swarm-bot, Proc. of the 3rd Int. Symp. on Autonomous Minirobots for Research and Edutainment, AMiRE 2005, pages 314-322. Springer Verlag, Berlin, 2006.

Ostergaard. E. H and Lund. H.H, (2004), Distributed Cluster Walk for the ATRON Self-Reconfigurable Robot, In *Proceedings of the The 8th Conference on Intelligent Autonomous Systems (IAS-8)*, Amsterdam, Holland, Pages 291-298, March 10-13, 2004

Dorigo M., Tuci E., Trianni V., Groß R., Nouyan S., Ampatzis C., Labella T.H., O'Grady R., Bonani M., Mondada F (2006), SWARM-BOT: Design and Implementation of Colonies of Self-assembling Robots, *Computational Intelligence: Principles and Practice, Gary Y. Yen and David B. Fogel (eds.), IEEE Computational Intelligence Society*, NY, 2006, 103-135.

Mondada F., Gambardella L.M., Floreano D., Nolfi S., Deneubourg J.-L., Dorigo M, (2005) SWARM-BOTS: Physical Interactions in Collective Robotics, *IEEE Robotics & Automation Magazine*, vol. 12, june 2005, pp. 21-28.

O'Grady R., Groß R., Mondada F., Bonani M., Dorigo M. (2004) In M.S. Capcarrere and A.A. Freitas and P.J. Bentley and C.G. Johnson and J. Timmis, editors, Self-assembly on demand in a group of physical autonomous mobile robots navigating rough terrain, *Advances in Artificial Life, Proceedings of the 8th European Conference on Artificial Life*, pages 272-281, Springer-Verlag.

12

Formation Control for Non-Holonomic Mobile Robots: A Hybrid Approach

Juan Marcos Toibero[1], Flavio Roberti[1], Ricardo Carelli[1] and Paolo Fiorini[2]
[1]Universidad Nacional de San Juan, INAUT, [2]Università di Verona, ALTAIR
[1]Argentina, [2] Italy

1. Introduction

Many cooperative tasks in real world environments, such as exploring, surveillance, search and rescue, transporting large objects and capturing a prey, need the robots to maintain some desired formations when moving. Formation control refers to the problem of controlling the relative position and orientations of robots in a group, while allowing the group to move as a whole. Problems in formation control that have been investigated include assignment of feasible formations, moving into formation, maintenance of formation shape (Desai et al., 2001) and switching between formations (Desai et al., 1999; Fierro et al., 2002). The work in (Das et al., 2002) is a very good example of the state of the art in robot formation control, in which it is presented a complete framework to achieve a stable formation for car-like and unicycle-like mobile robots. A feasible solution to address these problems is by using hybrid control systems in formation control. In fact, several papers can be found in the literature using hybrid control systems: including a discrete event system at the supervisory level and continuous controllers to give the control actions (Desai et al., 1999; Ogren & Leonard, 2003; Chio & Tarn, 2003; Ogren, 2004; Shao et al., 2005).
In this chapter, a hybrid approach for the autonomous navigation of a mobile robots team in a specified formation is developed considering a centralized leader-follower controller (Gava et al., 2007) (see for example (Shao et al., 2005) for a review on the leader-following method) and the non-holonomic constraint of the unicycle-like mobile robots (Gulec & Unel, 2005). In this last paper, the authors state that a complicated coordinated task can be interpreted in terms of simpler coordinate tasks that are to be manipulated sequentially. The leader robot of the team, which navigate independently according to its own control laws, has a laser range-finder, odometry sensors and an omnidirectional camera, whereas the followers have odometry and collision (sonar) sensors. The laser range-finder and odometry sensors of the leader robot are used to implement the leader robot controller (Toibero et al., 2007); and the omnidirectional camera is used to identify the follower postures relative to the leader coordinate system needed in the implementation of the centralized formation controller (Gava et al., 2007). The existence of such a relative sensor is not a constraint since the leader could get access to these positions using another absolute position sensor such as, for instance, a GPSs or odometry and then convert them to the framework attached to the leader robot. In addition, the centralized control architecture, where the control actions for all the followers are generated by the leader, could be decentralized by allowing the

followers to estimate the leader movements (angular and translational velocities) and performing a minimal communication between the robots (Fredslund & Mataric, 2002). Therefore, a decentralized control scheme could also be supported by this strategy.
The focus of this chapter is not in the formation control framework, but in the way that a hybrid system can improve the performance of the formation controller in many applications by adding a few simple behaviors and a supervisor which generates switching signals while guaranteeing the asymptotic stability of the hybrid formation control system. The hybrid control strategy developed along this chapter involves mobile robot formation control when considering obstacles. Its main objectives are: *i*) place the follower robots at the desired positions in the given formation before starting the leader navigation, this is the so-called static formation problem (Antonelli et al., 2006); *ii*) reduce the temporary large formation errors during the autonomous navigation of the complete robot team; *iii*) avoid unknown obstacles while maintaining the formation geometry, instead of changing the formation geometry as in (Das et al., 2002). For this last objective, it is considered the obstacle contour-following strategy for the leader robot as presented in (Toibero et al., 2006). Regarding the others two major objectives, a hybrid approach based on a formation controller is proposed.
The rest of the chapter includes: Section 2 presents a review of the stable leader-based formation controller. Then, in Section 3 it is described the hybrid control system including simulations results and stability considerations. In Section 4 some comparative simulation results are presented. Finally, in Section 5 experimental results are reported to state conclusions in Section 6.

2. Stable Formation Control

The kinematics model employed in this paper considers formation errors with respect to a Cartesian mobile coordinate system over the leader robot, which Y-axle coincides with the heading of this robot (Fig.1.) The movement of each robot in the world coordinate system (with upper index w) is ruled by the well-known unicycle-like mobile robot kinematics: for the leader in (1) and for the i-th follower in (2)

$$\begin{aligned} {}^{w}\dot{x} &= v\cos\left({}^{w}\theta\right) \\ {}^{w}\dot{y} &= v\sin\left({}^{w}\theta\right) \\ {}^{w}\dot{\theta} &= \omega \end{aligned} \tag{1}$$

$$\begin{aligned} {}^{w}\dot{x}_i &= v_i\cos\left({}^{w}\theta_i\right) \\ {}^{w}\dot{y}_i &= v_i\sin\left({}^{w}\theta_i\right) \\ {}^{w}\dot{\theta}_i &= \omega_i \end{aligned} \tag{2}$$

The leader movement is controlled through its absolute velocities: v and ω. The formation controller objective is to find the values of the velocities v_i and ω_i for the follower robots in such a way that the formation errors decay asymptotically to zero. A third kinematics model must be considered in order to obtain the i-th follower coordinates relative to the leader ($O\ {}^{L}\mathrm{X}\ {}^{L}\mathrm{Y}$) coordinate system which moves at a linear velocity v and angular velocity ω

$$ {}^{L}\dot{x}_i = v_i \cos\left({}^{L}\theta_i\right) + \omega l \sin(\varsigma_i) \tag{3} $$

$$ {}^{L}\dot{y}_i = v_i \sin\left({}^{L}\theta_i\right) - \omega l \cos(\varsigma_i) - v \tag{4} $$

$$ {}^{L}\dot{\theta}_i = \omega_i - \omega \tag{5} $$

here, l is the distance between the robot centre and the origin of the mobile coordinate system and ς_i is the angle between the ^{L}X axis and l (Fig.2.) Note that for a static leader, these equations reduce to

$$ \begin{aligned} {}^{L}\dot{x}_i &= v_i \cos\left({}^{L}\theta_i\right) \\ {}^{L}\dot{y}_i &= v_i \sin\left({}^{L}\theta_i\right) \\ {}^{L}\dot{\theta}_i &= \omega_i \end{aligned} \tag{6} $$

which describe the i-th follower movement on the leader coordinate system. The consideration of the postures of the followers relative to the leader coordinate system allows managing the entire formation without knowing the absolute positions of the followers.

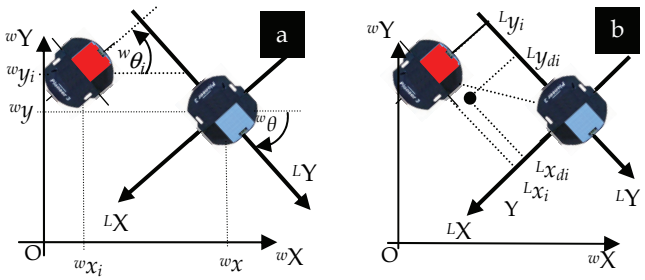


Figure 1. a) Reference systems: world (absolute) reference coordinate system (O ^{w}X ^{w}Y), and a second coordinate system attached to the leader robot (O ^{L}X ^{L}Y) where the desired positions for each of the followers are defined. b) i-th follower robot positioned at coordinates (${}^{L}x_i$, ${}^{L}y_i$) on the leader Cartesian reference where the reference position is given by (${}^{L}x_{di}$, ${}^{L}y_{di}$)

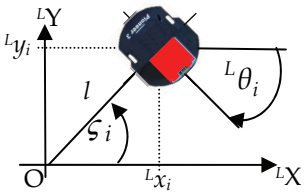


Figure 2. i-th follower in the leader coordinate system

This relative posture can be obtained using a sensor system (for example a cathadioptric vision system) mounted on one of the robots (Fig.3). However, if the absolute postures

information is available, it can be easily converted to the leader coordinate system with the transformation:

$$ {}^{L}x_i = \left({}^{w}x_i - {}^{w}x\right)\sin\left({}^{w}\theta\right) - \left({}^{w}y_i - {}^{w}y\right)\cos\left({}^{w}\theta\right) \tag{7} $$

$$ {}^{L}y_i = \left({}^{w}x_i - {}^{w}x\right)\cos\left({}^{w}\theta\right) + \left({}^{w}y_i - {}^{w}y\right)\sin\left({}^{w}\theta\right) \tag{8} $$

$$ {}^{L}\theta_i = {}^{w}\theta_i - {}^{w}\theta + \pi/2 \tag{9} $$

which gives the relation between the absolute $({}^{w}x_i, {}^{w}y_i)$ and the relative $({}^{L}x_i, {}^{L}y_i)$ coordinates. On the other hand, the *i*-th follower absolute heading angle can be transformed to the leader coordinate system by using equation (9).

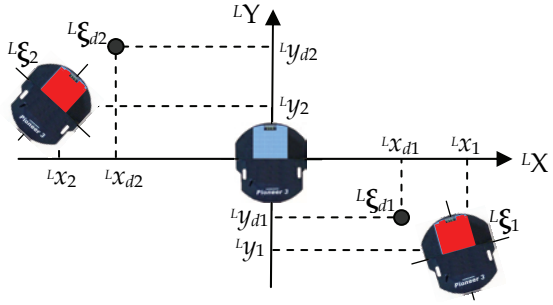


Figure 3. Representation of the vectors ${}^{L}\xi_i$ and ${}^{L}\xi_{di}$

In Fig.4 it can be seen the block diagram of the proposed controller. From this figure, it must be noted the formation controller independence on the leader motion generation, that is, the leader navigates according to its own motion laws and the formation controller only needs the commands computed by the leader controller.

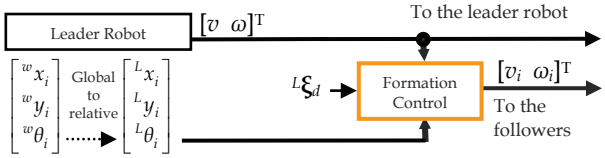


Figure 4. Formation control block diagram

In order to calculate an error indicator between the current and the desired positions of the robots in the formation, let consider that (10) is the position vector of the *i*-th follower robot, and that (11) denotes the *i*-th follower desired position, with *i*=1, 2,..., *n*.

$$ {}^{L}\xi_i = \begin{bmatrix} {}^{L}x_i & {}^{L}y_i \end{bmatrix}^{\mathrm{T}} \tag{10} $$

$$ {}^{L}\xi_{di} = \begin{bmatrix} {}^{L}x_{di} & {}^{L}y_{di} \end{bmatrix}^{\mathrm{T}} \tag{11} $$

Both vectors are defined on the framework attached to the leader robot (Fig.3.) The *n* individual position vectors (10) and (11) can be arranged in the global position vectors:

$$ {}^{L}\boldsymbol{\xi} = \begin{bmatrix} {}^{L}\boldsymbol{\xi}_1^{\mathrm{T}} & {}^{L}\boldsymbol{\xi}_2^{\mathrm{T}} & \cdots & {}^{L}\boldsymbol{\xi}_n^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}} \tag{12} $$

$$ {}^{L}\boldsymbol{\xi}_d = \begin{bmatrix} {}^{L}\boldsymbol{\xi}_{d1}^{\mathrm{T}} & {}^{L}\boldsymbol{\xi}_{d2}^{\mathrm{T}} & \cdots & {}^{L}\boldsymbol{\xi}_{dn}^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}} \tag{13} $$

The difference between the actual and the desired robot position is (14); and the formation error is defined in (15) as follows (Kelly et al., 2004; Carelli et al., 2006)

$$ {}^{L}\tilde{\boldsymbol{\xi}} = {}^{L}\boldsymbol{\xi}_d - {}^{L}\boldsymbol{\xi} \tag{14} $$

$$ \tilde{\mathbf{h}} = \mathbf{h}_d - \mathbf{h} = \mathbf{h}\left({}^{L}\boldsymbol{\xi}_d\right) - \mathbf{h}\left({}^{L}\boldsymbol{\xi}\right) \tag{15} $$

$$ \mathbf{h} = \mathbf{h}\left({}^{L}\boldsymbol{\xi}\right) = \mathbf{h}\left({}^{L}\boldsymbol{\xi}_d - {}^{L}\tilde{\boldsymbol{\xi}}\right) \tag{16} $$

where $\mathbf{h}$ is a suitable selected output variable representing the formation parameters, which captures information about the current conditions of the group of robots; $\mathbf{h}_d$ represents the desired output variable. For instance, $\mathbf{h}$ can be selected as the *xy*-position of each follower robot. Function $\mathbf{h}\left({}^{L}\boldsymbol{\xi}\right)$ must be defined in such a way to be continuous and differentiable, and the Jacobian matrix J has full rank.

$$ \dot{\mathbf{h}} = \mathbf{J}(\boldsymbol{\xi})\, {}^{L}\dot{\boldsymbol{\xi}} = \frac{\partial\, \mathbf{h}\left({}^{L}\boldsymbol{\xi}\right)}{\partial\, {}^{L}\boldsymbol{\xi}}\, {}^{L}\dot{\boldsymbol{\xi}} \tag{17} $$

Vector ${}^{L}\dot{\boldsymbol{\xi}}$ (robot translational velocities in the leader reference system) has two different components,

$$ {}^{L}\dot{\boldsymbol{\xi}} = {}^{L}\dot{\boldsymbol{\xi}}_s - {}^{L}\dot{\boldsymbol{\xi}}_l \tag{18} $$

where ${}^{L}\dot{\boldsymbol{\xi}}_s$ is the time variation of ${}^{L}\boldsymbol{\xi}$ produced by the velocities of the follower robots $\left\| {}^{L}\dot{\boldsymbol{\xi}}_{si} \right\| = v_i$; and ${}^{L}\dot{\boldsymbol{\xi}}_l$ is the time variation of ${}^{L}\boldsymbol{\xi}$ produced by the velocities of the leader robot. Now, (17) can be written as:

$$ \dot{\mathbf{h}} = \mathbf{J}\left({}^{L}\boldsymbol{\xi}\right)\left({}^{L}\dot{\boldsymbol{\xi}}_s - {}^{L}\dot{\boldsymbol{\xi}}_l\right) \tag{19} $$

The control objective is to guarantee that the mobile robots will asymptotically achieve the desired formation, that is, $\lim_{t\to\infty} \tilde{\mathbf{h}}(t) = 0$. To this aim, it is first defined a reference velocities vector as:

$$ {}^{L}\dot{\boldsymbol{\xi}}_r = \mathbf{J}^{-1}\left({}^{L}\boldsymbol{\xi}\right)\left[\dot{\mathbf{h}}_d + \mathbf{K}\, \mathbf{f}_{\tilde{\mathbf{h}}}\left(\tilde{\mathbf{h}}\right)\right] + {}^{L}\dot{\boldsymbol{\xi}}_l \tag{20} $$

where K is a symmetric and positive definite gain matrix; $\mathbf{f}_{\tilde{\mathbf{h}}}\left(\tilde{\mathbf{h}}\right)$ is a saturation function applied to the output error, such that $\mathbf{x}^{\mathrm{T}}\, \mathbf{f}_{\tilde{\mathbf{h}}}(\mathbf{x}) > \mathbf{0} \quad \forall \mathbf{x} \neq \mathbf{0}$. This function could be selected for example as $\mathbf{f}_{\tilde{\mathbf{h}}}(\mathbf{x}) = \tanh(\mathbf{x})$. ${}^{L}\dot{\boldsymbol{\xi}}_r$ represents the velocities of the followers robots on the framework attached to the leader robot that allow them to reach (and to maintain) the desired formation while following the leader. Assuming perfect velocity servoing

$$ {}^{L}\dot{\boldsymbol{\xi}}_s \equiv {}^{L}\dot{\boldsymbol{\xi}}_r \tag{21} $$

then from (19),(20) and (15) the following closed loop equation can be obtained:

$$ \dot{\tilde{\mathbf{h}}} + \mathbf{f}_{\tilde{\mathbf{h}}}\left(\tilde{\mathbf{h}}\right) = \mathbf{0} \tag{22} $$

Now, in order to consider the formation errors analysis under the perfect servoing assumption the following Lyapunov candidate function (Slotine & Li, 1991) is introduced

$$ V = \tilde{\mathbf{h}}^T \tilde{\mathbf{h}} / 2 \tag{23} $$

with its time-derivative along system trajectories

$$ \dot{V} = \tilde{\mathbf{h}}^{\mathrm{T}} \dot{\tilde{\mathbf{h}}} = -\tilde{\mathbf{h}}^{\mathrm{T}} \mathbf{K} \mathbf{f}_{\tilde{\mathbf{h}}}\left(\tilde{\mathbf{h}}\right) < 0 \tag{24} $$

It is clear that if ${}^{L}\dot{\boldsymbol{\xi}}_s \equiv {}^{L}\dot{\boldsymbol{\xi}}_r \Rightarrow \tilde{\mathbf{h}}(t) \to 0$ asymptotically.

Remark 1. This condition is verified for the ideal case in which the robots follow exactly the reference velocity (21). However, for a real controller this velocity equality will eventually be reached asymptotically. The convergence of the control error to zero under this real condition will be analyzed at the end of this section.

Vector ${}^{L}\dot{\boldsymbol{\xi}}_l$ in (20) is computed using the knowledge of linear and angular velocities of the leader robot, and the relative positions of the follower robots:

$$ r_1 = v / \omega \tag{25} $$

$$ r_{2i} = \sqrt{(r_1 + x_i)^2 + y_i^2} \tag{26} $$

$$ \beta_i = \arctan\left(y_i / (r_1 + x_i)\right) \tag{27} $$

$$ \left\| {}^{L}\dot{\boldsymbol{\xi}}_{li} \right\| = \omega\, r_{2i} \;\text{ and} \begin{bmatrix} {}^{L}\dot{\xi}_{lxi} \\ {}^{L}\dot{\xi}_{lyi} \end{bmatrix} = \begin{bmatrix} -\left\| {}^{L}\dot{\boldsymbol{\xi}}_{li} \right\| \sin\left(\beta_i\right) \\ \left\| {}^{L}\dot{\boldsymbol{\xi}}_{li} \right\| \cos\left(\beta_i\right) \end{bmatrix} \tag{28} $$

where r_1 and r_{2i} are virtual turning radius (Fig.5) and subscript i denotes the i-th follower.

Remark 2. In the case $\omega = 0$, vector ${}^{L}\dot{\boldsymbol{\xi}}_{li}$ is calculated as ${}^{L}\dot{\boldsymbol{\xi}}_{li} = [0 \quad v]^{\mathrm{T}}$.

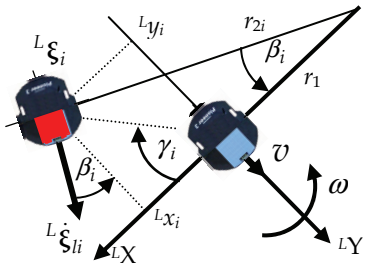


Figure 5. Velocity computation

The commands for the linear and angular velocities of each robot are computed in order to secure that the robots will reach asymptotically the velocity reference (${}^{L}\dot{\boldsymbol{\xi}}_s \to {}^{L}\dot{\boldsymbol{\xi}}_r$).

The proposed control law for heading control is:

$$\omega_i = k_{\omega i} f\left({}^L\tilde{\theta}_i\right) + {}^L\dot{\theta}_{ri} + \omega \tag{29}$$

where ${}^L\tilde{\theta}_i = {}^L\theta_{ri} - {}^L\theta_i$ is the angular error between the i-th follower robot heading ${}^L\theta_i$ and the angle of its reference velocity ${}^L\theta_{ri} = \angle\left(\dot{\boldsymbol{\xi}}_{ri}\right)$; consequently ${}^L\dot{\theta}_{ri}$ is the time derivative of this reference velocity heading for the i-th robot; ω is the angular velocity of the mobile framework attached to the leader robot; and $f\left({}^L\tilde{\theta}_i\right)$ is a saturation function applied to the angular error, which has the same properties of function $\mathbf{f}_{\tilde{\mathbf{h}}}\left(\tilde{\mathbf{h}}\right)$ included in (20) ; and $k_{\omega i}$ is a positive constant. Next, by equating (29) and (5), the following closed-loop equation can be obtained:

$${}^L\dot{\tilde{\theta}}_i + k_{\omega i} f\left({}^L\tilde{\theta}_i\right) = 0 \tag{30}$$

Now, in order to analyze the stability for the heading control, it is introduced the following Lyapunov candidate (Slotine & Li, 1991)

$$V = {}^L\tilde{\theta}_i^2 / 2 \tag{31}$$

and its time derivative

$$\dot{V} = {}^L\tilde{\theta}_i \; {}^L\dot{\tilde{\theta}}_i = -k_{\omega i} \; {}^L\tilde{\theta}_i \, f\left({}^L\tilde{\theta}_i\right) < 0 \tag{32}$$

which implies that ${}^L\tilde{\theta}_i(t) \to 0$ as $t \to \infty$. That is, the robot orientation on the leader Cartesian coordinate system tends asymptotically to the desired reference orientation, which guarantees maintaining the desired formation. Once it was proved that ${}^L\theta_i(t) \to {}^L\theta_{ri}(t)$, it must now be proved that the same occurs for $\left\| {}^L\dot{\boldsymbol{\xi}}_{si} \right\| = v_i \to \left\| {}^L\dot{\boldsymbol{\xi}}_{ri} \right\|$. To this aim, the following control law for the linear velocity is proposed:

$$v_i = \left\| {}^L\dot{\boldsymbol{\xi}}_{ri} \right\| \cos\left({}^L\tilde{\theta}_i\right) \tag{33}$$

which obviously produces that $v_i \to \left\| {}^L\dot{\boldsymbol{\xi}}_{ri} \right\|$, since it has been proved that ${}^L\tilde{\theta}_i(t) \to 0$. The factor $\cos\left({}^L\tilde{\theta}_i\right)$ has been added to prevent high control actions when a large angular error exists. Now, we have proved that ${}^L\dot{\boldsymbol{\xi}}_r - {}^L\dot{\boldsymbol{\xi}}_s = \boldsymbol{\rho}$ with $\boldsymbol{\rho}(t) \to 0$, which is a more realistic assumption than (21). Then, formation errors are considered again in order to analyze its stability under the new condition. So, (22) can be written as :

$$\dot{\tilde{\mathbf{h}}} + \mathbf{K}\,\mathbf{f}_{\tilde{\mathbf{h}}}\left(\tilde{\mathbf{h}}\right) = \mathbf{J}\,\boldsymbol{\rho} \tag{34}$$

Let us consider the same Lyapunov candidate (23) but now with its time derivative:

$$\dot{V} = \tilde{\mathbf{h}}^{\mathrm{T}} \dot{\tilde{\mathbf{h}}} = -\tilde{\mathbf{h}}^{\mathrm{T}} \left(\mathbf{K}\,\mathbf{f}_{\tilde{\mathbf{h}}}\left(\tilde{\mathbf{h}}\right) - \mathbf{J}\,\boldsymbol{\rho} \right) \tag{35}$$

A sufficient condition for (35) to be negative definite is

$$\left\|\mathbf{f}_{\tilde{\mathbf{h}}}\left(\tilde{\mathbf{h}}\right)\right\| > \frac{\|\mathbf{J}\|\|\boldsymbol{\rho}\|}{\|\mathbf{K}\|} \tag{36}$$

As $\boldsymbol{\rho}(t) \to \mathbf{0}$ and taking into account the properties of function $\mathbf{f}_{\tilde{\mathbf{h}}}\left(\tilde{\mathbf{h}}\right)$, it can be concluded that $\left\|\tilde{\mathbf{h}}(t)\right\| \to 0$ as $t \to \infty$.

3. Hybrid Formation Control

The interaction between the leader and the follower controllers must be in such a way that the followers always maintain their desired positions independently of the leader manoeuvres. This allows preserving the formation and therefore the involved robots can perform a cooperative task. Our approach is based on the detection of leader movements that will significantly increase transitory formation errors. In Fig.6 we present a hybrid formation control strategy where it can be appreciated the inclusion of a supervisor which generates switching signals at both levels: leader (σ_L) and followers (σ_{Fi}) based on: *i*) the follower posture, *ii*) the leader absolute posture and *iii*) the leader control actions. Besides, it was also included a new orientation controller, that corrects the followers heading accordingly to a given logic (next, in Fig.9).

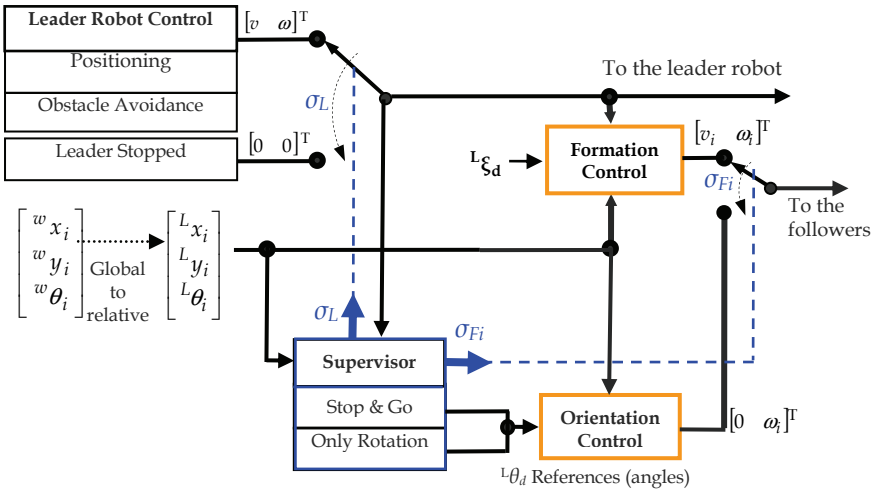


Figure 6. Hybrid formation control block diagram

The main idea is to detect leader movements that will immediately produce formation errors. These errors will arise due to the non-holonomic constraint of the unicycle-like wheeled mobile robots (mostly due to different robot headings.) Hence, the headings of the followers are set to values that prevent these initial errors and only after this correction is done, the leader is allowed to continue with its planned movement. These leader movements (which are detected directly from the leader control commands) are namely: *i*) "stop & go" (a step in the forward velocity) and *ii*) "only-rotation movements" (a step in the angular velocity command with null forward velocity).

The leader motion control is based on the results of (Toibero et al., 2007) and gives the robot the capability to get a desired posture $[{}^{w}x_{DES}, {}^{w}y_{DES}, {}^{w}\theta_{DES}]^{T}$ in the world coordinate system while avoiding obstacles. This motion could only be stopped by the supervisor ("leader stopped" in the block diagram of Fig.6). This strategy allows separating completely the control analysis into the leader motion control analysis and the follower motion control analysis. For the leader this analysis is trivial since its motion control is asymptotically stable, then the new control system which is assumed to include the possibility to stop the leader during a finite time, will also be asymptotically stable. Now, regarding the follower robots, the inclusion of the orientation controller must be considered into the stability analysis (Section 3.2). Note the existence of a switching signal σ_{Fi} for each follower, and consequently, an orientation controller available for each follower robot.

3.1 Follower Robots: Heading Control

In this section it is introduced a proportional only-bearing controller that allows the follower robots to set their headings to desired computed values that will provide good initial heading conditions for the future formation evolution. The position of each follower robot in the formation is defined by its coordinates $({}^{L}x_{id}, {}^{L}y_{id})$ regardless of its orientation. Taking advantage of the unicycle kinematics, it will be assumed from here on that the robots can rotate without distorting the formation (allowing change the "formation heading"). In other words, for instance, if the robots are transporting an object, they must be able to turn freely over its own centers without changing the transported object orientation. It is proposed a proportional controller for the heading error

$$ {}^{L}\tilde{\theta}_i = {}^{L}\theta_d - {}^{L}\theta_i \tag{37}$$

where the desired value ${}^{L}\theta_d$ is computed according to Section 3.2. Then, proposing the following Lyapunov function (38) with the control action (39) the asymptotic stability of this control system could be immediately proved by (40)

$$ V_{ori} = {}^{L}\tilde{\theta}_i^{2} / 2 \tag{38}$$

$$ \omega_i = -k_{\omega i} \tanh\left({}^{L}\tilde{\theta}_i\right) \tag{39}$$

$$ \dot{V}_{ori} = {}^{L}\tilde{\theta}_i \, {}^{L}\dot{\tilde{\theta}}_i = -k_{\omega i} \, {}^{L}\tilde{\theta}_i \tanh\left({}^{L}\tilde{\theta}_i\right) < 0 \tag{40}$$

The importance of introducing the heading controller can be appreciated from Fig.7, starting with a null error formation (Fig.7.a) the leader develops a pure-rotational evolution (Fig.7.b), and the follower tries to keep the formation with a significant transition error. It is clear that this error could be avoided if the starting orientation of the follower robot is set to ψ before the leader starts its rotation. This angle is the orientation of the first velocity reference vector ${}^{L}\dot{\boldsymbol{\xi}}_r$ and is computed depending of the sign of the leader angular velocity according to:

$$ \psi_i = \gamma_i + \operatorname{sgn}(w)\pi / 2 \tag{41}$$

where the angle γ_i is given by

$$ \gamma_i = \tan^{-1}\left({}^{L}y_{id} / {}^{L}x_{id}\right) \tag{42}$$

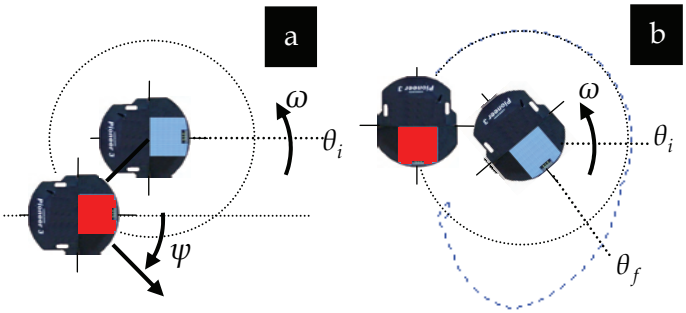


Figure 7. Formation control without orientation control for a leader (at the centre) and a follower: a) initial configuration; b) the path described by the follower (dotted line)

Moreover, depending on the leader angular velocity, formation errors could be greater or even produce follower backward movements. This ψ-angle correction avoids transitory formation errors improving the whole control system performance. The same analysis could be done for the leader "stop & go" movement with heading errors on the follower robots. In this case, the leader attempts to start its translational motion and it is easy to see that the robot configuration that will present minimal formation error at this transitory will be the formation in which all robots have the same heading angle.

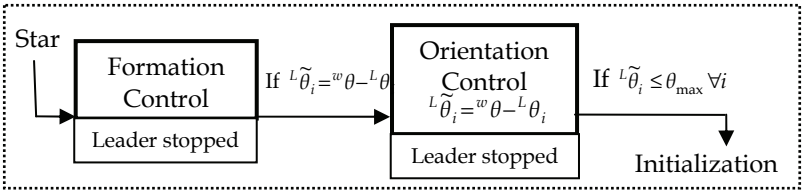


Figure 8. Initialization logic: static formation

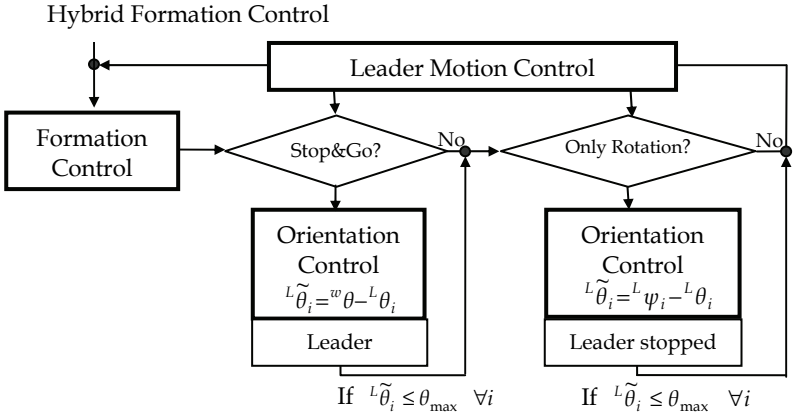


Figure 9. General hybrid formation logic

3.2 Stability Analysis

The supervisor logics were divided into two cases: an initialization case (or static formation case of Fig.8) that corrects followers' initial postures to a new posture with null formation error and with the same leader heading; and the general case that allows keeping the formation geometry (Fig.9) when the leader starts moving. The orientation control is used in two situations: one related to the leader only-bearing movement that corrects the ψ-angle for each follower; and the other related to the leader "stop & go" movement, that equals all the followers headings to the leader heading. In both cases the objective is to minimize the heading errors before starting the leader movement. Accordingly to the exposed logics, it is considered a switching between the formation controller of Section 2 and an orientation controller of Section 3.1 for each follower which stability at switching times must be analyzed. This is done by considering Multiple Lyapunov Functions (Liberzon, 2003): It must be guaranteed that the sequence associated to the discontinuous Lyapunov Functions (when are active) be decreasing for all the controllers involved and furthermore, it must be guaranteed also that the switching is not arbitrarily fast. In Fig.10 it is depicted a typical switching instant (at $<t_1>$) for a three robot formation. At this point, the leader is stopped, and the orientation controllers compute their references (note the existence of different values for each robot); then, at $<t_2>$ follower 1 has achieved the maximum acceptable error θ_{max}, however the formation controller will not start with its movement after instant $<t_3>$ when the second follower has achieved its maximum heading error.

In consequence the logics secure: *i*) that the switching from the orientation control back to the formation control is slow enough to allow the followers to achieve its desired postures avoiding the undesirable *chattering* effect; *ii*) that the value of (23) is the same before and after the switching since it does not depend on the follower' headings because the formation error is defined only as a function of the followers' positions ${}^{L}\xi$. This fact can be seen in Fig.10 where $V(t_1)=V(t_3)$. This way, the asymptotic stability proved for the formation controller (and its performance) will not be affected by the proposed switching.

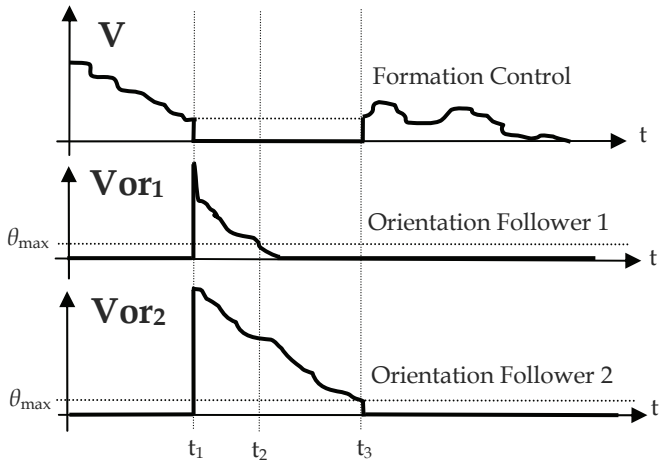


Figure 10. Multiple Lyapunov Function approach

4. Simulation Results

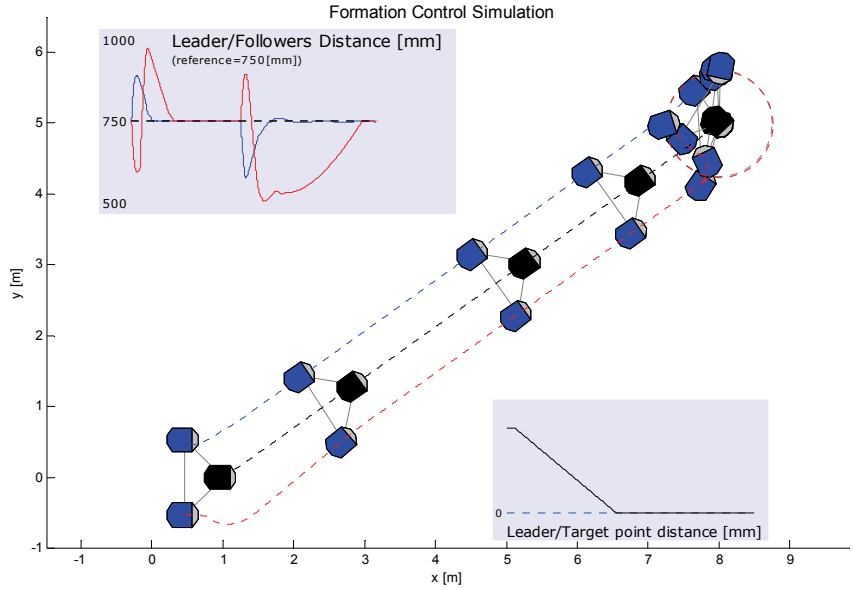


Figure 11. Formation control simulation

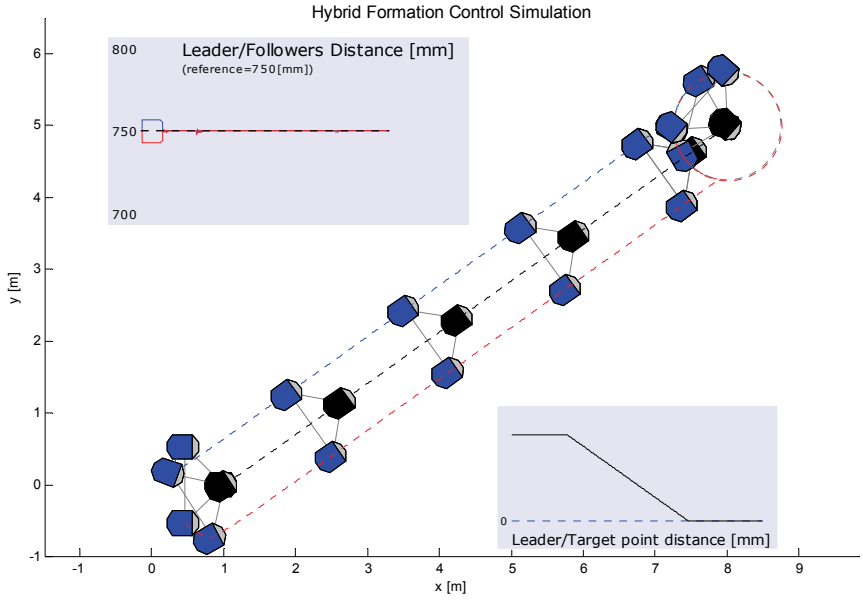


Figure 12. Hybrid formation control simulation

Before introducing the experimental results obtained with real robots we present some comparative simulation results. The strategies of Sections 2 and 3 are compared with the aim of highlight the improvement on the performance achieved with the hybrid formation proposal.

It is considered the same simulation experiment for a three-robot triangle formation under both controllers. The task consists of a simple free obstacle navigation between two points. The experiment includes both situations mentioned along this chapter: "stop & go" and "only rotation".

Figure 11 shows the simulation results for the formation controller of Section 2 where large transitory formation errors can be appreciated. These formation errors appear due to the "stop & go" situation at the beginning of the experiment and due to the leader "only heading" movement when the leader robot achieve the goal point. In spite of the stability of this controller (the formation errors tend asymptotically to zero), those transitory errors could be unacceptable for many applications.

On the other hand, Fig.12 presents the results for the same simulation experiment but using the hybrid formation controller. It can be noted that this hybrid strategy is able to deal better with the "stop & go" and "only heading" movement, considerably reducing transitory formation errors.

5. Experimental Results

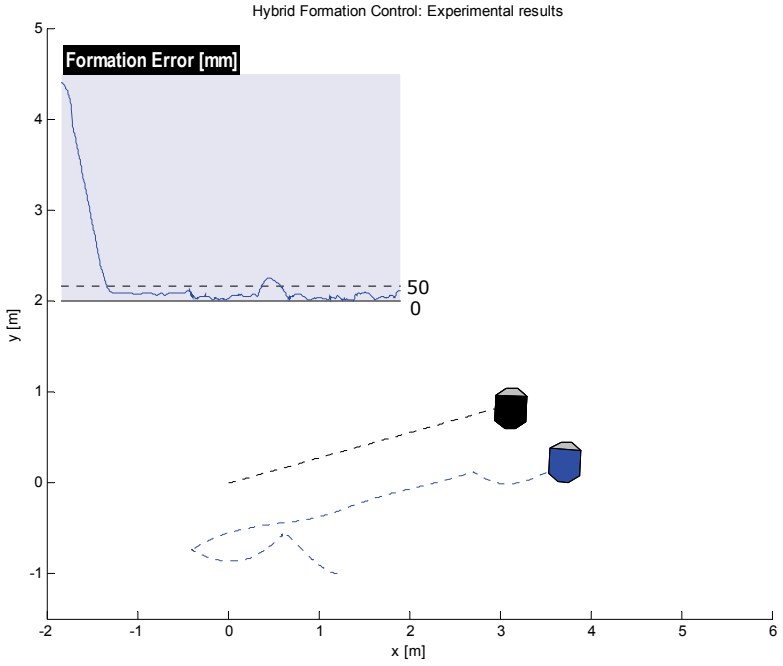


Figure 13. Experimental results: formation control without obstacles including initial formation error

Experimental results were performed by using two Pioneer robots with onboard PCs and wireless internet connection. This way, each sample time (Ts = 100ms) the leader robot asks for the follower position and after computing the control commands, sends them back to the follower. It was considered a reference translational velocity of 150mm/s and a maximum angular velocity of 50º/s for the leader.

In Fig.13 it can be appreciated the formation evolution within a room without obstacles. In the first part, it is considered the static formation problem for a follower initial posture of $[{}^{L}x_1, {}^{L}y_1, {}^{L}\theta_1]^T = [1200, -1000, 180^o]^T$ and a desired formation at $({}^{L}x_{d1}, {}^{L}y_{d1}) = (600, -600)$. It can be appreciated the formation error correction according to the initialization logic of Fig.9. After the formation geometry is achieved, the leader robot is allowed to start with its motion towards the goal point at $[{}^{w}x_{DES}, {}^{w}y_{DES}, {}^{w}\theta_{DES}]^T = [3100, 850, 90^o]^T$ from the initial posture at $[0, 0, 90^o]^T$.

Finally, Fig.14 shows the robot trajectories for a similar experiment but considering an obstacle which is detected with the leader laser range finder. In this case the desired formation point was set to $({}^{L}x_{d1}, {}^{L}y_{d1}) = (0, -600)$ and the maximum formation error was 105mm while the mean value was of 28mm (that could be compared with the error values for the previous experiment: maximum value of 720mm and mean value of 22mm). From these two previous plots, it can be concluded the low formation error values achieved for the robot formations.

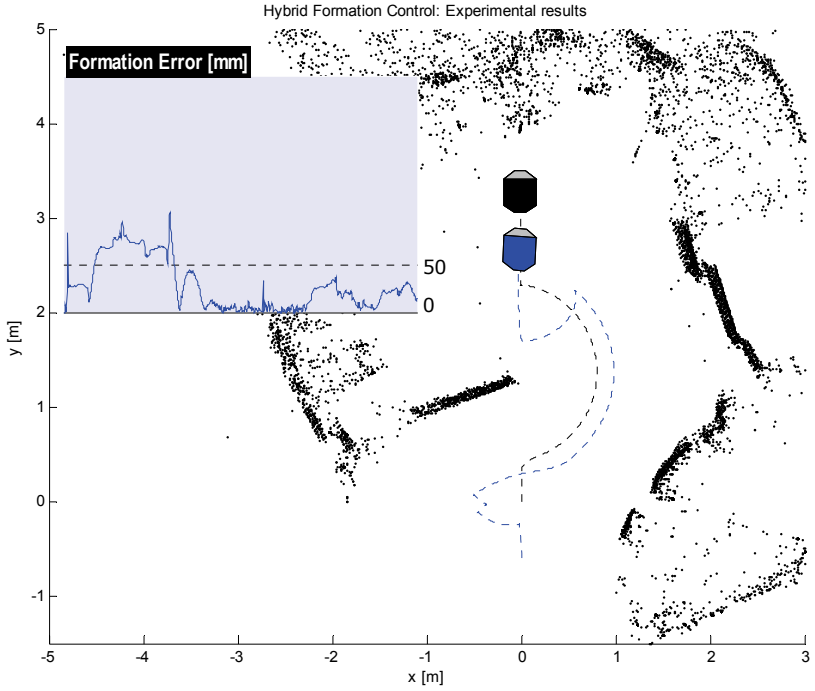


Figure 14. Experimental results: formation control with obstacles

6. Conclusions

In this chapter it has been addressed the problem of the autonomous navigation for a group of non-holonomic mobile robots. In a first stage we considered the classic leader-based formation control problem. In spite of the stability property of this controller, we have detected large transitory errors in some circumstances, being these errors unacceptable for many applications, such as transporting large objects in a cooperative way.
Based on these observations and in order to present a formal solution, we have developed a hybrid approach for the formation problem. The continuous formation controller has been complemented with an orientation controller for each follower, allowing a considerable reduction of formation errors during leader manoeuvres. The resulting hybrid control system presents a switched architecture characterized by the presence of a supervisor which generates a switching signal indicating the active controller at any moment.
Besides, it has been included a formal stability proof for the whole switched system based on the theory of multiple Lyapunov functions.
At the end of this chapter, we exposed simulations results that allow comparing both main strategies. Next, we have included experimental results for a two-robots formation navigating on different settings: without obstacles, and avoiding isolated obstacles by considering a reactive algorithm on the leader robot. Through these experimental results it can be concluded the good performance of the hybrid approach.
Future works on this area will be related to the improvement of the obstacle avoidance capability and to increase the perception abilities of the follower robots (adding new sensors).

7. Acknowledgments

The authors gratefully acknowledge SEPCIT (FONCYT) and CONICET of Argentina for partially funding this research.

8. References

Antonelli, G.; Arrichiello, F. & Chiaverini, S. (2006). Experiments of formation control with collision avoidance using the null-space-based behavioral control, *Proceedings of IEEE Mediterranean Conference on Control and Automation,* pp. 1-6, ISBN: 0-9786720-0-3, Ancona, Italy, June 2006.

Carelli, R.; De la Cruz, C. & Roberti, F. (2006). Centralized formation control of non-holonomic mobile robots, *Latin American Applied Research*, Vol. 36, Nº 2, 2006, pp.63-69, ISSN: 0327-0793.

Chio, T. & Tarn, T. (2003). Rules and control strategies of multi-robot team moving in hierarchical formation, *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 2701-2706, ISBN: 0-7803-7736-2, Taipei, Taiwan, September 2003.

Das, A. K.; Fierro, R.; Kumar, V.; Ostrowski, J. P.; Spletzer, J. & Taylor, C. J. (2002). A vision-based formation control framework, *IEEE Transaction on Robotics & Automation*, Vol. 18, Nº 5 pp. 813-825, ISSN: 1042-296X.

Desai, J. P.; Kumar, V. & Ostrowski, J. P. (1999). Control of changes in formation for a team of mobile robots, *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 1556-1561, ISBN: 0-7803-5180-0, Detroit, MI, USA, May 1999.

Desai, J. P.; Ostrowski, J. P. & Kumar, V. (2001). Modeling and control of formations of nonholonomic mobile robots, *IEEE Transaction on Robotics & Automation*, vol. 17, Nº 6, pp.905-908, ISSN: 1042-296X.

Fierro, R.; Song, P.; Das, A. K. & Kumar, V. (2002). Cooperative control of robot formations, In: *Cooperative control and cooperation*, R. Murphey and P. Pardalos, (Ed.), pp. 73-93, Kluwer Academic Press, ISBN: 1-4020-0549-0, Dordrecht, The Netherlands.

Fredslund, J. & Mataric, M.J. (2002). A general algorithm for robot formations using local sensing and minimal communication, *IEEE Transactions on Robotics and Automation*, vol. 18, Nº 5, pp. 837-846, ISSN: 1042-296X.

Gava, C. C.; Vassallo, R.; Roberti, F.; Carelli, R. & Freire Bastos, T. (2007). Nonlinear control techniques and omnidirectional vision for team formation on cooperative robotics, *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 2409-2414, ISBN: 1-4244-0601-3, Roma, Italy, April 2007.

Gulec, N. & Unel, M. (2005). Coordinated motion of autonomous mobile robots using nonholonomic reference trajectories, *Proceedings of Conference of IEEE Industrial Electronics Society*, pp. 339-344, ISBN: 0-7803-9252-3, Raleigh, NC, USA, November 2005.

Kelly, R.; Carelli, R.; Ibarra Zannatha, J. M. & Monroy, C. (2004). Control de una pandilla de robots móviles para el seguimiento de una constelación de puntos objetivo, *Congreso Mexicano de Robótica*, pp. 83-89, Torreón, México, October 2004.

Liberzon, D. (2003). *Switching in Systems and Control*, Birkhauser, ISBN: 0-8176-4297-8, Boston, MA, USA.

Ogren, P. & Leonard, N. E. (2003). Obstacle avoidance in formation, *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 2492-2497, ISBN: 0-7803-7736-2, Taipei, Taiwan, September 2003.

Ogren, P. (2004). Split and join of vehicle formations doing obstacle avoidance, *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 1951-1955, ISBN: 0-7803-8232-3, New Orleans, LA, USA, April 2004.

Shao, J.; Xie, G.; Yu, J. & Wang, L. (2005). Leader-following formation control of multiple mobile robots, *Proceedings of IEEE International Symposium on Intelligent Control*, pp. 808-813, ISBN: 0-7803-8936-0, Limassol, Cyprus, June 2005.

Slotine, J. & Li, W. (1991). *Applied non linear control*, Prentice Hall, ISBN: 0-1304-0890-5, New Jersey, USA.

Toibero, J. M.; Carelli, R. & Kuchen, B. (2006). Switching Contour-Following controller for wheeled mobile robots, *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 3724-3729, ISBN: 0-7803-9505-0, Orlando, USA, May 2006.

Toibero, J. M.; Carelli, R. & Kuchen, B. (2007). Switching control of mobile robots for autonomous navigation in unknown environments, *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 1974-1979, ISBN: 1-4244-0601-3, Roma, Italy, April 2007.

13

A Novel Modeling Method for Cooperative Multi-robot Systems Using Fuzzy Timed Agent Based Petri Nets

Hua Xu
State Key Laboratory of Intelligent Technology and Systems, Tsinghua National Laboratory for Information Science and Technology, Department of Computer Science and Technology, Tsinghua University, Beijing
P. R. China

1. Introduction

Characterized as cooperation and high efficiency, cooperative multi-robot systems (CMRS) have emerged as usual manufacturing equipments in current industries (Cao et al., 1997). Differing from generic control systems, the cooperation needs to be considered in the realization of CMRS (Cao et al., 1997). So the system modeling, analysis and refinement always meet with difficulties. As one of the typical multi-agent systems (MAS), CMRS can be regarded as one MAS in distributed artificial intelligence (Jennings et al., 1998). For modeling MAS, some attempts to use object-oriented methodology have been tried and some typical agent objects have been proposed, such as *active object*, etc (Guessoum & Briot, 1999). However, agent based object models still can not depict the structure and dynamic aspects of MAS, such as cooperation, learning, temporal constraints, etc (Jennings et al., 1998).

On one hand, as one of the most proper and promised realization technology, object-oriented concurrent programming (OOCP) methodology is always used to realize MAS (Guessoum & Briot, 1999). In OOCP realization, one special object called *active object* is proposed (Guessoum & Briot, 1999), which can be used to model generic agent architectures and behaviors with OO methodology. Although OOCP can solve MAS realization problem favorably, the modeling problem mentioned above still exists.

On the other hand, as a kind of powerful formal description and analysis method for dynamic systems (Murata, 1989), Petri net (PN) has become a bridge between practical application and model theories (Murata, 1989). With object-oriented concepts introduced into PN, typical object Petri nets such as HOONet (Hong & Bae, 2000) etc, al. are presented, which are strict OPNs. Then for providing the ability of modeling time critical complex systems, timed hierarchical object oriented Petri net (TOPN) (Xu & Jia, 2006) is proposed on the base of HOONet (Hong & Bae, 2000). It supports temporal knowledge description and object-oriented concepts. Modeling features in TOPN support describing and analyzing dynamic systems such as MAS and CMRS. Recently, some attempts have been conducted on modeling MAS by using OPN (Chainbi, 2004). Although Petri nets can be used to model and analyze different systems, they have failed to model learning ability and the aging effects in

dynamic systems. Recently, fuzzy timed Petri net (FTPN) (Pedrycz & Camargo, 2003) has been presented to solve these modeling problems. As a kind of reasoning and learning ability, fuzzy reasoning in FTPN can be considered as supporting autonomous judging or reasoning ability in MAS. In order to solve the reasoning ability and other modeling problems in large-scale MAS, fuzzy timed object-oriented Petri net (FTOPN) (Xu & Jia, 2005) is proposed on the base of TOPN (Xu & Jia, 2006) and FTPN (Pedrycz & Camargo, 2003). In FTOPN, agent can be modeled as one FTOPN object with autonomy, situatedness and sociality. However, in FTOPN every agent should be modeled from common FTOPN objects. To model CMRS—"one of the typical MAS" needs generic FTOPN agent objects on the base of *active object*s (Guessoum & Briot, 1999).

This chapter proposes a high level PN called fuzzy timed agent based Petri net (FTAPN) on the base of FTOPN (Xu & Jia, 2005). As one of the typical *active objects*, ACTALK object is modeled by FTOPN and is introduced into FTAPN, which is used as one generic agent object in FTAPN. The aim of FTAPN is to solve the agent or CMRS modeling ability problem and construct a bridge between MAS models and their implementations.

This chapter is organized as the following. Section 2 reviews the relative preliminary notations of active objects, TOPN and FTOPN quickly. Section 3 extends FTOPN to FTAPN on the base of ACTALK model. Section 4 discusses the learning which is important for representing dynamic behaviors in CMRS. Section 5 uses FTAPN to model one CMRS in the wafer etching procedure of circuit industry and makes some modeling analysis to demonstrate its benefits in modeling MAS. Finally, the conclusion and future work can be found in section 6.

2. Preliminary Notations

In this section, the basic concepts of ACTALK are firstly reviewed. Then the definitions of TOPN are introduced quickly. Finally, the concepts of FTOPN are reviewed formally.

2.1 ACTALK

2.1.1 Active Objects (Guessoum & Briot, 1999)

Object-oriented concurrent programming (OOCP) is one of the most appropriate and promising technologies for implementing or realizing agent based systems or MAS. Combining the agent concept and the object-oriented paradigm leads to the notion of agent-oriented programming (Shoham, 1993). The uniformity of objects' communication mechanisms provides facilities for implementing agent communication, and the concept of encapsulating objects or encapsulation support combining various agent granularities. Furthermore, the inheritance mechanism enables knowledge specialization and factorization.

The concept of an active object has been presented, which makes it possible to integrate an object and activity (namely a thread or process). It also provides some degree of autonomy for objects in that it does not rely on external resources for activation. Thus, it provides a good basis for implementing agent based systems or MAS. However, similar to common objects in Object-oriented systems, an active object's behavior still remains procedural and only reacts to message requests. More generally, the main feature of agent-based systems or MAS is autonomous. Agents should be able to complete tasks autonomously. That's to say, agents must be able to perform numerous functions or

activities without external intervention over extended time periods. In order to achieve autonomy, adding to an active object a function that controls message reception and processing by considering its internal state is one of effective realization methods (Briot, 1996) (Maruichi et al., 1990).

On one hand, for modelling and realizing MAS, there are two basic questions regarding how to build a bridge between implementing and modelling MAS requirements (Castelfranchi, 1995) (Gasser, 1992). On the other hand, the facilities and techniques OOCP provides (Gasser & Briot, 1992):

- How can a generic structure define an autonomous agent's main features?
- How do we accommodate the highly structured OOCP model in this generic structure?

The active-object (or actor) concept has been introduced to describe a set of entities that cooperate and communicate through message passing. This concept brings the benefits of object orientation (for example, modularity and encapsulation) to distributed environments and provides object-oriented languages with some of the characteristics of open systems (Agha & Hewitt, 1985). Based on these characteristics, various active object models have been proposed (Yonezawa & Tokoro, 1987), and to facilitate implementing active-object systems, several frameworks have been proposed. Actalk is one example.

When Actalk is used to model and realize the MAS, there still exist the following shortcomings:

- Active object is not an autonomous agent. It only manifests the procedural actions.
- Although active object can communicate, they do not own the ability to reduce the decision to communicate or order other active objects.
- If one active object has not received the information from other active objects. It is still in none-active state.

In order to overcome the shortcomings mentioned above, the concept of active object has been proposed and a general agent framework (Shoham, 1993) (Maruichi et al., 1990). An universal agent architecture has also been proposed so as to fulfil the modelling requirements of MAS (Gasser & Briot, 1992), which can be used to model and analyze MAS deeply. For either agent-based systems or MAS, the method mostly is on the base of active objects. So in this chapter, the concept of active object is firstly reviewed quickly.

2.1.2 ACTALK (Guessoum & Briot, 1999)

One of the typical active objects is Actalk. Actalk is a framework for implementing and computing various active-object models into a single programming environment based on Smalltalk, which is an object-oriented programming language. Actalk implements asynchronism, a basic principle of active-object languages, by queuing the received messages into a mailbox, thus dissociating message reception from interpretation. In Actalk, an active object is composed of three component classes (see Fig. 1), which are instances of the classes.

- Address encapsulates the active object's mailbox. It defines how to receive and queue messages for later interpretation.
- Activity represents the active object's internal activity and provides autonomy to the actor. It has a Smalltalk process and continuously removes messages from the mailbox, and the behavior component interprets the messages.
- ActiveObject represents the active object's behavior—that is, how individual messages are interpreted.

To build an active object with Actalk, we must describe its behavior as a standard Smalltalk (OOCP) object. The active object using that behavior is created by sending the message active to the behavior:

```
active
"Creates an active object with self as corresponding behavior"
^self activity: self activityClass address: self addressClass
```

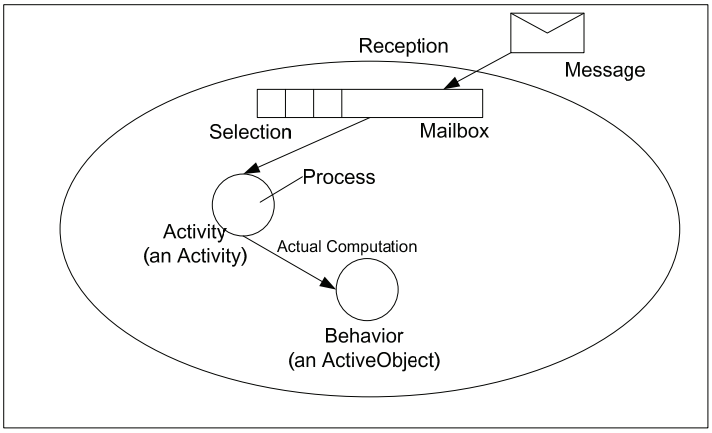


Figure 1. Components of an Actalk active object

The activityClass and addressClass methods represent the default component classes for creating the activity and address components (along the *factory method* design pattern). To configure the framework of Actalk means to define the components of its sub-classes. That's to say, it lets users to define special active object models. So Actalk is the basis to model agent based systems or MAS.

2.2 Timed Object-oriented Petri net (TOPN)

Formally TOPN is a four-tuple (OIP, ION, DD, SI), where (OIP,ION,DD) is an ordinary object Petri net-"HOONet" (Hong & Bae, 2000) and SI associates a static (firing) temporal interval SI: $\{o\}\rightarrow[a, b]$ with each object o, where a and b are rationals in the range $0\leq a\leq b\leq+\infty$, with $b\neq+\infty$. The four parts in TOPN have different function roles. Object identification place (OIP) is a unique identifier of a class. Internal timed object net (ION) is a net to depict the behaviors (methods) of a class. Data dictionary (DD) declares the attributes of a class in TOPN. And static time interval function (SI) binds the temporal knowledge of a class in TOPN. There are two kinds of places in TOPN. They are common places (represented as circles with thin prim) and abstract places (represented as circles with bold prim). Abstract places are also associated with a static time interval. Because at this situation, abstract places represent not only firing conditions, but also the objects with their own behaviors. So, abstract places (TABP) in TOPN also need to be associated with time intervals. One problem to be emphasized is that the tokens in abstract places need to have two colors at least. Before the internal behaviors of an abstract place object are fired, the color of tokens in it is one color (represented as hollow token in this paper). However, after fired, the color becomes the other one (represented as liquid token in this paper). At this time, for the following transitions, it is just actually enabled. There are three kinds of transitions in TOPN. The

timed primitive transition (represented as rectangles with thin prim) (TPIT), timed abstract transition (represented as rectangles with bold prim) (TABT) and timed communication transition (represented as rectangles with double thin prim) (TCOT).

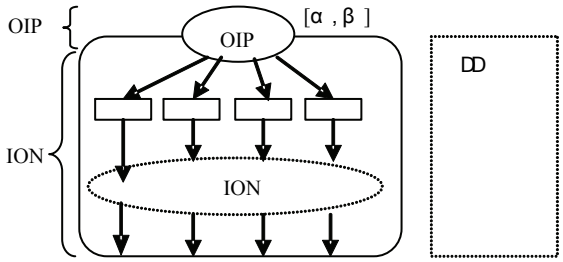
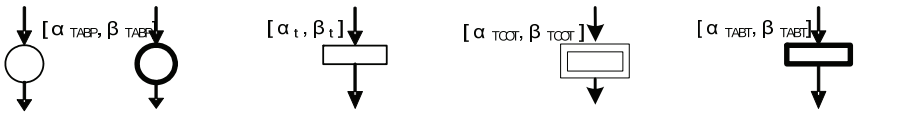


Figure 2. The General Structure of TOPN



(a) Basic Place (b) Abstract Place (c) Common Transition (d) Communication Transition (e) Abstract Transition

Figure 3. Places and Transitions in TOPN

Static time intervals change the behavior of TOPN just similar to a time Petri net in the following way. If an object o with SI(o)=[a, b] becomes enabled at time I_0, then the object o must be fired in the time interval [I_0+a, I_0+b], unless it becomes disabled by the removal of tokens from some input place in the meantime. The static earliest firing time of the object o is a; the static latest firing time of o is b; the dynamic earliest firing time (EFT) of t is I_0+a; the dynamic latest firing time (LFT) of t is I_0+b; the dynamic firing interval of t is [I_0+a, I_0+b].

The state of TOPN (Extended States-"ES") is a 3-tuple, where ES=(M, I, path) consists of a marking M, a firing interval vector I and an execution path. According to the initial marking M_0 and the firing rules mentioned above, the following marking at any time can be calculated. The vector--"I" is composed of the temporal intervals of enabled transitions and TABPs, which are to be fired in the following states. The dimension of I equals to the number of enabled transitions and TABPs at the current state. The firing interval of every enabled transition or TABP can be got according to the calculation formula of EFT and LFT in TOPN (Xu & Jia, 2006).

For enabling rules in TOPN, two different situations exist. A transition t in TOPN is said to be enabled at the current state (M, I, path), if each input place p of t contains at least the number of solid tokens equal to the weight of the directed arcs connecting p to t in the marking M. If the TABP object is marked with a hollow token, it is enabled. At this time, its ION is enabled. After the ION has been fired, the tokens in TABP are changed into solid ones.

An object o is said to be fireable in state (M, I, path) if it is enabled, and if it is legal to fire o next. This will be true if and only if the EFT of o is less than or equal to the LFT of all other enabled transitions. Of course, even with strong time semantics, o's being fireable in state (M, I, path) does not necessarily mean that t will fire in the time interval I.

2.3 Fuzzy Timed Object-oriented Petri net (FTOPN)

Similar to FTPN (Xu & Jia, 2005-1), fuzzy set concepts are introduced into TOPN (Xu & Jia, 2006). Then FTOPN is proposed, which can describe fuzzy timing effect in dynamic systems.

Definition 1: FTOPN is a six-tuple, FTOPN= (OIP, ION, DD, SI, R, I) where

1. Suppose OIP=(oip, pid, M_0, status), where oip, pid, M_0 and status are the same as those in HOONet (Hong & Bae, 2000) and TOPN (Xu & Jia, 2006).
   - oip is a variable for the unique name of a FTOPN.
   - pid is a unique process identifier to distinguish multiple instances of a class, which contains return address.
   - M_0 is the function that gives initial token distributions of this specific value to OIP.
   - status is a flag variable to specify the state of OIP.
2. ION is the internal net structure of FTOPN to be defined in the following. It is a variant CPN that describes the changes in the values of attributes and the behaviors of methods in FTOPN.
3. DD formally defines the variables, token types and functions (methods) just like those in HOONet (Hong & Bae, 2000) and TOPN (Xu & Jia, 2006).
4. SI is a static time interval binding function, SI: {OIP}→Q*, where Q* is a set of time intervals.
5. R: {OIP} → r, where r is a specific threshold.
6. I is a function of the time v. It evaluates the resulting degree of the abstract object firing. □

Definition 2: An internal object net structure of TOPN, ION = (P,T,A,K,N,G,E,F,M_0)

1. P and T are finite sets of places and transitions with time restricting conditions attached respectively.
2. A is a finite set of arcs such that P∩T=P∩A=T∩A=Φ.
3. K is a function mapping from P to a set of token types declared in DD.
4. N, G, and E mean the functions of nodes, guards, and arc expressions, respectively. The results of these functions are the additional condition to restrict the firing of transitions. So they are also called additional restricting conditions.
5. F is a special arc from any transitions to OIP, and notated as a body frame of ION.
6. M_0 is a function giving an initial marking to any place the same as those in HOONet (Hong & Bae, 2000) and TOPN (Xu & Jia, 2006). □

Definition 3: A set of places in TOPN is defined as P=PIP□TABP, where

1. Primary place PIP is a three-tuple: PIP =(P,R,I), where
   - P is the set of common places similar to those in PN (Murata, 1989) (Peterson, 1991).
2. Timed abstract place (TABP) is a six-tuple: TABP= TABP(pn, refine state, action, SI, R, I), where
   - pn is the identifier of the abstract timed place.
   - refine state is a flag variable denoting whether this abstract place has been refined or not.
   - action is the static reaction imitating the internal behavior of this abstract place.
   - SI, R and I are the same as those in Definition 1.

Definition 4: A set of transitions in TOPN can be defined as T= TPIT□TABT□TCOT, where

1. Timed primitive transition TPIT = TPIT (BAT, SI), where
   - BAT is the set of common transitions.

2. Timed abstract transition TABT= TABT (tn, refine state, action, SI), where
   - tn is the name of this TABT.
3. Timed communication transition TCOT=TCOT (tn, target, comm type, action, SI).
   - tn is the name of TCOT.
   - target is a flag variable denoting whether the behavior of this TCOT has been modeled or not. If target="Yes", it has been modeled. Otherwise, if target="No", it has not been modeled yet.
   - comm type is a flag variable denoting the communication type. If comm type ="SYNC", then the communication transition is synchronous one. Otherwise, if comm type="ASYN", it is an asynchronous communication transition.
4. SI is the same as that in Definition 1.
5. refine state and action are the same as those in Definition 3.

Similar to those in FTPN (Xu & Jia, 2005-1), the object t fires if the foregoing objects come with a nonzero marking of the tokens; the level of firing is inherently continuous. The level of firing (z(v)) assuming values in the unit interval is governed by the following expression:

$$z(v) = (\overset{n}{\underset{i=1}{T}}(r_i \rightarrow x_i(v'))\, sw_i)\, t\mathrm{I}(v) \tag{1}$$

where T (or t) denotes a t-norm while "s" stands for any s-norm. "v" is the time instant immediately following v'. More specifically, x_i(v) denotes a level of marking of the i^{th} place. The weight w_i is used to quantify an input coming from the i^{th} place. The threshold r_i expresses an extent to which the corresponding place's marking contributes to the firing of the transition. The implication operator ($\rightarrow$) expresses a requirement that a transition fires if the level of tokens exceeds a specific threshold (quantified here by r_i).

Once the transition has been fired, the input places involved in this firing modify their markings that is governed by the expression

$$x_i(v)=x_i(v')t(1-z(v)) \tag{2}$$

(Note that the reduction in the level of marking depends upon the intensity of the firing of the corresponding transition, z(v).) Owing to the t-norm being used in the above expression, the marking of the input place gets lowered. The output place increases its level of tokens following the expression:

$$y(v)=y(v')sz(v) \tag{3}$$

The s-norm is used to aggregate the level of firing of the transition with the actual level of tokens at this output place. This way of aggregation makes the marking of the output place increase.

The FTOPN model directly generalizes the Boolean case of TOPN and OPN. In other words, if x_i(v) and w_i assume values in {0, 1} then the rules governing the behavior of the net are the same as those encountered in TOPN.

3. Agent Objects and Fuzzy Timed Agent Based Petri Nets

The *active object* concept (Guessoum & Briot, 1999) has been proposed to describe a set of entities that cooperate and communicate through message passing. To facilitate implementing *active object* systems, several frameworks have been proposed. ACTALK is

one of the typical examples. ACTALK is a framework for implementing and computing various *active object* models into one object-oriented language realization. ACTALK implements asynchronism, a basic principle of *active object* languages, by queuing the received messages into a mailbox, thus dissociating message reception from interpretation. In ACTALK, an *active object* is composed of three component classes: *address*, *activity* and *activeObject* (Guessoum & Briot, 1999).

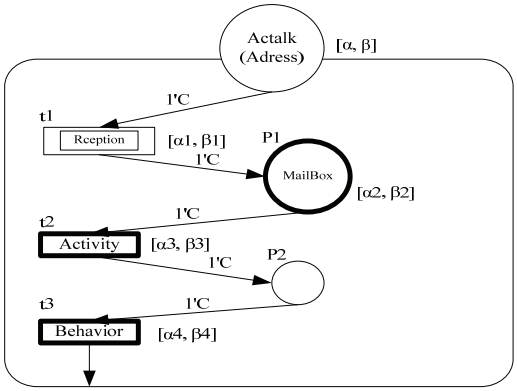


Figure 4. The FTOPN Model of ACTALK

ACTALK model is the basis of constructing *active object* models. However, *active object* model is the basis of constructing multi-agent system model or agent-based system model. So, as the modeling basis, ACTALK has been extended to different kinds of high-level agent models. Because of this, ACTALK is modeled in Fig.4 by FTOPN.

In Fig.4, OIP is the describer of the ACTALK model and also represents as the communication address. One communication transition is used to represent as the behavior of message reception. According to the communication requirements, it may be synchronous or asynchronous. If the message has been received, it will be stored in the corresponding mail box, which is one first in and first out queue. If the message has been received, the next transition will be enabled immediately. So mail box is modeled as abstract place object in FTAPN. If there are messages in the mail box, the following transition will be enabled and fired. After the following responding *activity* completes, some *active behavior* will be conducted according to the message.

Fig.4 has described the ACTALK model based on FTOPN on the macroscopical level. The detailed definition or realization of the object "*Activity*" and "*Behavior*" can be defined by FTOPN in its parent objects in the lower level. The FTOPN model of ACTALK can be used as the basic agent object to model agent based systems. That is to say, if the agent based model—ACTALK model is used in the usual FTOPN modeling procedure, FTOPN has been extended to *agent based modeling methodology*. So it is called *fuzzy timed agent based Petri net* (*FTAPN*).

4. Learning in Fuzzy Timed Agent Based Petri Nets

The parameters of FTAPN are always given beforehand. In general, however, these parameters may not be available and need to be estimated just like those in FTPN (Xu & Jia,

2005-1). The estimation is conducted on the base of some experimental data concerning marking of input and output places. The marking of the places is provided as a discrete time series. More specifically we consider that the marking of the output place(s) is treated as a collection of target values to be followed during the training process. As a matter of fact, the learning is carried out in a supervised mode returning to these ***target*** data.
The connections of the FTOPN (namely weights w_i and thresholds r_i) as well as the time decay factors α_i are optimized (or trained) so that a given performance index Q becomes minimized. The training data set consists of (a) initial marking of the input places $x_i(0),\ldots,x_n(0)$ and (b) target values—markings of the output place that are given in a sequence of discrete time moments, that is target(0), target(1),…, target(K).
In FTAPN, the performance index Q under discussion assumes the form of Eq.(4).

$$Q=\sum_{k=1}^{K}(t\arg et(k)-y(k))^2 \tag{4}$$

where the summation is taken over all time instants (k =1, 2,… , K).
The crux of the training in FTOPN models follows the general update formula in Eq.(5) being applied to the parameters:

$$\mathbf{param}(\text{iter}+1)=\mathbf{param}(\text{iter})-\gamma\nabla_{\mathbf{param}}Q \tag{5}$$

where γ is a learning rate and $\nabla_{\mathbf{param}}Q$ denotes a gradient of the performance index taken with respect to all parameters of the net (here we use a notation **param** to embrace all parameters in FTOPN to be trained).
In the training of FTOPN models, marking of the input places is updated according to Eq.(6):

$$\tilde{x_i} = x_i(0)T_i(k) \tag{6}$$

where $T_i(k)$ is the temporal decay. And $T_i(k)$ complies with the form in Eq.(7). In what follows, the temporal decay is modeled by an exponential function,

$$T_i(k)=\begin{cases}\exp(-\alpha_i(k-k_i)) & if \quad k>k_i,\\ 0 & others\end{cases} \tag{7}$$

The level of firing of the place can be computed as Eq.(8):

$$z=(\overset{n}{\underset{i=1}{T}}((r_i\rightarrow\tilde{x_i})sw_i)) \tag{8}$$

The successive level of tokens at the output place and input places can be calculated as that in Eq.(9):

$$y(k) = y(k\text{-}1)sz,\ x_i(k) = x_i(k\text{-}1)t(1\text{-}z) \tag{9}$$

We assume that the initial marking of the output place y(0) is equal to zero, y(0)=0. The derivatives of the weights w_i are computed as the form in Eq.(9):

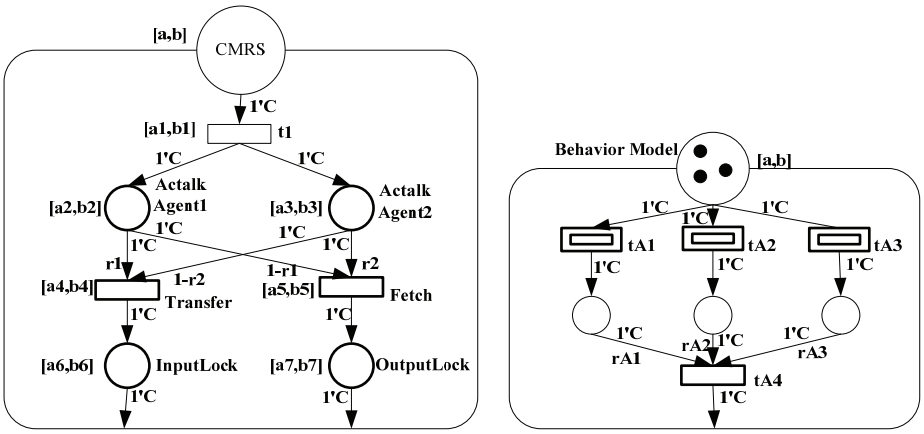
$$\frac{\partial}{\partial w_i}(t\arg et(k)-y(k))^2=-2(t\arg et(k)-y(k)\frac{\partial y(k)}{\partial w_i}) \tag{10}$$

where i=1,2,…, n. Note that y(k+1)=y(k)sz(k).

5. A Modeling Example

5.1 A CMRS Model

In the etching tools, usually there is a Brooks Marathon Express (MX) (Lee & Lee, 2004) CMRS platform made up of two transferring robots. These two cooperative robots are up to complete transferring one unprocessed wafer from the input lock to the chamber and fetch the processed wafer to the output lock. Any robot can be used to complete the transferring task at any time. If one robot is up to transfer one new wafer, the other will conduct the other fetching task. They will not conflict with each other. Fig. 5 depicts this CMRS FTAPN model, where two agent objects (ACTALK) are used to represent these two cooperative robots.



(a) The Agent Based FTAPN Model (b) The Behavior Model in Every Agent

Figure 5. The FTAPN Model

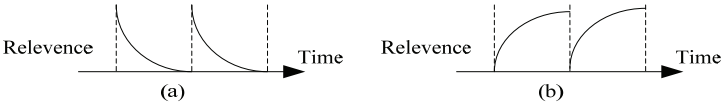


Figure 6. The Relevance

Fig.5 (a) has depicted the whole FTAPN model. The agent object—"ACTALK" is used to represent every robot model. Different thresholds are used to represent the firing level of the behavior conducted by the corresponding robot (agent). They also satisfy the unitary requirements and change according to the fuzzy decision in the behavior of every agent in Fig.5 (b). In the model of Fig.5 (b), three communication transition objects are used to represent the behavior for getting different kinds of system states. These states include the state of the other robot, its own goal and its current state, which can be required by the conductions of the communication transitions tA1, tA2 and tA3. When one condition has been got, the following place will be marked. In order to make control decisions (transition object tA4) in time, all of these state parameters are required in the prescriptive time interval. However, the parameter arrival times complies with the rule in Fig.5 (a). The other two kinds of information comply with that in Fig.5 (b). After the decision, a new decision

command with the conduction probability will be sent in this relative interval and it also affects which behavior (transfer or fetch) will be conducted by updating the threshold in Fig.5 (a).

5.2 Application Analysis

Table.1 summarizes the main features of FTAPN and contrast these with the structures with which the proposed structures have a lot in common, namely MAS and FTOPN. It becomes apparent that FTAPN combines the advantages of both FTOPN in terms of their learning abilities and the glass-style of processing (and architectures) of MAS with the autonomy.

Characteristics	MAS	FTOPN	FTAPN
Learning Aspects	Significant dynamic learning abilities. Dynamic learning and decision abilities are supported in every autonomous agent.	Significant learning abilities. Distributed learning (training) abilities are supported in different independent objects on various system model levels.	Significant dynamic learning abilities. Distributed dynamic learning and decision abilities are supported in every autonomous agent.
Knowledge Representation Aspects	Transparent knowledge representation (glass box processing style) the problem (its specification) is mapped directly onto the topology of the agent model. Additionally, agents deliver an essential feature of continuity required to cope with dynamic changes encountered in a vast array of problems (including autonomous decision tasks)	Glass Box Style (Transparent Knowledge Representation) and Black Box Processing is supported at the same time. The problem (its specification) is mapped directly onto the topology of FTOPN. Knowledge representation granularity reconfiguration reacts on the reduction of model size and complexity.	Glass Box Processing Style and Black Box Processing style are all supported. The problem (its specification) is mapped directly onto the topology of FTAPN, which can not only represent dynamic knowledge, but also deal with dynamic changes with well-defined semantics of agent objects, places, transitions, fuzzy and temporal knowledge.

Table 1. MAS, FTOPN and FTAPN: a comparative analysis

5.3 Application Aspects of FTAPN

Owing to the nature of the facet of temporal knowledge, fuzzy sets and object-oriented concepts in this extension of PN, they become viable models in a wide range of engineering problem augmenting the already existing high level Petri nets, cf. (Hong & Bae, 2000) (Xu & Jia, 2005-1). Two main and commonly categories of models are worth elaborating here.

5.3.1 Models of Multi-agent Systems

The multi-agent paradigm and subsequently a variety of models are omnipresent in a number of areas. In a nutshell, in spite of the existing variety of improved models, it still lacks of a powerful modeling method, which can bridge the gap between model and practical implementations. Petri nets come with objects, temporal knowledge and fuzzy sets can use active objects to model generic agents with situatedness, autonomy and flexible. This helps us to use the object to reduce the complexity of MAS systems and the dynamic learning and decision to support the autonomy of agents.

5.3.2 Models of Complex Real-time Systems

In models of complex real-time systems as usually encountered in industrial practice, the scale of the system module may be too complicated to be analyzed and the readings of different system state or sensors may be available at different time. The former may lead to the state explosion, while the latter needs adjustment of relevance of the information gathered at different time scales. The object models with temporal information degradation (aging) helps to abstract complicated model and quantify the confidence of the inferred results.

6. Conclusions and Future Work

CMRS is a kind of usual manufacturing equipments in manufacturing industries. In order to model, analyze and simulate this kind of systems, this chapter proposes fuzzy timed agent based Petri net (FTAPN) on the base of FTOPN (Xu & Jia, 2005-1) and FTPN (Pedrycz & Camargo, 2003). In FTAPN, one of the active objects— ACTALK is introduced and used as the basic agent object to model CMRS, which is a typical MAS. Every abstract object in FTOPN can be trained and reduced independently according to the modeling and analysis requirements for OO concepts supported in FTOPN. The validity of this modeling method has been used to model Brooks CMRS platform in etching tools. The FTAPN can not only model complex MAS, but also be refined into the object-oriented implementation easily. It has provided a methodology to overcome the development problems in agent-oriented software engineering. At the same time, it can also be regarded as a conceptual and practical artificial intelligence (AI) tool for integrating MAS into the mainstream practice of software development.

State analysis needs to be studied in the future. An extended State Graph (Xu & Jia, 2005-2) has been proposed to analyze the state change of TOPN models. With the temporal fuzzy sets introduced into FTAPN, the certainty factor about object firing (state changing) needs to be considered in the state analysis.

7. Acknowledgement

This work is jointly supported by the National Nature Science Foundation of China (Grant No: 60405011, 60575057) and the China Postdoctoral Foundation for China Postdoctoral Science Fund (Grant No: 20040350078) in China.

8. References

Agha, G., Hewitt, C.(1985). Concurrent Programming Using Actors: Exploiting Large Scale Parallelism, Lecture Notes in Computer Science, No. 206, S.N. Maheshwari,ed., Springer-Verlag, New York, 1985, pp.19–41.

Battiston, E., Cindio, F.D., Mauri, G.(1988). *OBJSA Nets: a class of high-level nets having objects as domains,* Proceedings of APN'88, Lecture Notes in Computer Science, Vol.340, pp.20-43

Briot, J.P. (1996). *An Experiment in Classification and Specialization of Synchronization Schemes*, Lecture Notes in Computer Science, No. 1107, pp. 227–249.

Cao, Y.U.; Fukunaga, A.S.; Kahng, A.B.; Meng, F. (1997); *Cooperative Mobile Robotics: Antecedents and Directions*, Autonomous Robots, Vol.4, pp.7–27

Castelfranchi, C. (1995).*A Point Missed in Multi-Agent*, DAI and HCI, Lecture Notes in Artificial Intelligence, No. 890, pp. 49–62

Chainbi, W.(2004) ; *Multi-agent systems: a Petri net with objects based approach*. In: Proceedings of IEEE/WIC/ACM International Conference on Intelligent Agent Technology, Beijing, pp.429 – 432

Gasser,L.(1992). *An Overview of DAI*, Distributed Artificial Intelligence, N.A. Avouris and L. Gasser, eds., Kluwer Academic, Boston, 1992, pp. 1–25.

Gasser, L, Briot,J.P.(1992). *Object-Oriented Concurrent Programming and Distributed Artificial Intelligence*, Distributed Artificial Intelligence, N.A. Avouris and L. Gasser, eds., Kluwer Academic, Boston, 1992, pp. 81–108.

Guessoum, Z., Briot, J.P.(1999). *From active objects to autonomous agents,* IEEE Concurrency, Vol.7, No.3, pp. 68 – 76

Hong, J.E., Bae, D.H.(2000). *Software Modeling And Analysis Using a Hierarchical Object-oriented Petri net,* Information Sciences, Vol.130, pp.133-164

Jennings, N.R., Sycara, K., Wooldridge, M.(1998). *A Roadmap of Agent Research and Development,* Autonomous Agents and Multi-Agent Systems, Vol.1, pp.7–38

Lee , J.H., Lee, T.E.(2004). SECAM: *A Supervisory Equipment Control Application Model for Integrated Semiconductor Manufacturing Equipment*, IEEE Robotics & Automation Magazine,Vol. 11, No. 1, pp.41 - 58

Maruichi, T., Ichikawa, M., and Tokoro, M. (1990). *Decentralized AI*, Modeling Autonomous Agents and Their Groups, Elsevier Science, Amsterdam, 1990, pp. 215–134.

Murata, T.(1989). Petri Nets: Properties, Analysis and Applications. *Proceedings of IEEE,* Vol.77, No.4, (April 1989) pp.541-580

Pedrycz, W., Camargo, H.(2003). *Fuzzy timed Petri nets*, Fuzzy Sets and Systems, Vol.140, No. 2, pp. 301-330

Peterson, J.L.(1991). *Petri Net Theory and the Modeling of Systems*, Prentice-Hall, N.Y., USA

Shoham, Y. (1993). *Agent-Oriented Programming*, Artificial Intelligence, Vol. 60, No.1, pp. 139–159.

Xu, H., Jia, P.F.(2005-1). *Fuzzy Timed Object-Oriented Petri Net,* Artificial Intelligence Applications and Innovations (Proceedings of AIAI2005), Springer, pp.148-160, N.Y., USA

Xu, H., Jia, P.F.(2005-2). *Timed Hierarchical Object-Oriented Petri Net*, GESTS International Transactions on Computer Science and Engineering, vol. 24, No.1, pp.65-76

Xu, H., Jia, P.F.(2006). *Timed Hierarchical Object-Oriented Petri Net-Part I: Basic Concepts and Reachability Analysis,* Lecture Notes In Artificial Intelligence (Proceedings of RSKT2006), Vol. 4062, pp.727-734

Yonezawa, A., Tokoro. M.(1987), Object-Oriented Concurrent Programming, A. Yonezawa and M. Tokoro, eds., The MIT Press, Cambrige, Mass., 1987.

14

Cooperative Control of Multiple Biomimetic Robotic Fish

Junzhi Yu[1], Min Tan[1] and Long Wang[2]
[1] *Lab. of Complex Systems and Intelligence Science, Institute of Automation Chinese Academy of Sciences*
[2] *Department of Mechanics and Space Technologies, College of Engineering Peking University China*

1. Introduction

There has been a spurt of interest in recent years in the area of group coordination and cooperative control among different research communities, involving biology, robotics, artificial intelligence, advanced control, sensor network, etc (Ögren et al., 2004; Kumar et al., 2005; Pettersen et al., 2006). As we know, fish shoals, bird flocks, etc exhibit typical aggregation behaviours, in which collective motions are employed to achieve useful tasks, e.g., avoiding predators, capturing prey, and breeding offspring. Similarly, when coordinating in unstructured or dynamic environments, it is possible that a team of relatively simple and cheap agents are capable of accomplishing complex tasks that exceed the capabilities of one single agent. In the robotic context, multi-robot systems presenting as distributed solutions have advantages such as increasing robustness to unexpected disturbances, fault-tolerance, thanks to redundancy, self-adaptation, and self-organization. Thus, applications of multi-robot systems are associated with a large group of autonomously functioning vehicles in the air, on land or sea or underwater, to jointly perform tasks such as demining operations, environmental surveillance, object transportation, search and rescue, and so forth (Kumar et al., 2002; Rabbath et al., 2007 ; Wang et al., 2007 ; Zhang et al., 2007).

Rapidly growing interests in cooperation and coordination of multi-robot systems have brought a lot of real-world applications as stated above. However, most significant efforts are devoted to ground and air based cooperation issues, and cooperative studies on underwater or surface vehicle are relatively few and immature (Rabbath et al., 2007). In particular, research on a group of robotic fish has not yet been implemented. The primary objective of this chapter is to build an artifical multi-fish system mimicking cooperative mechanism of fish flock, which provides a platform to test and verify the algorithms and strategies for cooperation of multiple underwater robots.

Although the topic of underwater biorobotics is not new, and there is a long history, the self-propelled, fish-like robots were indeed created in the 1990s (Triantafyllou & Triantafyllou, 1995; Triantafillou et al., 2004; Lauder et al., 2007a). The fish-like robot, i.e., robotic fish, as a marriage of biomechanism with engineering technology, is a cross-disciplinary subject

which mainly involves hydrodynamics based control and robotic technology. It is well known that fish have evolved to become the best swimmer in nature during millions of years' nature selection. They can easily achieve very high propulsive efficiency and maneuverability with little loss of stability through integrating multiple control surfaces including body, fins, and tail (Yu et al., 2004; Bandyopadhyay, 2005; Lauder et al., 2007a). As we expect, mimicking these biological systems will offer innovative, bio-inspired solutions to improving and even updating conventional underwater vehicles equipped with thrusters, which potentially bring increased performance in acceleration, speed, efficiency, maneuverability, stealth, etc (Sfakiotakis et al., 1999; Anderson & Chhabra, 2002; Triantafillou et al., 2004). Since the vast majority of the work to date on robotic fish has focused on the hydrodynamic mechanism of fish-like swimming, fish-like vehicles design, and control algorithms for fish-like locomotion, cooperative control of multiple robotic fish is surely an important subject for future real-world applications such as in military detection, undersea exploration, and management of water pollution. Moreover, the robotic fish based cooperation research will provide a useful reference for exploring the self-organizing mechanism of fish school, and understanding the collective behaviors by using only local information and interactions.

Despite fruitful work on cooperative control of multi-robot systems, most of these reference solutions can seldom be applied to the multi-fish system directly due to the unique locomotion mode of the robotic fish as well as the complex aquatic environment with many different sources of disturbance. Under such a dynamic environment, it is very difficult to design controllers that will guarantee system performance even in a local sense. So, the artificial multi-fish system, which is referred to as Multiple Robotic Fish cooperation System (MRFS), is established via behaviour-based approach. Specifically, using top-down design method, we propose a hierarchical architecture that comprises five levels: task level, role level, behavior level, action level, and controller level, to formalize the processes from task decomposition, role assignments, and control performance assessment. Furthermore, a vision-based closed-loop experimental system for multiple robotic fish is set up, where an improved parallel visual tracking method is formulated within a framework synthesizing features of fish features and surrounding disturbance. Since the robotic fish we employed have no ability of self-positioning, thus, an overhead, global vision subsystem is adopted to acquire information of the environment and the states of the fish. So our approach should be basically categorized into the centralized control. Then, rapid and accurate multi-fish tracking is crucial for decision-making. Currently, for the lab-based MRFS, a team of robotic fish (as large as eight) can perform some given tasks (primarily some challenging games) in a water tank with a dimension of 3.3 m × 2.3 m × 0.7 m (length × width × depth).

From a control engineering point of view, the MRFS can be considered as an integrated guidance packages that take into consideration the kinematics and the dynamics of the fish swimming, which functionally generates the motion commands for each fish. In this chapter, we provide a full development description for the MRFS, which is built on the basis of a series of free-swimming, radio-controlled, multi-link fish-like robots. There exist three salient features in MRFS:

- The custom-built fish-like robots are moderate sized, flexible, and easy to be controlled in a lab-based experiment configuration;
- The testbed is applicable to different cooperative tasks, and hence can be viewed a general one;

- High-level tasks within the hierarchical framework are ultimately decomposed into two primitive motion controllers, i.e., speed controller and orientation controller.

The rest of the chapter is organized as follows. An improved approach to design a multi-link robotic fish is outlined in Section 2. A vision-based subsystem to aid in identifying the fish and their surroundings is offered in Section 3. A hierarchical framework for coordinated control is presented in Section 4. The implementation of MRFS, including experimental results and discussions, are provided in Section 5 and Section 6. Section 7 contains the concluding remarks.

2. Design and Implementation of the robotic fish

In this section, we briefly present an overall design of a multi-link biomimetic fish prototype, describing its propulsive mechanism, mechatronic design, and motion control.

2.1 Multi-link based propulsive configuration

The rich variety of mechanisms employed by swimming organisms has long been an inspiration for biologists and engineers. Undulation of the axial structure, i.e., undulatory swimming, is the most general form of aquatic vertebrate locomotion. Regarding a fish as a hydrodynamic swimming machine, the central part of the fish body is support by jointed skeletal elements driven by muscle motors, while the tail acts as a propeller (Sfakiotakis et al., 1999; Lauder & Madden, 2007b). As we mentioned previously, self-propelled robotic fish capable of executing programmed motions not only provide a significant avenue for understanding locomotor questions in aquatic environment, but also enable estimating various conceptual designs for aquatic propulsion oriented robotic models. Attempts to replicate this delicate musculoskeletal structure are still not very successful. In general, there are two methods for designing fish-like propulsive mechanism in part (Yu et al., 2007). One is the hyperredundant, discrete body design, where a "multi-motor-multi-joint" structure is often adopted. Another is continuous body design, whose actuation is based on new-type smart materials such as polymers, elastomers and shape memory alloys, or on a special "single-motor-multi-joint" mechatronic layout. Because of the immaturity and delicacy of the continuous body design, the overall performance of the developed robotic fish is usually inferior to the discrete design scheme. With the purpose of developing a high-performance, self-propelled robotic fish with conventional actuation modes, we choose the multi-motor-multi-joint scheme. Particularly, the position-controlled servomotors are used to drive the multi-link fish. For more information on the continuous body design, the reader is referred to Bandyopadhyay (2004).

As biologists suggest, thrust is generated by momentum transfer to the surrounding water as a wave of bending passing down the deforming body during fish swimming. For convenience of describing these lateral body motions, strong emphases are laid on kinematic and anatomical data of vertebral column and tail. Typically, a propulsive wave form (hereafter referred to as body wave) that results from the progression of muscular contraction from head to tail is distilled to characterize the movement of the midline. Videler & Hess (1984) used a Fourier series to describe the body wave:

$$y_{body}(x,t) \simeq \sum_{j=1,3,5} \{a_j(x)\cos j\omega t + b_j(x)\sin j\omega t\} \tag{1}$$

where y_{body} represents the transverse displacement of the fish body, x denotes the displacement along the main axis, a_j and b_j denotes the Fourier coefficients derived from the digitized data. Barrett (1996) chose a propagating sine wave equation (2) as the reference body wave in the construction of a robotic tuna, which closely represents that of motion of a live tuna.

$$y_{body}(x,t) = (c_1 x + c_2 x^2)\sin(kx + \omega t) \tag{2}$$

where k stands for the body wave number ($k=2\pi/\lambda$), λ is the body wave length, c_1 is the linear wave amplitude envelope, c_2 is the quadratic wave amplitude envelope, and ω is the body wave frequency ($\omega=2\pi f=2\pi/T$). Specially, like the Fourier coefficients a_j and b_j, the adjustable parameters c_1 and c_2 are choosen according to the collected data on realistic fish movements.

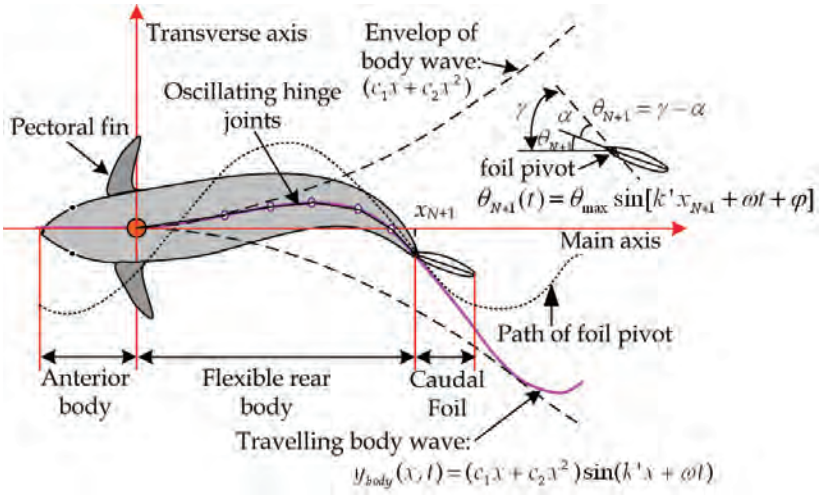


Figure 1. A simplified propulsive model for a multi-link robotic fish

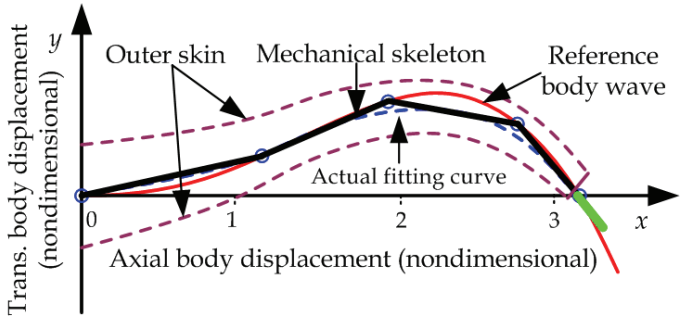


Figure 2. Actual fitting curve versus reference body wave in the multi-link robotic fish model

Fish is composed of tens of vertebra, as is observed, and each vertebra may be regarded as a miniature joint. The oscillatory part of the robotic fish, in this sense, could be discretely designed as a multi-link (or N-link) mechanism which consists of several oscillating hinge

joints actuated by motors. But it is hard to generate exactly a fish-like smooth wave with limited joints. How to use restricted joints to approximate the body waves exhibited in real fish is then a great challenge for robotics researchers. In our previous work, this problem is reduced to a numerical fitting issue of using a chain of links to approximate a discretized, spatial- and time-varying body wave (Fig. 1). During numerical operation, equation (2) is rewritten as (3) preserving the original body wave decomposed into two parts: the time-independent wave sequence $y_{body}(x, i)$ (i = 0, 1, . . . ,M − 1) in an oscillation cycle and the time-dependent oscillation frequency f which is regulated by the changing time interval between $y_{body}(x, i)$ and $y_{body}(x, i+1)$ when the robotic fish swims.

$$y_{body}(x,i) = (c_1 x + c_2 x^2)\sin(kx - \tfrac{2\pi}{M} i) \qquad i \in [0, M-1] \tag{3}$$

where i denotes the serial number in an oscillation cycle and M indicates the resolution of the discrete body wave. For more details of link-based body wave fitting we refer the reader to Yu et al. (2004).

In the above link-based body wave fitting, a precondition is imposed that the reference body wave (2) taken from the fast-swimming fish has an advantaged hydrodynamic and kinematic performance. We remark that this may not be true for various fishes in nature. When constructing a robotic prototype, a waterproof, outer skin is employed to envelop the multi-link based metal skeleton. The function of the elastic outer skin is to offer a smooth shape and reduce form drag. However, as shown in Fig. 2, an accompanying side effect that a relative difference between the actual fitting curve and the reference body wave arises after this elastic transition. In such a case, we can not ensure all points of link fall into the reference body wave in an oscillation cycle, but make each point in the multi-link move according to the reference body wave as closely as possible. Considering both ichthyologic characteristics and mechatronic constraints, an improved cyclic variable method is proposed to minimize the enveloped area between moving links and the reference body wave by searching the optimal link-length ratio ($l_1 : l_2 : \ldots : l_N$). The comparative results, before and after the optimization, have partly demonstrated satisfactory performance of this link-length ratio optimization in forward locomotion, turning maneuvers, and energy savings. The optimal link-length ratios for three-link and four-link robots are 1 : 0.72 : 0.65 and 1 : 0.73 : 0.63 : 0.61, respectively, which are applied to later robotic prototypes. More detailed geometric optimization of relative link lengths for robotic fish can be found in Yu et al. (2007).

Besides multi-link flexible fish body, another fundamental design feature is caudal fin for a fish propelled by body surface and caudal fin. In the artificial fish system, the rigid anterior body is mechanically connected to the front of the multi-link flexible fish body, while the caudal fin is fixed to the lattermost link. The attached caudal fin rotates around the fin pivot in a sinusoidal fashion taking the form of equation (4).

$$\theta_{N+1}(t) = \theta_{\max}\sin(kx_{N+1} + \omega t + \varphi) \tag{4}$$

where $\theta_{N+1}(t)$ indicates the pitch angle of the caudal fin with respect to the main axis, θ_{max} the amplitude of the pitch angle, φ the phase angle between the heave and the pitch, and x_{N+1} the x-component of the position of the moving foil pivot. As a practical way, θ_{max} can be calculated as:

$$\theta_{\max} = \gamma_0 - \alpha_0 = \arctan(\frac{\partial y}{\partial x}(x_{N+1}\,|_{y_{N+1}=0}, t)) - \alpha_0 \tag{5}$$

where γ_0 denotes the angle corresponding to the slope of $y_{body}(x_{N+1},t)$ at $y_{N+1}=0$, and a_0 is the angle of attack of the caudal fin at $y_{N+1}=0$.

As summarized in Yu et al. (2005), of the most prominent characteristics associated with fish-like robots, there are four parameters that constrain the swimming performance:

- The first one is the length ratio of the fish's oscillatory part to the whole body, R_l ($0 < R_l \leq 1$). The fish will switch from a carangiform swimmer to an anguilliform one relying on different values of R_l. With the decrease of R_l, in general, the efficiency and speed of fish swimming remarkably increase, but the maneuverability reduces to a certain extent.
- The second one is the number of simplified joints (segments) in oscillatory part, N. Larger value of N, in principle, will lead to better maneuverability and redundancy, but harder construction and control of the robot.
- The third one is the link-length ratio of links in the oscillatory part, $l_1 : l_2 : \ldots : l_N$. The length of each link in the direction from nose to tail, generally speaking, is getting smaller and smaller. The oscillatory amplitude, in contrast, increases gradually and reaches its maximum at the tail peduncle of the fish.
- The last one is the shape of the caudal fin.

Taking a further step towards capturing key features of the functional design of fishes, we have developed a custom-built executive routine DFS (Digital Fish Simulation). This fish-oriented simulation platform integrating the geometric optimization of relative link lengths, is based on a WINDOWS XP operation system with a compiler of Microsoft Visual C++ 6.0. A snapshot of this execution is illustrated in Fig. 3. It mainly consists of four components: theoretical calculation, body wave configuration, curve generator, and animation. Friendly Graphical User Interface (GUI), coupled with the ability to partly mimic the kinematics and hydrodynamics of locomotion in vivo, supplies a powerful tool for understanding key points in fish based propulsion and for preliminary assessment of fish-like robotic schemes.

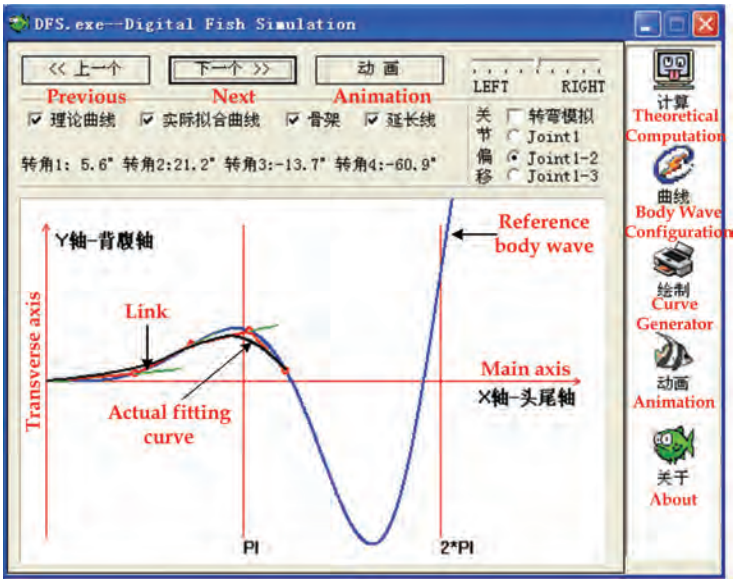


Figure 3. A snapshot of the Digital Fish Simulation platform

2.2 Mechatronic implementation of robotic fish

The fish-like devices, as addressed previously, have a huge potential for precise control of kinematics and non-biological parameters validation, which provides a self-contained test bed for designing bio-inspired underwater vehicles. A series of radio-controlled, self-propelled robotic fish, based on the simplified propulsive mechanism after optimization, have been developed. The mechanical configuration for a four-link robotic fish capable of up-down movement via a pair of artificial pectoral fins is illustrated in Fig. 4, and its drive and control architecture in Fig. 5. It can swim realistically like a fish in the water tank. Fig. 6 exhibits six types of fish-like robots, which are primarily consist of six parts:

- **Support unit** (aluminum exoskeleton + head + anterior body)
- **Actuator unit** (DC servomotors)
- **Sensor unit** (infrared, visual, ultrasonic, etc)
- **Communication unit** (wireless receiver)
- **Control unit** (microprocessor + peripherals)
- **Accessories** (batteries, waterproof skin, tail fin, etc)

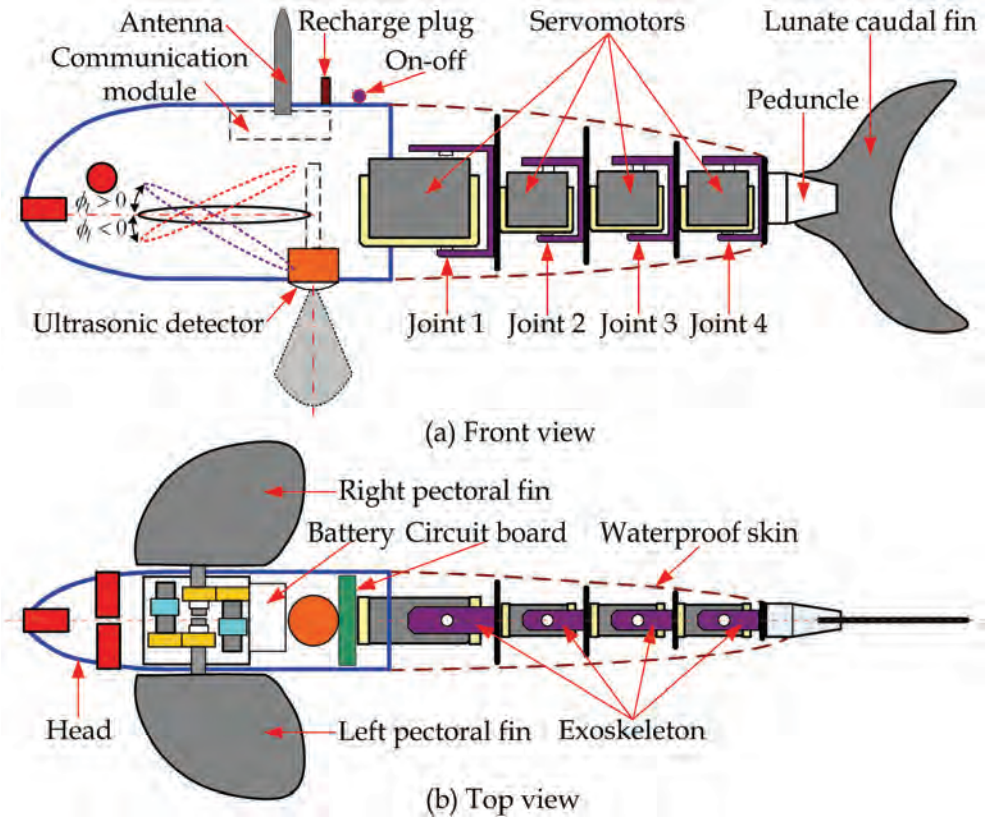


Figure 4. Mechanical structure of the multi-link robotic fish with a pair of pectoral fins

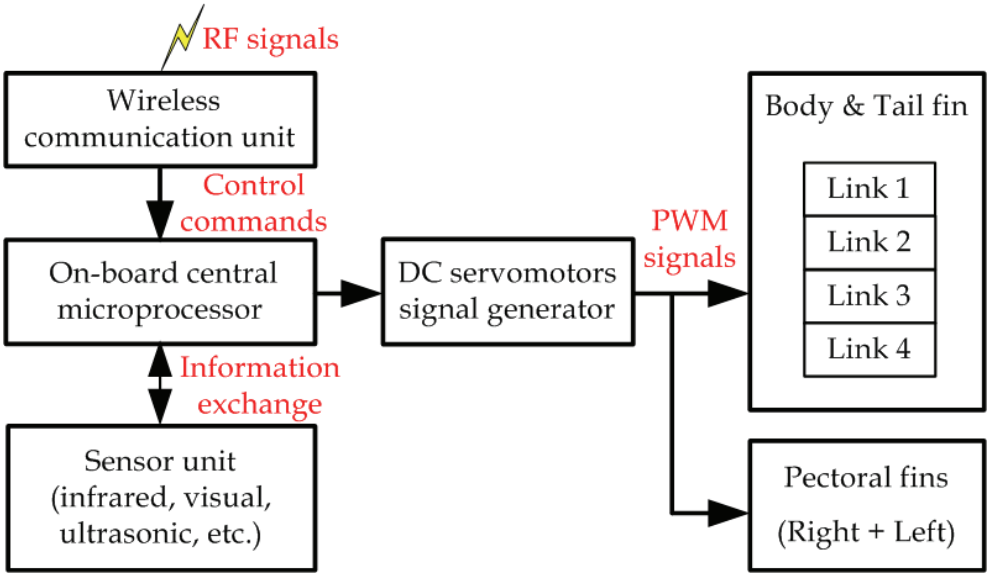


Figure 5. Architecture of the actuator and control units

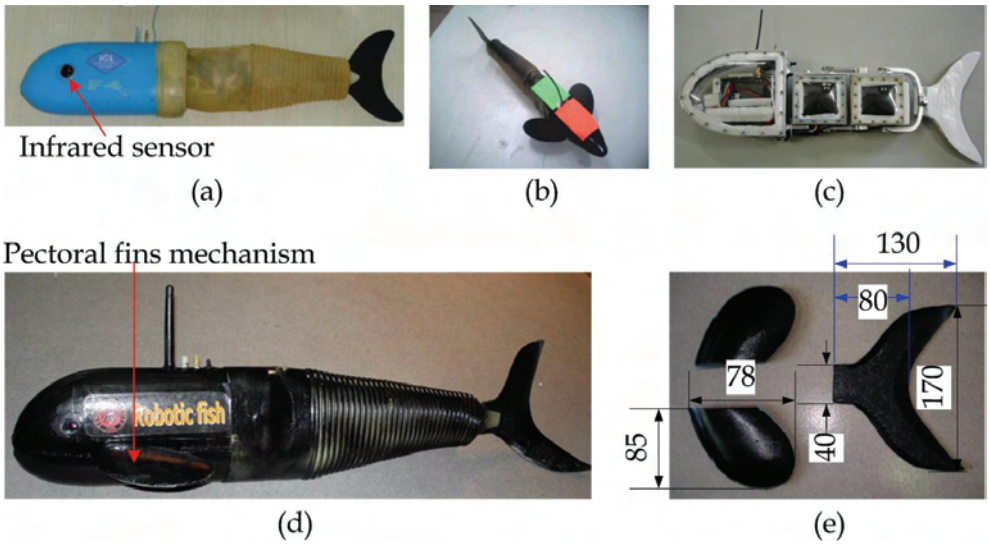


Figure 6. Prototypes of different robotic fish. (a) Four-link robotic fish equipped with infrared sensors, 405 mm in length; (b) three-link robotic fish, 380 mm in length; (c) two-module, reconfigurable robotic fish; (d) four-link, multimode robotic fish with the capability of autonomous three-dimensional (3-D) swimming, 650 mm in length; (e) shape and dimension of both pectoral and caudal fins of the multimode fish

For a lab-based purpose, the fish head and anterior body are united as a streamlined hull which is molded using fiberglass. The hollow head offers a considerable space to amount

the control unit, communication unit, sensor unit, mechanical pectoral fins, and batteries. The rear body is composed of multiple links actuated by DC servomotors, which actively performs lateral fish-like oscillations. Notice that the mechanical arrangement of the oscillatory links is referred to the optimized $l_1 : l_2 : \ldots : l_N$. The links are further externally connected by a lightweight exoskeleton, whose outside is wrapped by an impermeable but stretching skin. Meanwhile, a partly flexible lunate foil connected to the last link acts as the caudal fin. Taking into account that the swimming performance of the robot depends upon the material property of the tail fin to some degree, we adopt rubber to achieve chordwise and spanwise flexibility of the tail fin. Some extensible units, for an advanced version, can be integrated. For instance, three infrared sensors located at the front, the left, and the right of the anterior part of the robotic fish as shown in Fig. 6(a), are used to avoid obstacles autonomously during forward swimming, whereas an ultrasonic detector (510 kHz) located at the bottom of the head is utilized to measure the vertical distance between the fish and the floor of testing water tank. Also, a pair of mechanical pectoral fins (see Fig. 6(d)) whose actuation (DC servomotor) and control are independent of each other, are used for diving/climbing in the vertical plane. Specifically, the shape and dimension for both pectoral and caudal fins are illustrated in Fig. 6(e). Moreover, to ensure a reasonable balance between the resultant gravitational forces and buoyant forces, i.e., to achieve an approximately neutral buoyancy state, some balance weights may empirically be added to or removed from the lower side of the head and exoskeleton. For a large-scale robotic fish, an automatic adjustment mechanism can be fixed to achieve this end.

Two control modes, to date, have been developed: the manual control mode via a remote controller and the automatic control mode via a closed control loop through wireless communication. Table 1 presents basic technical parameters of the robot shown in Fig. 6(a). At this stage, without fish-based 3-D self-positioning ability, a free-swimming robotic fish enabling two-dimensional (2-D) steady swimming and turning maneuvers is chosen as the subject of cooperative control.

Items	Characteristics
Dimension (L × W × H)	~ 405 mm × 55 mm × 88 mm
Weight	~1.38 kg
Sensor	3 infrared sensors (front + left + right)
Number of the links	4
Length of the oscillatory part	~ 200 mm
Maximum forward speed	~ 0.42 mm
Minimum turning radius	~ 200 mm
Actuator mode	DC servomotor
Maximum input torque	3.2 kg.cm
Control mode	RF (433 Hz)
Working volt	4.8 V

Table 1. Technical parameters of a self-propelled, four-link robotic fish

2.3 Motion control

The conspicuous hallmark of fish-like swimming is the compound propulsive system that integrates the maneuvering hardware into the propulsion hardware. Due to this particular

propulsion mode, plus the complexity of the water environment, there exist several difficulties in controlling a robotic fish flexibly and robustly, which are listed as follows:

- Firstly, it is very difficult to establish a precise mathematical model for fish-like swimming via purely analytical methods, since how fish generate forces and maintain stability during propulsion and maneuvering is not well understood. So we can only predict approximately the response of the robot after the control commands are sent.
- Secondly, the robotic fish hardly track a straight line because of inherent lateral oscillation fashion. That is, the movement of a robotic fish is essentially nonlinear, and its swimming pattern is changing dynamically.
- Thirdly, the fish cannot move reversely like a wheel-like mobile robot during propulsion.
- Lastly, waves will be produced when a robotic fish moves. In this case, the movement of the robotic fish will be affected by the waves no matter they are in a stable state or not. This further leads to the uncertainty of the sensory information and the imprecise localization control.

To confront such challenge, we assume that the controllability of the fish relies on the internal shape (the joint angle ϕ_{ij}) for maneuverability and the oscillating frequency f of the moving links for speed. More specifically, the simplified propulsive model presented in Section 2.1, which relates frequency to speed and joint angle bias to turns, is used to generate a variety of swimming patterns. Then the motion control problem in the 2-D plane is decomposed into the speed control and the orientation control. Furthermore, for a robotic fish capable of up-and-down movements, in particular, submerging/ascending control has to be implemented in the 3-D workspace. For instance, the robotic fish is able to execute 3-D motion by adjusting the attack angle of the pectoral fins like sharks that do not have swim bladders.

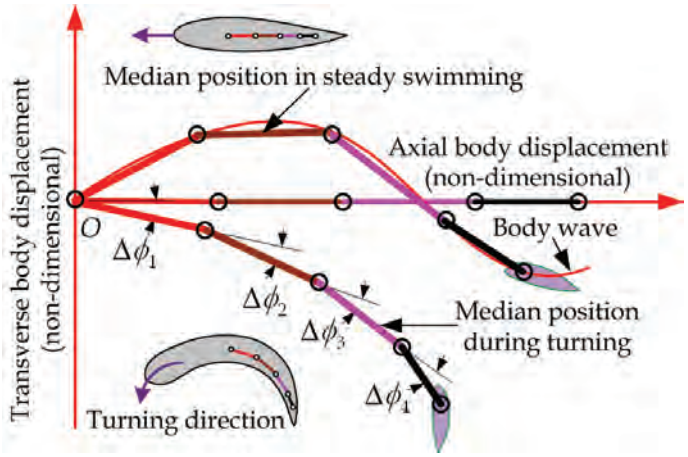


Figure 7. Schematic diagram of adding deflections to the oscillatory links enabling turning maneuvers

As for the speed control, there are three basic approaches to achieve different swimming speeds.

- **Speed control based on the oscillating frequency** *f*. As a general tendency, the advancing speed increases with increasing *f*, and *f* will approximate a constant when the desired speed is arrived.
- **Speed control based on the oscillatory amplitude**. A second order amplitude envelop ($c_1x+c_2x^2$) is applied to produce different body waves through different values of c_1 and $\mathbf{c}_2$, as shown in (2). In practice, oscillatory-amplitude based speed control method can generate different speeds when *f* is fixed.
- **Speed control based on the length of the oscillatory part**. As stated previously, R_l is a vital characteristic parameter, and not all the body or the oscillatory portion contributes to thrust generation at all time, it may be a feasible way to employ different lengths of the oscillatory part at various speeds. For a multi-link robotic fish, this method can be easily operated by locking or unlocking some links. Therefore, the robotic fish can switch between the anguilliform mode and the carangiform one so as to achieve various motions and better swimming performance.

With respect to varying the swimming directions, we add different deflections (i.e., dynamic offsets) to the straight, symmetric gaits propelled only by an oscillating posterior body and tail fin to accomplish various turns. That is, the flexible posterior body and tail moving in the form of body wave is forcibly deflexed to ensure an asymmetric motion, which provides large lateral forces for maneuvering (Read et al., 2003). As depicted in Fig. 7, an internally asymmetric shape of the robot can geometrically be yielded via adding different deflections $\Delta\phi_j$ to each joint. A 2-D array Ctrl_Data[*M*][*N*] (6) for the joint angles can eventually be calculated according to the link-based fitting method (Yu et al., 2004), where $\Delta\phi_j$ is the joint angle between the *j*th link and the (*j* − 1)th link at the interval of the *i*th (*i* = 0, 1, . . . , *M* − 1) link, satisfying $\phi_j = \theta_j - \theta_{j-1}$. Practically, Ctrl_Data[*M*][*N*] is saved as a lookup table stored in the nonvolatile electrically erasable programmable read-only memory (EEPROM) of microcontroller, which serves as the primitive control data for steady swimming and turning maneuvers.

$$\text{Ctrl\_Data}[M][N] = \begin{pmatrix} \phi_{01}+\Delta\phi_1 & \phi_{02}+\Delta\phi_2 & \dots & \phi_{0,N}+\Delta\phi_N \\ \phi_{11}+\Delta\phi_1 & \phi_{12}+\Delta\phi_2 & \dots & \phi_{1,N}+\Delta\phi_N \\ \dots & \dots & \dots & \dots \\ \phi_{M-1,1}+\Delta\phi_1 & \phi_{M-1,2}+\Delta\phi_2 & \dots & \phi_{M-1,N}+\Delta\phi_N \end{pmatrix} \tag{6}$$

For a given turning maneuver, in addition, it can theoretically be achieved by governing magnitude, position, and time of the deflections applied to the moving links. Since the turning maneuver can mathematically be triggered by multiplying a smoothed step function $u(t,\beta,t_0)$ to deflection angle $\Delta\phi_j$, we can then unify different turning maneuvers into a framework by choosing suitable step-function combinations $\sum\varpi_{j'}\mu(t,\beta,t_{j'})$ and $\Delta\phi_j$ (Yu et al., 2006a).

$$\mu(t,\beta,t_0) = \frac{\text{atan}(\beta(t-t_0)) + \arctan(\beta t_0)}{(\pi/2) + \text{atan}(\beta t_0)} \tag{7}$$

where β denotes the positive rise coefficient, t_0 indicates the initial rise moment, *j* is the *j*′th directed (i.e., positive or negative) width coefficient, and $t_{j'}$ is the *j*′th rise moment (*j*′ = 0, 1, .

. ., N' and N' is a positive integer). For example, when a deflection of $\sum \varpi_{j'} \mu(t,\beta,t_{j'}) \times 45°$ is applied to the first two joints of the flexible rear body, a simulated trajectory (Fig. 8) in which the robot can avoid the obstacle successfully is produced. The key to such a relatively complicated maneuver is to decide the suitable $\sum \varpi_{j'} \mu(t,\beta,t_{j'}) \times \Delta\phi_j$, where $\sum \varpi_{j'} \mu(t,\beta,t_{j'})$ takes a form of (8) plotted as Fig. 9.

$$\sum \varpi_{j'} \mu(t,\beta,t_{j'}) = \mu(t,\beta,t_0) - 2\times\mu(t,\beta,t_1) + 2\times\mu(t,\beta,t_2) - \mu(t,B,t_3) \tag{8}$$

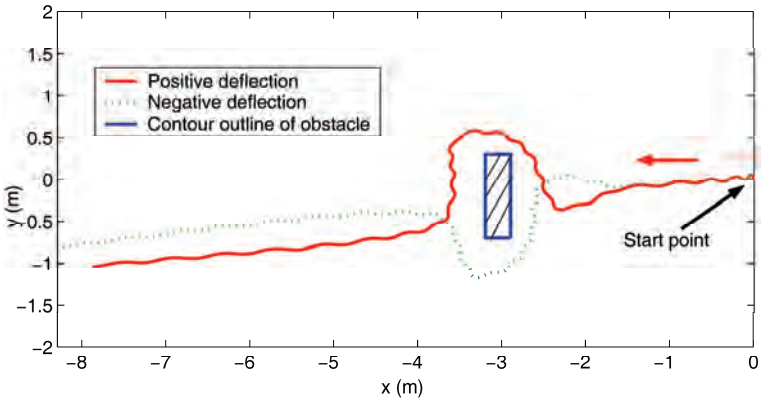


Figure 8. Simulated trajectory for a simple obstacle-avoidance case

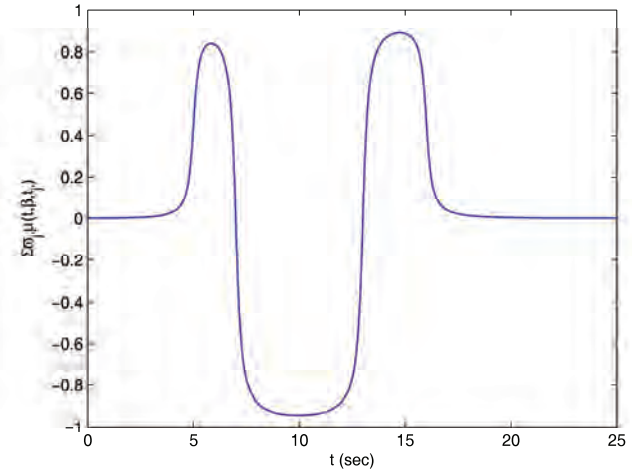


Figure 9. Graphical description of the step-function combination applied to an obstacle-avoidance case

As can be observed from Fig. 9, when the robotic fish swims without turning, $\sum \varpi_{j'} \mu(t,\beta,t_{j'}) \times \Delta\phi_j = 0$, else $\sum \varpi_{j'} \mu(t,\beta,t_{j'}) \times \Delta\phi_j \neq 0$. More specifically, $\sum \varpi_{j'} \mu(t,\beta,t_{j'})$

decides the moment of performing turning while $\Delta\phi_j$ is responsible for turning magnitude (i.e., turning diameter). So, we can infer that $t_0 \neq 0$ holds in the case of *turning during advancing*, that $t_0 = 0$ in the *turning from rest*, and that $|\Delta\phi_j| = \Delta\phi_{\max}$ in the *snap turning*, which corresponds to three basic turning modes defined by Yu et al. (2004). Naturally, some primitive swimming gaits: *straight cruising*, *turn left*, *turn right*, *braking*, etc, can be further devised.

Additionally, some task-oriented swimming modes are designed besides above-mentioned bio-inspired gaits. For instance, pushing an object (e.g., ball) exactly is beyond the natural swimming modes for a fish, but we can define such an artificial swimming mode. Fig. 10 depicts a video sequence of pushing ball, which is entirely composed of a series of primitive swimming modes: *straight cruising*, *turning during advance*, *snap turning*, etc.

To satisfy the demand of real time in cooperative control, we further adopt a PID controller for piecewise speed control and a fuzzy logic controller (FLC) for orientation control, which are presented particularly in our previous work of Yu et al. (2004).

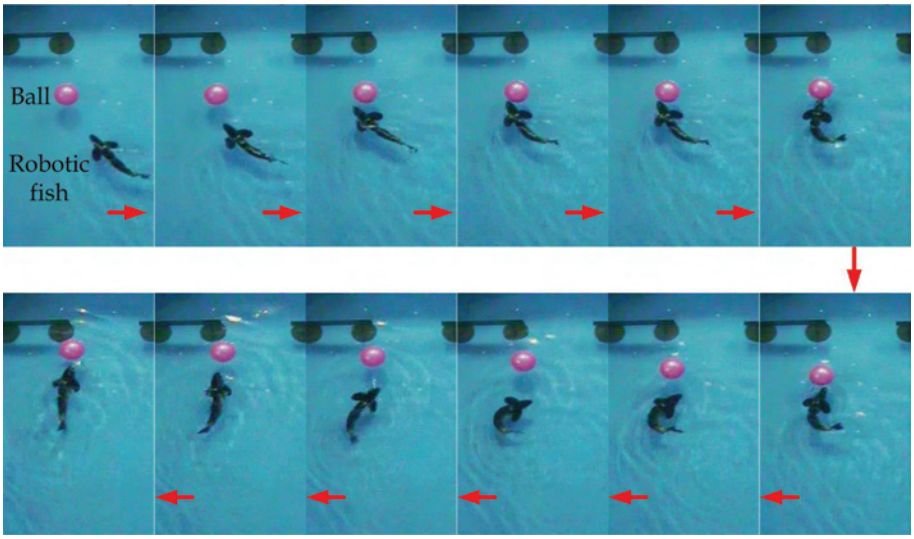
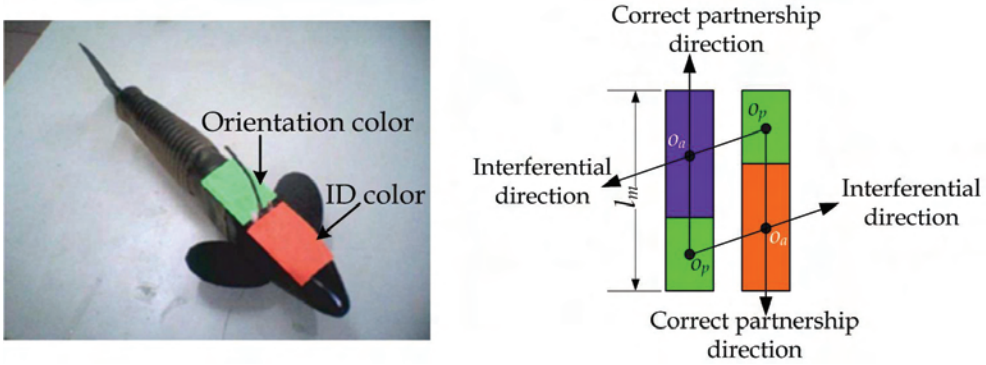


Figure 10. A continuous sequence of pushing ball

3. Visual tracking of multiple robotic fish for cooperative control

The aim of this section is to solve the problem of visual tracking of multiple robotic fish (as large as eight) for cooperative control, where a team of robotic fish in a larger tank by 3.3 m × 2.3 m × 0.7 m are required to perform some given tasks. Since the robotic fish we employed have no ability of self-positioning, thus, an overhead, global vision subsystem is adopted to acquire information of the environment and the states of the fish. As we know, if images are processed more quickly, decisions will be made more efficiently in a control cycle. Hence fast and accurate multi-fish tracking is a central issue for cooperation based decision-making.

For our testing, a CCD camera with wide view which is hung perpendicularly above the water tank (about 2.6 m high) is responsible for capturing the environment information, and the motion status information of the robotic fish within an area of 3.3 m × 2.3 m. The output of the camera is directly fed into a video digitizing card. The sampled resolution is 520 × 384 pixels for each frame, which frame processing rate holds 25 Hz. Unlike conventional ground-based identification, the robots work underwater and flex their body to move, which induces surface waves and even splashes. Moreover, a small quantity of colors less than six can robustly be identified in the laboratory environment, failing to meet the cooperative demand of as large as eight robots. To cope with this situation, anti-jamming measures including optical correction and foil superposition are firstly pre-implemented before segmentation. A color-based thresholding combined with binary coding is then adopted to binarize the sampled image, and a color-index based identification to distinguish the moving fish (Yu et al. 2003; Yu et al. 2006b).



(a) (b)

Figure 11. Scheme of color identification. (a) A robotic prototype attached a color mark; (b) schematic of the color mark

Fig. 11 illustrates a robotic prototype with a custom-built color mark composed of ID color and Orientation color. The robotic fish, as an elongate body, allows a short and narrow space for attaching the color mark. Naturally, two small, different color marks are employed to yield necessary position and orientation information of the fish, i.e., to calculate the centroid and the slope of the color block. Based on this heuristic scheme (Yu et al., 2003), suppose that the number of the colors that can be recognized clearly in various underwater conditions is Q, the vision system will accommodate at most $Q - 1$ robotic fish, where Q stands for a positive integer not less than two. However, experiments show $Q = 6$ for our lab-based lighting. So we can only identify five fish using this scheme, which may not satisfy the demand of at most eight robotic fish to be positioned in MRFS. After analysis, we find the orientation color is not used sufficiently in this scheme. As a remedy, available colors are uniformly arranged in two groups: anterior color group and posterior color group. Making a permutation and combination for the two groups, there is a sum of $int(Q/2) \times int(Q/2)$ to be theoretically distinguished, where $int(\ )$ denotes the integer function. Note that we are able to identify nine (3 × 3) fish with this scheme. The next task is how to work out this scheme considering all possible cases occurred in testing.

As shown in Fig. 11b, the anterior color block is aligned with the posterior block, and the orientation of the mark is prescribed as the centroid of the posterior block (o_p) directs towards the anterior one (o_a). A sticking point in this scheme is to precisely recognize partner of the anterior block. Empirically, we search the recorded centroid within a circular area, whose center coincides with the o_a and radius is chosen as $2l_m/3$, where l_m is the total longitudinal length of the anterior block plus the posterior one. In most cases, only one o_p can be found, which means the anterior block has successfully been matched its partner. A tag is simultaneously attached to globally declare its state. However, an unfavorable case would arise, in which two marks are too close and two posterior centroids can be found. To deal with this situation, like the heuristic mark identification method, we have to calculate the longitudinal centerline of the anterior block by a least square fitting so as to offer assistance in judging the right posterior partner. After all, there seems to be little probability of triggering this extreme condition.

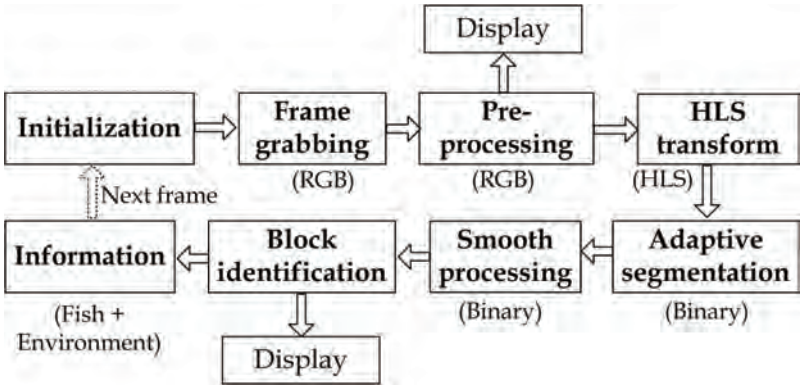


Figure 12. Block diagram of an overall multi-fish tracking procedure

To speed up locating interested pixels during adaptive segmentation and closure operation, all time-consuming, iterative image operations are optimized with the aid of parallelized SIMD (single-instruction, multiple-data) technologies embedded in processors. Fig. 12 summarizes the overall tracking operations in block diagram. Following a sequence of frame grabbing, pre-processing, HLS transform, adaptive segmention, smooth processing, identification, and information output, we can obtain the desired information on the multi-fish and their surroundings, which is served as a basis for effective decision-making. To reduce adverse disturbance resulting from capture hardware and illumination change, a pre-processing involving optical correction and foil superposition is conducted before the image segmentation. Also note that the sampled video in 24-bit RGB is converted to widely used HLS color space through HLS transform. This is because R, G, and B components of image data are scattered and correlated, rendering malfunction in most visual applications.

To further evaluate the performance of multi-fish tracking, strict tests are executed on a WINDOWS XP operation system with a Pentium IV 3.0 G processor, a memory of 1 G, and a compiler of Microsoft Visual Studio.net 2003. When the resolution disposed is set to 624×434 pixels in 24 bits, high-precision functions, *QueryPerformanceFrequency()* and *QueryPerformanceCounter()*, which are embedded in the operation system, are employed to measure the runtime of a processing cycle. In a static environment with projected lamplight, it only takes about 24.143 ms to identify eight fish marks and an obstacle demonstrated in

Fig. 13 in one cycle, compared with 167.893 ms coded in C++ Language. The speedup ratio is as high as 6.9. Meanwhile, a typical update rate for the frame grabbing is 40 ms allowing plenty of time for cooperative algorithms and other operations.

(a) (b)

Figure 13. Experimental scenario and results of multi-object tracking. (a) Eight fish marks and an obstacle floating on the water tank; (b) identification results associated with positions and moving direction

4. Hierarchical architecture for cooperative applications

In this section, we propose a hierarchical control algorithm for cooperative applications in MRFS. A five-level hierarchical architecture is illustrated in Fig. 14.

The first level is task planner level. In this level, a coordinator is designed to acquire and, more importantly, to process the vision information involving the fluid circumstance and the states of robotic fish. Therein, the kinematics of the fish contained in the sampled image can be extracted by analyzing moving marks in real time. The required task is then decomposed into different roles on the basis of global, vision-derived information. A prediction should be ensured that these roles are competent for the assigned subtask during the decomposition.

After producing different roles, we concentrate on the role assignments which are executed in the second level. Since any fish member has the same capabilities and can take on any role within the task, we should select the most qualified candidate for each role according to some appropriate rules and try to achieve the task effectively as a whole. To confront with specific requirements for different tasks, we introduce both static and dynamic role assignments mechanism. For static assignments, once roles are decided at the beginning of the task, they will not shift during the implementation of the tasks. But for the dynamic assignments mechanism, the fish may switch their roles according to the progress of the task.

The third level is the behavior level, in which different component behaviors are designed for each role. In this level, protocols in the form of finite state automaton (FSA) are employed to organize these behaviors, formalizing the switches and restraints between them. In practice, the design of behaviors, bio-inspired or artifical, has a great impact on the efficiency and performance of cooperative control. Note that the robotic behaviors and roles are designed to be complementary to each other, so that a fish group exhibiting good collective behavior can be achieved.

Following the behavior level is the action level, in which a sequence of actions is assembled for a specific behavior. Considering the fluid circumstance and mechanical constrains of the fish body, five basic actions are designed, which are *turning during advancing*, *snap turning*, *turning from rest*, *forward advancing*, and *drifting*. The first three actions may be further

divided into two categories: *turn to right* and *turn to left*. More advanced behaviors such as obstacle avoidance, moving to goal, and wander can be achieved by combining these basic actions in a certain sequence.

The last level, which is called the controller level, is the lowest one. As mentioned above, the fish's motion control, that is action control, can ultimately be resolved into two parallel types of control laws: the speed controller and the orientation controller. Given a desired speed, a PID controller is applied. Meanwhile, a fuzzy logic controller is imported as a solution to the orientation control.

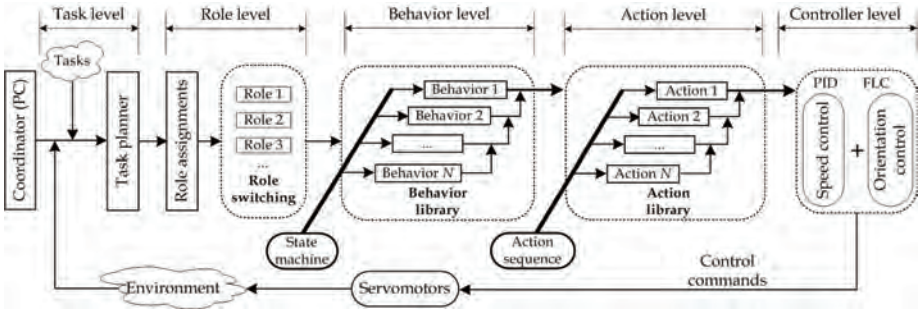


Figure 14. Block diagram of a hierarchical architecture for cooperative control

5. Design and implementation of the MRFS platform

For the sake of investigating fish-like propulsion and underwater robots based cooperative control, we have developed an experimental platform MRFS shown in Fig. 15. The hardware platform can roughly be decomposed into four subsystems: robotic fish subsystem, visual tracking subsystem, decision-making subsystem, and wireless communication subsystem. The robotic fish, as the executors, swim in the tank and implement control commands generated from the host computer. The information about the fish and their environments which are marked with specified colors in the experiments, are captured by an overhead camera with wide view. The camera is directly connected to the frame grabber embedded in the computer, and the captured information is sent to the computer in real time. After the image information is effectively processed in the visual tracking subsystem, they are sent to the decision-making subsystem as inputs. Then, based upon input signals and specific control strategies for given tasks, the decision-making subsystem generates corresponding control commands and transits them to each fish through the wireless communication subsystem. Thus, a full control loop is implemented.

A task-oriented software platform that is compatible to the hardware architecture is further developed. This software incorporates visual tracking, cooperative control, and bidirectional communications, on which users can perform multifold functions related to task selection, parameter setting, image processing, control algorithms loading, as well as real-time display. Fig. 16 depicts the well-designed graphical user interface (GUI) for MRFS. This GUI, from the perspective of function, can be divided into the following six components:

- **Real-time display**, where the globally captured image covering the overall experimental scene is displayed after optical correction;
- **Track results**, where the identification results relevant to the robotic fish (position and moving direction) and environmental objects (position) are synchronously displayed;

- **Parameters settings**, in which the user can define color information for robotic fish and environments (goals, obstacles, etc);
- **Individual fish control**, where the user can manually test and check the status of robotic fish, including velocity, direction, adopted mode, etc;
- **Online debugging information**, in which the running status of the system is shown to facilitate the later debugging;
- **Task selection**, where the operator is able to preset, select, and even add different cooperative tasks.

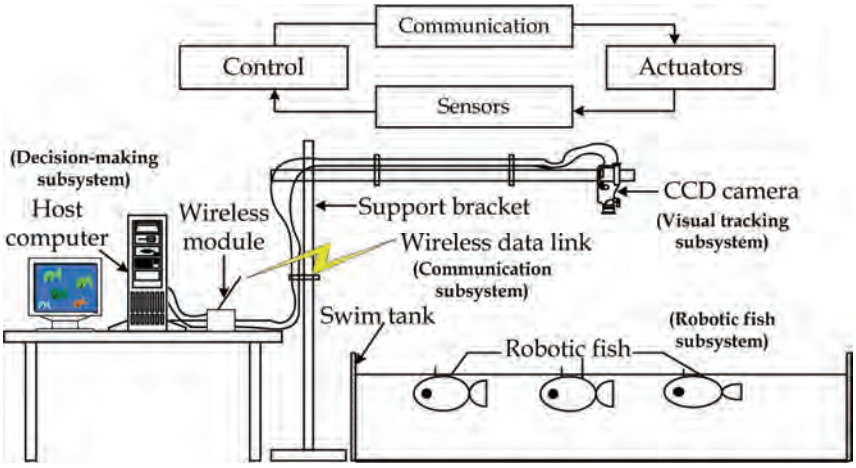


Figure 15. Hardware platform for the MRFS

Figure 16. Graphical user interface of the MRFS

Therefore, the MRFS is competent for a variety of environments (goals, obstacles, etc) and tasks, which provides a standard platform to integrate and examine cooperation algorithms for underwater robots.

6. Experiments and results

In this section, three cooperative tasks including passing hole, Fish versus Man (FVM), and water polo are presented to show how fish cooperate or compete with each other to achieve certain tasks. Relevant control strategies and experimental results are detailed as follows.

6.1 Passing hole

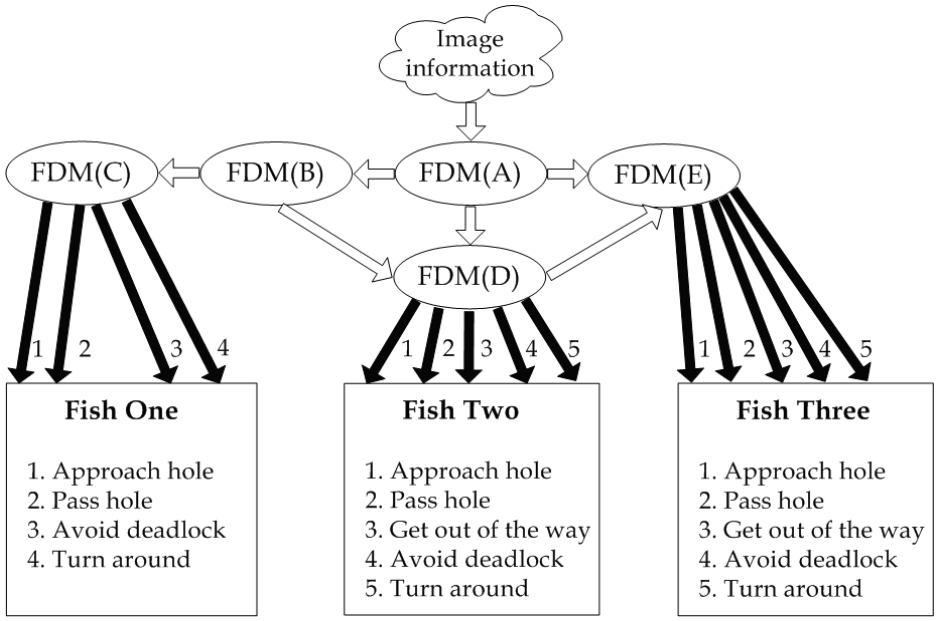


Figure 17. FDM-based behavior architecture for the passing hole. Notice that FDM(A) decides which fish passes the hole now, FDM(B) judges whether other fish should get out of the way, FDM(C) decides which behavior will be taken by Fish One, FDM(D) determines which behavior should be performed by Fish Two and whether Fish Three should get out of the way or not, and FDM(E) selects a suitable behavior for Fish Three

Three fish, in this testing, are required to pass a 150 mm wide hole located in the middle of a water tank according to an appointed order and come back orderly. They also need avoiding collision and mutual crowding during the coordinated motion. Attended roles associating with the appointed passing order, can simply be described as Fish One, Fish Two, and so on. A set of behaviors directed at fast and collision-free swimming are designed as follows:

- **Approach hole**, where the hole is regarded as the goal and the typical PTP algorithm is called.

- **Pass hole**, where the fish adjusts its direction in a small scale when the hole is near. Notice that it will swim straightly at a full speed when the direction is accurate.
- **Avoid deadlock**, where the fish alter its assigned goal to a temporary one in order to get rid of the standstill resulting from insufficient workspace.
- **Turn around**, where some favorable swimming movements are assembled to prepare coming back when a fish has entirely passed the hole. An available alternative is that the fish in full speed stop moving suddenly and turn right or left on a large scale. Thus, the fish can turn around quickly at a very small turning radius.
- **Get out of the way**, where the fish with low priority should get out of the way when obstructing the high-priority one. In particular, this behaviour will work for all fish except Fish One.

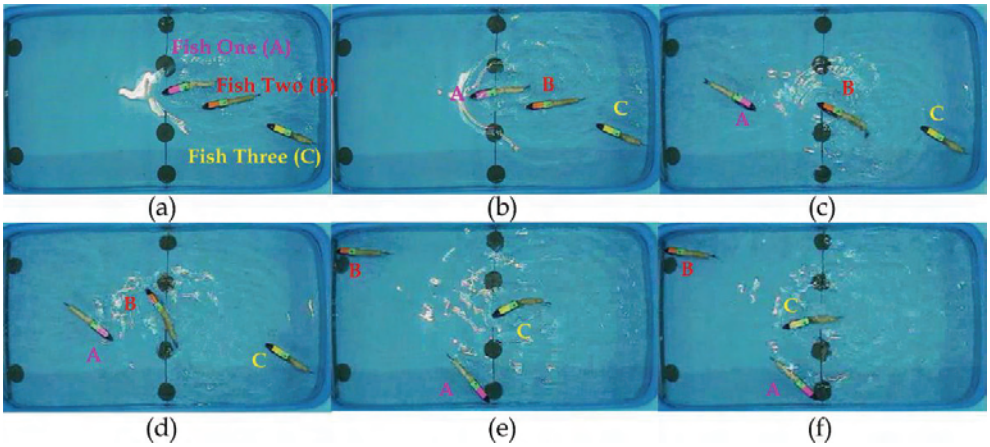


Figure 18. Snapshot sequence of a successful cooperative passing hole [from (a) to (f)]

In order to improve the implementation success rate, a Fuzzy Decision-Making function (FDM) is introduced to coordinate different actions into behaviors, various behaviors into roles, and select suitable roles for different robotic fish. As shown in (9), a FDM has an input variable and two kinds (sometimes only one kind) of output variables.

$$[I_{out}, Be] = \mathrm{FDM}(I_{in}) \tag{9}$$

where I_{in} and I_{out} are the information variables related to the states of fish, environment information, and relative fuzzy rules, and Be is the behavior variable composed of behavior object and action signal. The FDM is based on the minimal "work" criterion, i.e., the sum of the work functions for all selected behaviors and the additive work functions will be minimal. Note that the work function is defined to measure how much work the robotic fish should apply to fulfil the given behaviour, and that the additive work function is also created to coordinate multiple behaviors in terms of fuzzy rules. For instance, if the behaviour X violates any fuzzy rule, it will be set to a large additive work. In contrast to the maximal contribution criterion (Vadakkepat et al., 2004), the minimal work criterion is much easier to perform, since the contribution of a selected behavior is reasonably assessed in an unambiguous fashion. Additionally, the fuzzy rules within this architecture have been translated into functions, i.e., the actions and behaviors coordination are generated via fuzzy

techniques, which are more flexible and easier to operate. Fig. 17 shows the FDM-based behavior architecture for the passing-hole. Therein, the elliptic blocks stand for FDMs and rectangular blocks for sets including executors and optional behaviours. The hollow arrows direct towards only other FDMs, while the solid arrows towards executors and the indexes adjacent to the arrows correspond to the numbered behaviors listed in the rectangular blocks. Notice also that no two behavior variables or information variables can concurrently be generated by a FDM, which avoid any possible conflicts. A successful snapshot for cooperative passing hole is depicted in Fig. 18. As can be observed, there exhibits good coordination among three fish, and the posterior fish will voluntarily give way to the anterior one during going through the narrow hole.

6.2 Fish VS Man

The second cooperation testing is a competitive game between three fish controlled by the computer and a manually controlled fish, called Fish versus Man (FVM). As illustrated in Fig. 19, there exist three goals (about 120 mm wide) in a static pond 3050 mm × 1830 mm × 560 mm (length×width×depth): G_d, G_a, and G_m. Specifically, G_d is the middle goal which divides the field into two section: the left section I is for the three automatic fish denoted as AF and the right part II is occupied by the manually controlled fish which is denoted as MF. G_a and G_m are goals for AF and MF respectively. The rules for this competition are defined as which side first enters into the opponent's goal wins the game. Obviously, one automatic fish can hardly stand against a manually controlled fish. However, when the three members cooperate effectively, the results may be different. Next, we will design the cooperative strategy for the AF team according to the formed hierarchical architecture.

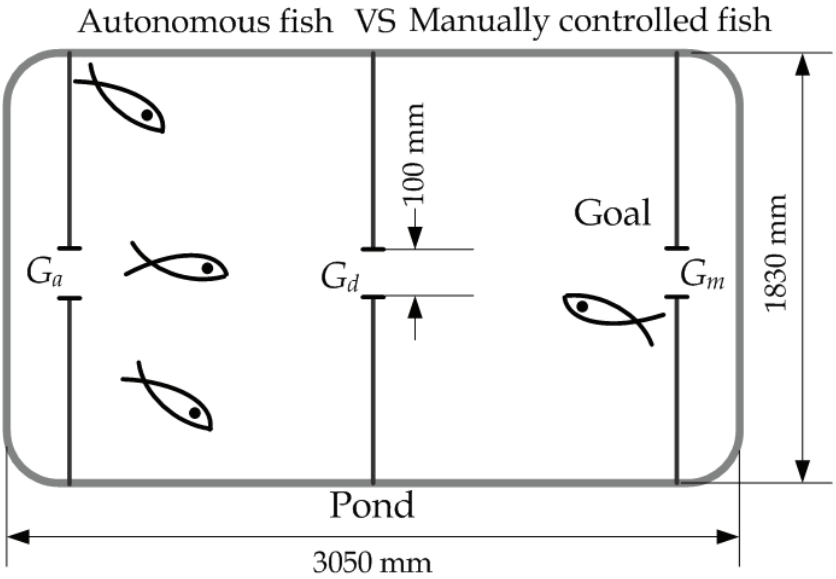


Figure 19. Illustration of the FVM game

The objective for AF team is to swim into G_m as quickly as possible and meanwhile, to prevent MF from intruding G_a. So, achieve this task collectively, the three automatic fish may be divided into three roles: the main attacker who are responsible for swimming to the opponent's goal, the main defense who are responsible for preventing the opposition MF from scoring and buying time for the attacker, and the secondary defense, to assisting the main attacker and blocking the movement of the opponent.

After the roles are produced, we will assign these roles to members of AM in such an order: first, to determine who take on the main defense role, then the secondary defense role, and finally, the attacker. During the assignments, we follow the rules as below.

- The fish nearest to the optimal defending point will be selected as the main defense.
- The one nearest to the optimal secondary defending point becomes the secondary.
- The remainder, naturally, is the attacker. Note that role assignments are dynamic in this case, so the fish may switch between different roles according to progress of the game.

For the attacker, five behaviors are designed as: Go-To-Goal, Avoid-Obstacle, Go-Through-Gate, Stop, and Recovering. For the defenses (both main and secondary), five behaviors are proposed too, which are Get-In-Position, Avoid-Obstacle, Blocking, Stop, and Recovering. Go-To-Goal and Get-In-Position are typical PTP control algorithms presented in Yu et al. (2004). Avoid-Obstacle is to negotiate the nearest obstacle during the movement from the initial position to the destination. Go-Through-Gate behavior allows the fish to pass through the gate in a safe way. Blocking behavior is for the defenses to head the opponent MF off before it approaches G_a. Stop denotes the state without any control input. Recover is the restart behavior, which is employed when the fish has a failure. For each behavior, several actions presented as in Section 2 are organized orderly and executed by setting different parameters to both the speed and the orientation controllers.

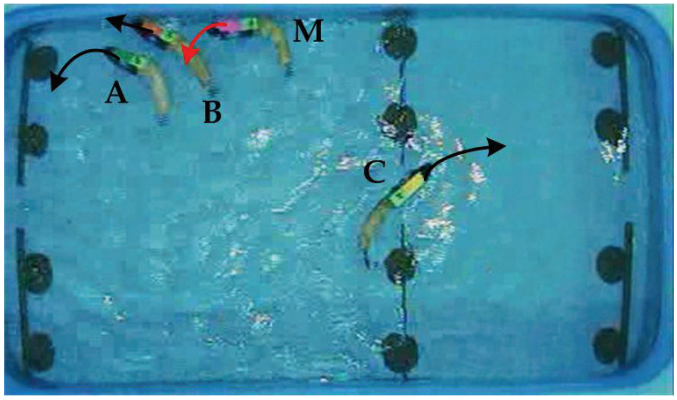


Figure 20. Snapshot of a running FVM game

An experimental snapshot of the game is shown in Fig. 20 and moving trajectories of four robotic fish (A, B, C, and M) in Fig. 21. The three coordinated fish, as we expected, blocked the manually controlled fish successfully and won the game. Clearly, the used coordination strategies worked out. As can be seen from Fig. 21, the scattered points falling into the left-top area of the pond are dense, whereas the points describing the fish C take a figure-eight motion lasting for 25 s. This means the defensive strategies facing the fish M are effective to

some extent, but the Go-Through-Gate behavior usually used the attacker (fish C) are of low efficiency.

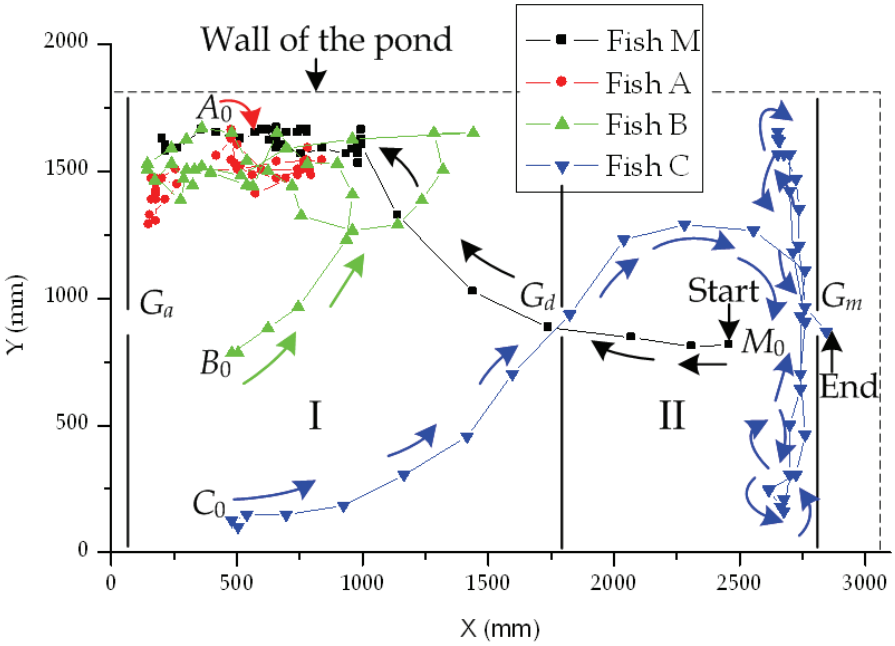


Figure 21. Moving trajectories of the fish in the FVM game lasting 41 s

6.3 Water polo

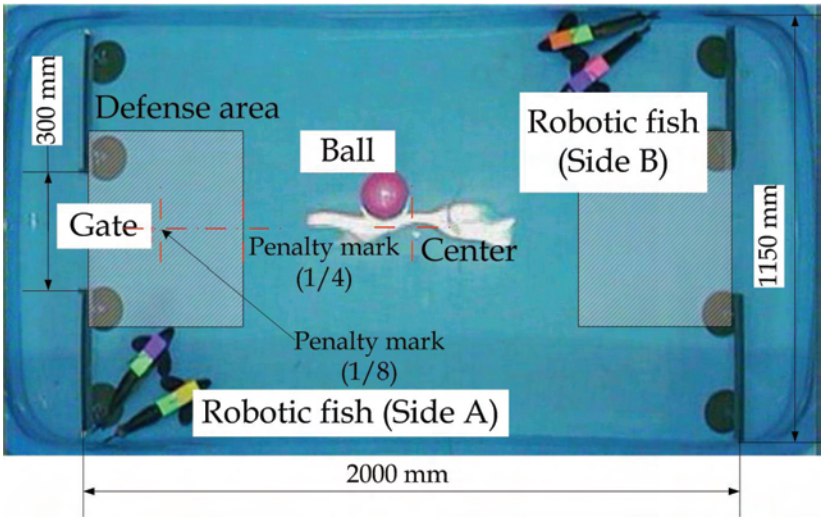


Figure 22. Illustration of the water polo

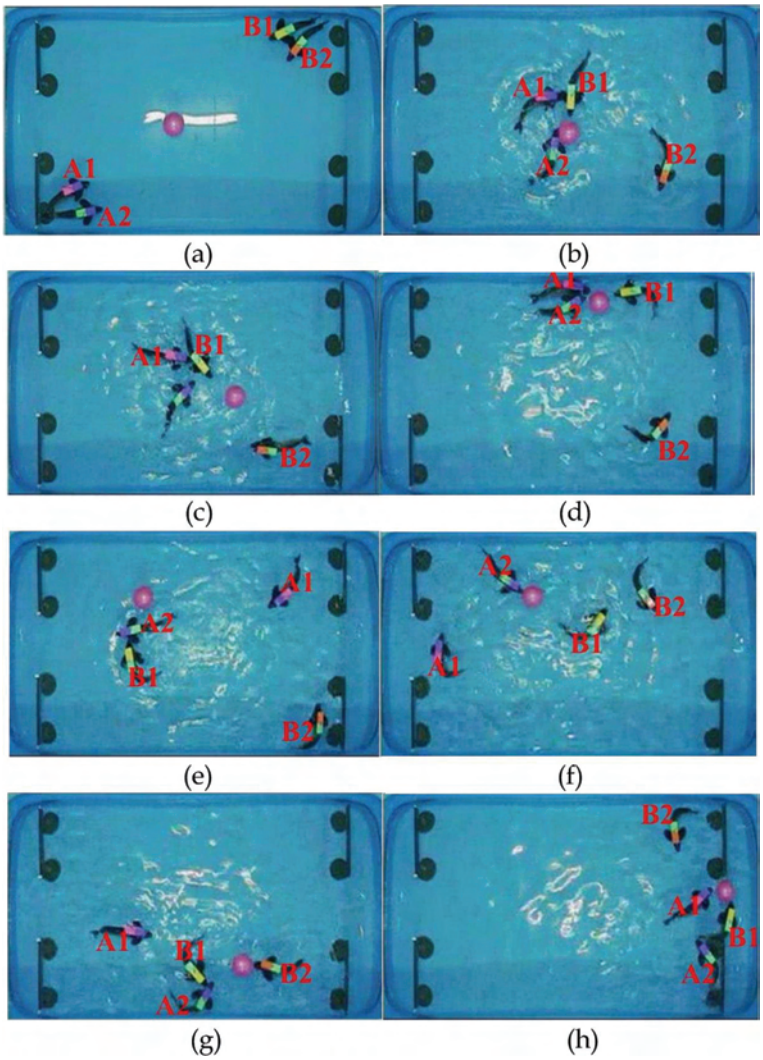


Figure 23. Snapshot sequence of a typical water polo [from (a) to (h)]

The third experiment is designed to let two teams of robotic fish compete with each other through push a polo, which is referred to as "water polo". Similar to the robot soccer game, water polo is intended for providing a standard platform to test and validate aquatic cooperation algorithms. At present, a water polo competition with four robotic fish, i.e., two versus two, is conducted. As shown in Fig. 22, the playground is limited as a water area of 2000 mm × 1150 mm on which four fish grouped as Side A and Side B compete against each other to push the ball to the opponent gate. The rules of the game can be summarized as follows.

- **The robotic fish**. A kind of three-link robotic fish is used as the subject. For convenience of locating robots in motion, two distinguishable color marks are deployed for two sides before the game. Apart from the vision-based color marks, at current stage, no other fish-based sensors are allowed to be mounted for advanced control. In particular, if a robotic fish does not function properly, the game will be paused until a substitution fish in the "second string" enter on the playgroup. Notice that the robotic fish may not be substituted at will and that only three chances of replacements are permitted in a game.
- **The playground**. As shown in Fig. 22, a specific defense area is situated at each end of the water tank. Also penalty marks are set to 1/4 and 1/8 of the longitudinal midline. A pink, 2/3 submerged, inflatable ball is served in the center or penalty mark when starting the water polo. Like the robotic fish, the ball may not be substituted at will unless it becomes defective.
- **The method of scoring.** Like football games, a goal will be scored when the center of the ball gets across the gate baseline. As a rule, the side scoring the greater number of goals in a game is the winner.
- **The duration of game.** The game lasting 30 minutes is divided into two equal playing periods for the moment.

Fig. 23 depicts an image sequence of a running water polo, in which two teams Side A and Side B take different strategies: in Side A, both robotic fish are attackers, whereas in Side B, B1 is an attacker and B2 is a defense. Finally A1 hits the ball into the opponent gate and hence Side A makes a point. We remark that the game water polo emphasizes the strategy, and different strategies may lead to different results. Here, we only adopt one possible strategy, and other choices should be further examined and verified.

6.4 Discussions

Extensive experiments have been carried out to verify the feasibility and effectiveness of the formulated cooperative framework and adopted strategies. However, the experimental results for the moment are not perfect but successful and promising, which are mainly affected by the following unsolved or open factors.

- **Controllability**. This factor involves the controllability of both the fish-like robots and the behavioral framework. For the former, fish-like swimming skills and intelligence can not be fully mimicked. The necessities of high swimming skills include hydrodynamic based modeling and sensors based perception. Unfortunately, dynamic models, smart sensors, communication links, as well as on-board processing integrated control algorithms, are still not well tackled in underwater environments and in fish-like robots. This deficiency will debase the capability of swimming and further impact the performance of the collective behaviors to a large extent. For the latter, how to avoid conflicts and ensure operational efficiency is the key to fulfil high-quality cooperative control since the behavioral framework do not directly define an input-output structure.
- **Uncertainty.** There are two uncertainties that constrain the repeatability of the applied control strategies. One is from the measurement error of the vision subsystem due to changing lighting and splashed waves. This would bring trouble in performing precise motion control. Another source is the manufacture of the robotic fish. Because the

robots are made manually, there exists small performance difference among the developed robots, such as the maximum speed and minimum turning radius, etc.

- **Scalability.** Because of the restriction of the experiment field, to avoid mutual collision frequently and to improve operational efficiency, there is an upper limit on the fish number participating in the cooperation. When a larger space is available, more fish will swim freely in the pond, and other complicated tasks such as self-organized schooling and cooperative formation control may be designed, but in turn complicating the control laws design.
- **Compatibility.** The conceived cooperative experiments are performed in the indoor environment. Especially, the global vision subsystem prohibits the widespread use of the MRFS in complex outdoor environments. To achieve real autonomy, some fish-based sensors and processors should be imported. In addition, the MFRS only handles 2-D swimming behaviors. Since vivo fish gracefully execute maneuvers with ease in 3-D aquatic environments, design and implementation of robotic fish enabling 3-D movements are highly needed to future real-world applications. Of course, the current behaviour cooperation architecture can also be applied to 3-D hydrodynamic environments but requires some further adaptations so that it becomes a part of the admissible, 3-D swimming-oriented system behaviors.

7. Conclusion

This chapter has reviewed some of the issues involved in creating a multiple robotic fish cooperation platform inspired by the astonishing cooperative power exhibited by fish school. Grounded on an optimized kinematic and dynamic model of robotic fish, a group of radio-controlled, multi-link fish-like robots as well as their motion control were developed. To enable a closed control loop, a vision-based multi-object tracking subsystem for multiple robotic fish was built, where an improved parallel visual tracking method is formulated within a framework synthesizing features of fish features and surrounding disturbance. A hierarchical architecture for the artificial multi-fish system was further proposed to generate coordinated behaviours and actions. Experimental results on three typical cooperation tasks demonstrated the effectiveness of the proposed scheme.

The ongoing and future work will focus on developing a 3-D dynamic model for free-swimming, multi-link robots and on investigating experimentally the corresponding control laws, which makes preparations for cooperative control of multiple robotic fish in 3-D aquatic environments. In addition, some learning algorithms will be incorporated into the cooperative framework to enhance the cooperation efficiency.

8. Acknowledgement

The authors would like to thank all of the students in the Intelligent Control Laboratory at Peking University before March 2006, including Ruifeng Fan, Yimin Fang, Jinyan Shao, Dandan Zhang, Lizhong Liu, Wei Zhao, and Yonghui Hu, for their daily technical assistance and helpful discussions.

This work was supported in part by the National Natural Science Foundation of China under Grant 60505015, Grant 60635010, and Grant 60775053, in part by the Municipal Natural Science Foundation of Beijing under Grant 4082031, in part by the National 863

Program under Grant 2007AA04Z202, and in part by the CASIA Innovation Fund for Young Scientists.

9. References

Anderson, J.M. & Chhabra, N.K. (2002). Maneuvering and stability performance of a robotic tuna, *Integ. and Comp. Biol.*, Vol. 42, 2002, pp. 118–126

Bandyopadhyay, P.R. (2004). Guest editorial: Biology-inspired science and technology for autonomous underwater vehicles, *IEEE J. Ocean. Eng.*, Vol. 29, No. 3, Jul. 2004, pp. 542–546

Bandyopadhyay, P.R. (2005). Trends in biorobotic autonomous undersea vehicles, *IEEE J. Ocean. Eng.*, Vol. 30, No. 1, 2005, pp. 109–139

Barrett, D.S. (1996). Propulsive efficiency of a flexible hull underwater vehicle, *Dissertation for the Doctoral Degree*, Cambridge, MA: Massachusetts Institute of Technology

Das, A.K.; Fierro, R.; Kumar, V.; Ostrowski, J.P.; Spletzer, J. & Taylor, C.J. (2002). A vision-based formation control framework, *IEEE Transactions on Robotics and Automation*, Vol. 18, No. 5, October 2002, pp. 813–825

Kumar, V. ; Leonard, N. & Morse, A. S. (Eds.) (2005). *Cooperative Control*, *Lecture Notes in Control and Information Sciences*, Vol. 309, Berlin: Springer-Verlag

Lauder, G.V. ; Anderson, E.J.; Tangorra, T.J. & Madden, P.G.A. (2007a). Fish biorobotics: kinematics and hydrodynamics of self-propulsion, *The Journal of Experimental Biology*, Vol. 210, 2007, pp. 2767–2780

Lauder, G.V. & Madden, P.G.A. (2007b). Fish locomotion: kinematics and hydrodynamics of flexible foil-like fins, *Exp. Fluids*, Vol. 43, 2007, pp. 641–653

Ögren, P. ; Fiorelli, E. & Leonard, N.E. (2004). Cooperative control of mobile sensor networks: adaptive gradient climbing in a distributed environment, *IEEE Transactions on Automatic Control*, Vol. 49, No. 8, August 2004, pp. 1292–1302

Pettersen, K.Y. ; Gravdahl, J.T. & Nijmeijer, H. (Eds.) (2006). *Group Coordination and Cooperative Control*, *Lecture Notes in Control and Information Sciences*, Vol. 336, Berlin: Springer-Verlag

Rabbath, C.A. ; Su, C.Y. & Tsourdos, A. (2007). Guest editorial: introduction to the special issue on multivehicle systems cooperative control with application, *IEEE Transactions on Control System Technology*, Vol. 15, No. 4, July 2007, pp. 599–600

Read, D.A.; Hover, F.S. & Triantafyllou, M.S. (2003). Forces on oscillating foils for propulsion and maneuvering, *J. Fluids and Structure*, Vol. 17, 2003, pp. 163–183

Sfakiotakis, M.; Lane, D.M. & Davies, J.B.C. (1999). Review of fish swimming modes for aquatic locomotion, *IEEE J. Oceanic Eng.*, Vol. 24, No. 2, Apr. 1999, pp. 237–252

Triantafyllou, M.S. & Triantafyllou, G.S. (1995). An efficient swimming machine, *Sci. Amer.*, Vol. 272, No. 3, March 1995, pp. 64–70

Triantafillou, M.S.; Techet, A.H. & Hover, F.S. (2004). Review of experimental work in biomimetic foils, *IEEE J. Ocean. Eng.*, Vol. 29, No. 3, 2004, pp. 585–594

Yu, J.; Tan, M. & Wang, S. (2003). A parallel algorithm for visual tracking of multiple free-swimming robot fishes based on color information, *Proc. IEEE Int. Conf. Robot., Intelligent Syst. Signal Process*, 2003, pp. 359–364, Changsha, China

Yu, J.; Tan, M.; Wang, S. & Chen, E. (2004). Development of a biomimetic robotic fish and its control algorithm, *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, Vol. 34, No. 4, July 2004, pp. 1798–1810

Yu, J.; Wang, L. & Tan, M. (2005). A framework for biomimetic robot fish's design and its realization, *Proc. Amer. Contr. Conf.*, 2005, pp. 1593–1598, Port-land, OR, USA

Yu, J.; Liu, L. & Wang, L. (2006a). Dynamics and control of turning maneuver for biomimetic robotic fish, *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5400–5405, October 2006, Beijing, China

Yu, J.; Fang, Y.; Wang, L. & Liu, L. (2006b). Visual tracking of multiple robotic fish for cooperative control, *Proc. of IEEE International Conference on Robotics and Biomimetics (ROBIO 2006)*, pp. 85–90, Kunming, China, Dec. 2006

Yu, J.; Wang, L. & Tan, M. (2007). Geometric optimization of relative link lengths for biomimetic robotic fish, *IEEE Trans. Robot.*, Vol. 23, No. 2, 2007, pp. 382–386

Videler, J.J. & Hess, F. (1984). Fast continuous swimming of two pelagic predators, saithe (Pollachius virens) and mackerel (Scomber scombrus): a kinematic analysis. *J. Exp. Biol.*, Vol. 109, 1984, pp. 209–228

Vadakkepat, P.; Miin, O. C.; Peng, X. & Lee, T.H. (2004). Fuzzy behavior-based control of mobile robots, *IEEE Transactions on Fuzzy Systems*, Vol. 12, No. 4, 2004, pp. 559–564

Wang, X. ; Yadav, V. & Balakrishnan, S. N. (2007). Cooperative UAV formation flying with obstacle/collision avoidance, *IEEE Transactions on Control System Technology*, Vol. 15, No. 4, July 2007, pp. 672–679

Zhang, D.; Wang, L. & Yu, J. (2007). Coordinated transport by multiple biomimetic robotic fish in underwater environment, *IEEE Transactions on Control System Technology*, Vol. 15, No. 4, July 2007, pp. 658–671

15

K-ICNP: a Multi-Robot Management Platform

Tao Zhang[1], Christophe Agueitaz[1], Yun Yuan[1] and Haruki Ueno[2]
[1]*Department of Automation, Tsinghua University,*
[2]*Intelligent Systems Research Division, National Institute of Informatics*
[1]*China,* [2]*Japan*

1. Introduction

As robots are bound to greatly grow in importance in a not-so-far future's everyday life, the question of their integration within their environment naturally arises as one of major importance. One cannot expect random users to adapt themselves to complex system beyond a certain level, this statement becoming even truer if elderly users are to be involved. Generally speaking, the more intuitive the system is to use, the better. In the ideal case, users would interact with a group of domestic robots using only natural language and gestures, and would then be able to communicate with them without virtually needing any prior knowledge about the devices involved. Such a system is recently nominated by a symbiotic autonomous human-robot system (Ueno, 2002). As of today, such an approach is not realistic, as it would require a human-like intelligence in order to resolve ambiguities and guess intentions. A more feasible approach would be putting stronger constraints on the interaction protocol, that is, for instance, asking the user only to use simple language structures and words. That is what we chose to do, trading user learning time for feasibility.
To aim at implementing such a multi-robot system with intelligent interaction, coordinative control of robots is one of crucial issues. So far, there are many researches proposing various approaches facing to different requests. For example, Huntsberger (Huntsberger, 2003) proposed a software/hardware framework for co-operating multiple robots performing tightly coordinated tasks, such as exploration of high-risk terrain areas. Yoshida (Yoshida, 2003) developed a system for sharing a common coordinate system so that multiple robots can be operated in the same environment. Unfortunately, most of all contributions on coordinative control of robots are only concerning identical type of robots with low-level human-robot intelligent interaction and coordination. Hence, this paper proposes a novel multi-robot management platform, called Knowledge-Based Intelligent Coordinative Network Platform (K-ICNP).
The K-ICNP was created with the intent of integrating a group of domestic robots as well as possible within its environment. It is a java-based platform aimed at providing 'symbiotic' integration to different robots with different features and ways of interacting with the user. These features may include vision, mobility, speech synthesis, textual input and output, providing the final user with a great range of capabilities. The platform aims at making these various features accessible in a unified way, be it for the third party programmer or for the final user, as well as providing inference mechanisms for mixing knowledge with empirical data.

Since the core of this kind of network platform is a knowledge model, a multi-robot system is firstly modeled by means of frame-based knowledge representation (Minsky, 1974). With this knowledge model, a multi-robot system can be clearly defined in the developed K-ICNP by use of XML format. K-ICNP can be responsible for exchanging messages of communication among robots and human, generating control instructions of robots, responding the information from each member of this system, scheduling behaviours of robots for each service to human, etc. It not only can explain the knowledge model of multi-robot system, but also integrate several techniques, such as distributed software agents, tele-operation via wireless network, etc. The coordinative control of robots can be therefore implemented according to human commands. The effectiveness of K-ICNP was verified by the experiments considering actual scenarios of activities of multi-robot system comprised of humanoid robots, mobile robot and entertainment robot.

2. Modeling of Multi-Robot System

2.1 Multi-robot system

A multi-robot system discussed in this research comprises many different types of robots for various purposes. According to their functions, all robots in the multi-robot system can be classified into two groups. One group is to communicate with human in order to obtain human requests through human-robot interface. The robots in this group have the ability of conducting intelligent interaction with human by means of vision, speech, body movement, etc. Another group is consisted of robots which will perform the task according to human commands. These robots have various specific functions, such as holding something, autonomously moving, etc. Therefore, such a kind of multi-robot system is a complex system, whose construction must be depended on the integration of various techniques, such as robotic technique, pattern recognition, software engineering, tele-operation, communication, etc.

2.2 Knowledge representation of multi-robot system

As mentioned in the Introduction, the core of K-ICNP is a knowledge model of multi-robot system. The knowledge model is constructed by use of the frame-based knowledge representation approach. The frame structure is originally proposed by M. Minsky (Minsky, 1974). Considering the complexity of the multi-robot system, the frame structure is re-defined comparing the original case (Zhang, 2005). The structure of a frame defined in this research is therefore consisted of the following items, such as Name, Type, A-kind-of, Descendants, Slots, etc. As the element of a frame, each slot has the items of Name, Type, Values, Conditions, etc. The meanings of each item in a frame are as given in Table 1 and the meanings of each item in a slot are given in Table 2.

With frame-based knowledge representation, features of different types of robots, activity of human-robot interaction, operations of robots, etc., in this multi-robot system can be defined by the following types of frames.

- **Robot frames**: are the frames for describing the features of various robots, including types, spatial positions, components, functions, etc.
- **User frames**: are the frames for describing different users who are distinguished by their names, occupations, etc. They can be classified into frames for new users and known users.

- **Behavior frames**: are the frames for describing the behaviors of robots. There have following three types of behavior frames. The first type is about the atomic actions of each type of robot, such as walking, sitting, standing, etc. The second type is about the combination behaviors of robots. Each frame describes a atomic action series for one purpose. The third type is to describe the intelligent interaction between human and robot, by means of vision, speech, etc. In addition, with the items of "Semantic-link-from" and "Semantic-link-to" in a frame, the relations among behavior frames can be defined, including relations of synchronization, succession, restriction, etc. Therefore, the activities of multi-robot system can be easily defined by use of behavior frames. Table 3 shows an example of the frame "AskUserName", which is to define the robot behavior to ask user's name.

Items	Meanings
Frame name	Identification for frame
Frame type	Type of frame
A-kind-of	Pointer to parent frame for expressing IS_A relation
Descendants	Pointer list to children frame
Has-part	Components of this frame
Semantic-link-from	Links from other frames according to their semantic relation
Semantic-link-to	Links to other frames according to their semantic relations
Slots	Components of the frame

Table 1. Meanings of each item in a frame

Items	Meanings
Slot name	Identification for slot
Role	Purpose of slot
From	Source of slot
Data type	Explain the attribute of information recorded into the value
Value	Slot value
Condition	Condition of slot
Argument	Argument for slot
If-required	If slot is required, check this item.
If-shared	If slot can be shared with other frames, check this item.
If-unique	If slot is unique from other slots, check this item.
Frame-related	A frame related with slot
Frame-list-related	Several frames related with slot
Default	If the slot value is not determined, the default value can be recorded. But the value and default value cannot be given at the same time.

Table 2. Meanings of each item in a slot

Items	Contents
Name:	AskUserName
Type:	Instance
A-kind-of:	Speech
Descendants:	NULL
Has-part:	NULL
Semantic-link-from:	NULL
Semantic-link-to:	NULL
Slot #1	
Slot name:	mUser
Data type:	Instance
Value:	NULL
Condition:	Any
Argument:	NewUser
If-required:	Checked
If-shared:	Checked
If-unique:	Checked
Slot #2	
Slot name:	mMouth
Data type:	Instance
Value:	NULL
Condition:	Any
Argument:	Mouth
If-required:	Checked
If-shared:	Checked
If-unique:	Checked
Slot #3	
Slot name:	mfunction
Data type:	String
Value:	sendmsg
Condition:	Any
Argument:	NULL
If-required:	No Checked
If-shared:	Checked
If-unique:	Checked

Table 3. Definition of frame "AskUserName"

Therefore, the knowledge model of a multi-robot system is consisted of various frames which forming a frame system. All these frames are organized by the ISA relation corresponding to the item of A-kind-of in a frame. The ISA relation means that the lower frame inherits all features of the upper frame, except some concrete features that are not defined in the upper frame. Some lower frames can be also regarded as instances of upper frames. Fig.1 illustrates the organization structure of all frames in the knowledge model of a multi-robot system.

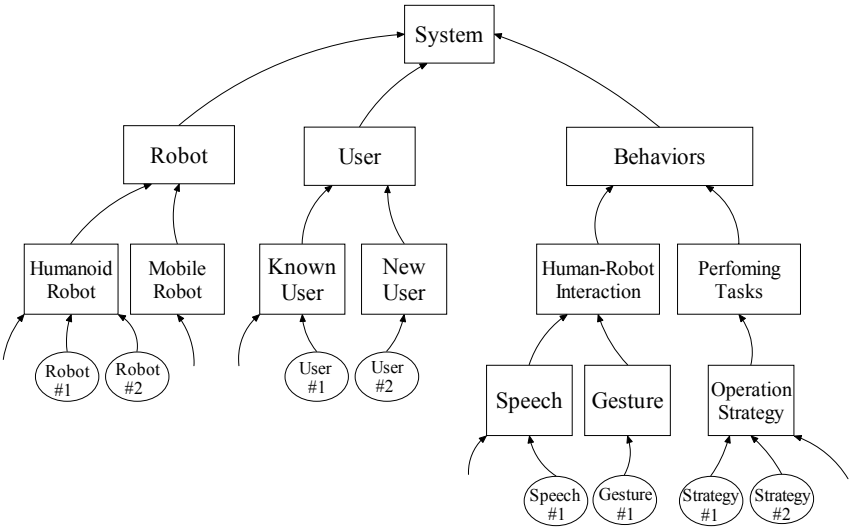


Figure 1. Organization structure of all frames in the knowledge model of a multi-robot system

3. Knowledge Model-based Intelligent Coordinative Network Platform

3.1 Features of K-ICNP

One of the key ideas underlying K-ICNP's design was abstraction. The platform aims at unifying different features and capabilities provided by different robots, and collecting them into a frame-based knowledge base where they will be processed in a uniform way. Abstraction is made of any hardware and software specificity, starting with networking. This implies, of course, that a bit of code specific to the platform has to be run on the robot itself, at least for interfacing the robot on a network. Indeed, the platform is network-based and designed to manage remote devices in a transparent fashion. More specifically, K-ICNP consists of a central management module, typically run on a computer, handling all of the management tasks as well as the intelligence aspects detailed later. This central module handles the task of agents synchronization over a TCP/IP network, reacts to input stimuli and distributes appropriate reaction directions. The attractive features of this network platform are "platform-independent" as existing robots and software modules often rely on different platforms or operation systems, "network-aware" as the modules must interact on a network, supporting "software agent" and being "user friendly". K-ICNP is targeted to be the platform on which a group of cooperative robots (or their agents) operate on top of frame knowledge.

The mechanism that transforms perceived stimuli into these appropriate reactions is the inference mechanism. As stated before, it is modelized using a frame-based knowledge representation. In this representation, functionalities offered by the different robots are represented by frame classes and processed the same way as any knowledge present in the knowledge base is. This way, linking features presented by a robot with semantic content for instance becomes a straightforward task involving frames manipulation only. The knowledge base has of course to contain prior knowledge about the world robots will roam in, but the system is also capable of filling the knowledge base with learned frames coming from user interaction, from simple questions asked by the system for example.

K-ICNP is a platform, and as such is designed to be used as a development base. As we saw before, the use of K-ICNP requires the embedment of some codes on the robots' side. This can be done using Agent Base Classes, which are java classes provided with the platform. More generally, the whole platform bears an easily extendible, plastic structure providing system designers with a flexible base. That is why K-ICNP includes a graphic knowledge base editor as well as a java script interpreter, the combination providing powerful behaviour control for the developer to use.
K-ICNP consists of six software components:

- GUI interface: It is a user-friendly graphical interface to the internal knowledge manager and the inference engines. It provides the users direct access to the frame-based knowledge.
- Knowledge database and knowledge manager: This is the K-ICNP core module that maintains the frame systems as Java class hierarchy, and performs knowledge conversion to/from XML format.
- Inference engine: Inference engine is to verify and process information from external modules that may result in instantiation or destruction of frame instances in the knowledge manager, and execution of predefined actions.
- JavaScript interpreter: It is adopted to interpret JavaScript code which is used for defining conditions and procedural slots in a frame. It also provides access to a rich set of standard Java class libraries that can be used for customizing K-ICNP to a specific application.
- Basic class for software agent: It provides basic functionality for developing software agents that reside on networked robots.
- Network gateway: This is a daemon program allowing networked software agents to access knowledge stored in K-ICNP. All K-ICNP network traffics are processed here.

In K-ICNP defines many kinds of Java classes representing the agents, such as server, user, robots, etc. The server agent serves as a message-switching hub, a center for relaying messages among robots and user agents. A user agent represents each user on the system, relays commands from the user to other agents, queries states of the robot agents, and provides the user with enough feedback information. A robot agent represents each robot under control. There are also some other software agents, e.g. a software agent to parse a sentence. K-ICNP generates the commands to robots relying on key words. We have developed a simple sentence parser for K-ICNP using the technique of Case Grammar taking into account the features of the operation of robot arm (Bruce, 1975).
All robots are connected with server computers in which K-ICNP is running, over a wireless TCP/IP network. Any information exchange between robots and K-ICNP are through wireless network. Therefore, tele-operation is an important means for implementing coordinative control of multi-robot system.

3.2 Definition of multi-robot system in K-ICNP

In K-ICNP, a multi-robot system is described in XML format according to its knowledge model. XML is a markup language for documents containing structured information (http://www.xml.com). With text-based XML format, frame hierarchy can be serialized and stored in a local file. It can be also transmitted over the network to a remote K-ICNP. In addition, the frame system can be illustrated in K-ICNP Graphic User Interface. Corresponding to XML file, there is an interpreter to translate XML specification into

relative commands. With the XML format, the knowledge model of multi-robot system can be defined in K-ICNP and the coordinative control of robots can be implemented as the following explanation. Table 4 is an example of frame definition in K-ICNP by use of XML.

```
<FRAME>
  <NAME>Greeting</NAME>
  <ISA>Speech</ISA>
  <ISINSTANCE>FALSE</ISINSTANCE>
  <SLOTLIST>
    <SLOT>
      <NAME>mUser</NAME>
      <TYPE>TYPE_INSTANCE</TYPE>
      <CONDITION>COND_ANY</CONDITION>
      <ARGUMENT>KnownUser</ARGUMENT>
      <VALUE></VALUE>
      <REQUIRED>TRUE</REQUIRED>
      <SHARED>TRUE</SHARED>
      <UNIQUE>TRUE</UNIQUE>
    </SLOT>
    <SLOT>
      <NAME>mMouth</NAME>
      <TYPE>TYPE_INSTANCE</TYPE>
      <CONDITION>COND_ANY</CONDITION>
      <ARGUMENT>Mouth</ARGUMENT>
      <VALUE></VALUE>
      <REQUIRED>TRUE</REQUIRED>
      <SHARED>TRUE</SHARED>
      <UNIQUE>TRUE</UNIQUE>
    </SLOT>
    <SLOT>
      <NAME>onInstantiate</NAME>
      <TYPE>TYPE_STR</TYPE>
      <CONDITION>COND_ANY</CONDITION>
      <ARGUMENT></ARGUMENT>
      <VALUE>Sendmsg(s.mMouth, “Hi! Nice to meet you. Welcome to visit my
room!”);</VALUE>
      <REQUIRED>FALSE</REQUIRED>
      <SHARED>TRUE</SHARED>
      <UNIQUE>TRUE</UNIQUE>
    </SLOT>
  </SLOTLIST>
</FRAME>
```

Table 4. Definition of frame “Greeting” in K-ICNP by use of XML format

3.3 Coordinative control of multi-robot system by means of K-ICNP

(1) Human-robot interaction

In order to implement coordinative control of multi-robot system according to human requests, human-robot interaction is an essential because the results of human-robot interaction can trigger the behaviours of multiple robots. Human-robot interaction can be implemented by many kinds of techniques, such as image recognition, speech, sentence parsing, etc. In K-ICNP, human-robot interaction is defined by use of behavior frames, such as greeting, face detection, etc. In the behavior frames, many independent programs for performing various functions of robots are adopted by the specific slot of "onInstantiate". If these behavior frames are activated, these functions will be called and robots will conduct their relative actions.

(2) Cooperative operation of robots

In K-ICNP, cooperative operations of multiple robots have been defined by behavior frames. Each behavior frame has a command or a command batch about actions of robots. The organization of these frames is based on the ISA relation so that the relations of robot behaviors can be known, which basically including synchronization, succession and restriction. The synchronization relation means that several robots can be operated simultaneously for a specific task. Their control instructions are generated referring to a same time coordinate. The succession relation means that one action of a robot should start after the end of another action of the same robot or other robots. The actions of several robots should be performed successively. The restriction relation means that as one robot is conducting a certain action, other robots can not be conducting any actions at the same time. With these behavior relations, even a complex task could be undertaken by cooperative operation of multiple robots. Besides, before activating a behavior frame, the conditions given in the slots should be completely satisfied. Therefore, we can define many safe measures to guarantee the reliability of robot behaviors, such as confirming the feedback of robot actions, checking the status of robots in real-time, etc.

The execution of these frames for cooperative operation of multiple robots is by use of the inference engines defined in K-ICNP. The inference engines are for doing forward and backward chaining. The forward chaining is usually adopted when a new instance is created and we want to generate its consequences, which may add new other instances, and trigger further inferences. The backward chaining starts with something we want to prove, and find implication facts that would allow us to conclude it. It is used for finding all answers to a question posed to the knowledge model.

In addtion, local control programs of robots are always put to the robot sides. When performing cooperative operation of multiple robots, the instruction from K-ICNP will be converted to the command of local control program by software agents so that local robot controllers can execute. Thus, as developing software agents the features of local controllers should be understood. But when defining any human-robot systems in K-ICNP, it is no need to take into account the local robot control programs.

Besides, when performing coordinative control of robots, feedback signals on activities of multi-robot system should be easily obtained. In the environment where user and robots are staying, several sensors (camera, etc.) can be set up to observe the actions of robots. Based on the user's judgment on the actions of robots, K-ICNP can adjust its control instructions or generate new tasks. Another way to get the feedback signals is by robots themselves. As robots ended their actions, they should automatically send back responses corresponding to their actions. Moreover, since there are many sensors in robot bodies, they can also send some signals detected by these

sensors to K-ICNP, which could be useful for K-ICNP to know the status of activities of multiple robots. These feedback signals can be defined in the frame as the conditions of slots. Finally, the coordinative control of multi-robot system can be carried out successfully.

4. Experiments

In order to verify the effectiveness of K-ICNP, experimental work was made by employing actual different types of robots, such as humanoid robots, mobile robot, entertainment robot, etc., meanwhile considering actual scenarios of activities of multi-robot system.

4.1 Experimental components

In the experimental work, the following four types of robots are employed, as illustrated by Fig.2.

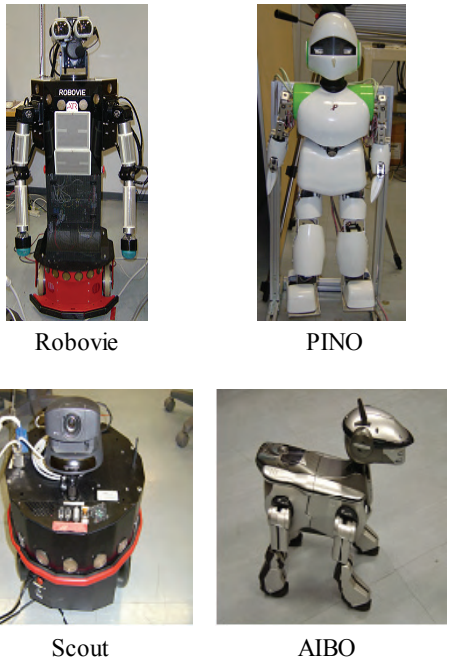


Figure 2. Robots employed in the experimental work

- **Robovie:** is a humanoid robot with human upper torso placed on an ActivMedia wheel robot. The movements of both arms and the head can be controlled from the software. It has two eye cameras, which connect through a video source multiplexer, to the frame grabber unit of a Linux PC inside the ActivMedia mobile unit, and a speaker at its mouth. Thus, Robovie can interact with user by gesture of its arms and head, or by using voice, like a kind of autonomous communication robots. A wireless microphone is attached to Robovie head so that we can process user voice information as well. Since Robovie has capability to realize human-robot communication, therefore, in this system Robovie plays the role for human-robot interaction. We also installed some programs for human-robot interface in the Linux PC of Robovie by means of the techniques of

image analysis, speech, etc., such as face processing module using the algorithms described in (Turk, 1991)(Rowley, 1998), the festival speech synthesis system developed by CSTR (http://www.cstr.ed.ac.uk/projects/festival), etc.

- **PINO:** is another kind of humanoid robots. It has 26 degrees of freedom (DOFs) with the low-cost mechanical components and well-designed exterior. It can act stable biped walking, moving its arms and shaking its hands like human.
- **Scout:** is an integrated mobile robot system with ultrasonic and tactile sensing modules. It uses a special multiprocessor low-level control system. This control system performs sensor and motor control as well as communication. In Scout, there are differential driving systems, 16 sensors, 6 independent bumper switches, CCD camera, etc.
- **AIBO:** is a kind of entertainment robots. It can provide high degree of autonomous behavior and functionality. In our experimental system, we use AIBO ESP-220, which is able to walk on four legs. It has a total of 16 actuators throughout its body to control its movements, and 19 lights on its head, tail, and elsewhere to express emotions like happiness or anger and reactions to its environment.

All robots are connected with K-ICNP via wireless TCP/IP network.

4.2 Scenario of task

With this multi-robot system, a simple task can be fulfilled. The scenario of this task is shown in Table 5.

User *A*	(User A appears before the eye cameras of Robovie.)
Robovie	(Robovie is looking at the user *A*'s face and trying to recognize it.) How are you! I have never seen you before. What is your name?
User *A*	My name is XXX.
Robovie	Hi, XXX. Nice to meet you. Welcome you to visit my room. (Robovie *A* shakes its both hands.)
Robovie	This is my friend, PINO.
PINO	(PINO walks to user *A* and shake its hand with user *A*.)
Robovie	What do you want to drink?
User *A*	Tee, please.
Scout	(Scout brings a cup of tee for user *A*.)
Robovie	This is a robot dog AIBO. Please enjoy yourself with it.
AIBO	(AIBO walks to user *A* and lies down near user *A*.)
	...

Table 5. A scenario of multi-robot system

4.3 Modeling of multi-robot system and its definition in K-ICNP

With frame-based knowledge representation, this multi-robot system can be modeled and defined in K-ICNP. Fig.3 is the K-ICNP knowledge editor showing the frames hierarchy for the multi-robot system. Each frame is represented by a click-able button. Clicking on the frame button brings up its slot editor. Fig.4 is a slot editing table for "AskUserName" frame. Each row represents a slot in this frame. For this frame, if two instances ("NewUser" and "Mouth") are set up, this frame will be created and execute the JavaScript codes written in "onInstantiate" slot. In this slot, special functions "sendmsg()" for Robot A speech is defined as the values of this slot.

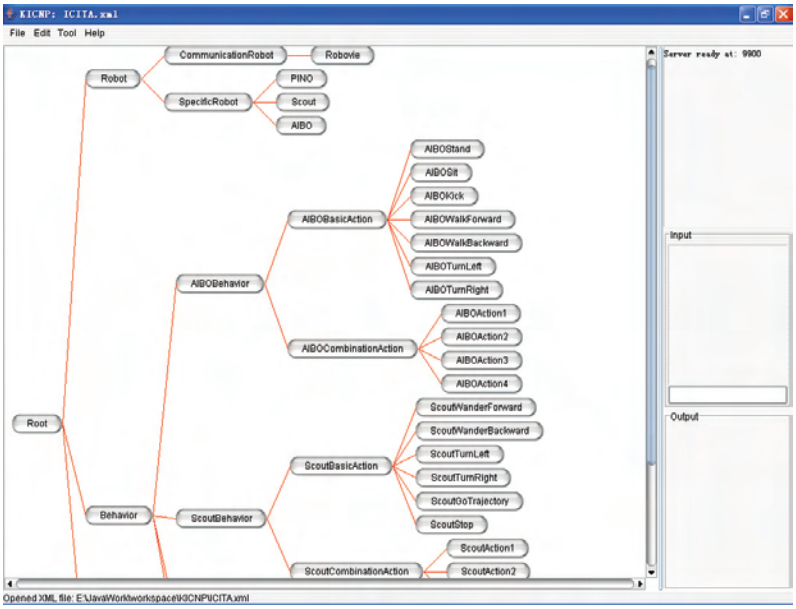


Figure 3. K-ICNP knowledge editor showing the frames hierarchy for multi-robot system

Property of AskUserName

File Edit

Name	Type	Value	Condition	Argument	Required	Shared	Unique
Name	String	AskUserN...	ANY		☐	☐	☐
mUser	Instance		ANY	NewUser	☑	☑	☑
mMouth	Instance		ANY	Mouth	☑	☑	☑
onInstantiate	String	sendmsg(...	ANY		☐	☑	☑

Figure 4. Slot editing table of "AskUserName" frame

4.4 Implementation of coordinative control of robots

Based on the above definition, the cooperative operation of multiple robots can be carried out according to the scenario. The cooperative operation of multiple robots in multi-robot system is firstly activated by human-robot interaction. For example, the human-robot interaction conducted by Robovie can be linked with frames of "GotNewName", "FirstMeet", "FaceDetection" and "Greeting". When a face is detected by Robovie, an instance of "User" frame is created. This instance will be checked whether it belongs to any subclasses of "KnownUser" classes. If there is a match and this face is a known user, the "Greeting" behavior will be fired to greet the user. Otherwise, the new user instance will be treated as "NewUser" and "FirstMeet" behavior will be triggered to ask user of this name. The user's response will be sent to "GotNewName" frame which will register the new name as a sub-frame of "KnownUser".

Based on human-robot interaction, cooperative operation of multiple robots can be implemented. For example, in this scenario there are several different tasks performed by different robots. Each task is fulfilled by several actions of each robot. Each action of robot is conducted as the following example. Concerning about the "AIBOAction1" frame, if three instances ("RobovieToAIBOCommand1", "Mouth" and "AIBO") are set up, the first action of

AIBO will be performed with the corresponding functions existed inside of AIBO. When making the connection between K-ICNP and Robovie, "Mouth" frame can be automatically activated. As Robovie is performing the interaction with user, "RobovieToAIBOCommand1" frame can be activated. Before performing actions of robots, it needs to indicate the operation object of robots. With interpreting the speech of Robovie, "AIBO" frame can be activated. Then, the first action of AIBO can be conducted. Similarly, other actions of robots can be conducted. Therefore, multiple robots in a multi-robot system can perform more complex planned activities for users.

5. Conclusions

The Knowledge Model-based Intelligent Coordinative Network Platform for multi-robot system is proposed in this paper. In K-ICNP features of robots in multi-robot system as well as their operations can be described by frame-based knowledge representation. By means of K-ICNP, coordinative control of multi-robot system can be implemented. The effectiveness of K-ICNP was verified by the experimental work considering actual scenarios of activities of multi-robot system comprised of humanoid robots Robovie and PINO, mobile robot Scout and entertainment robot AIBO.

Further developments will improve the system's capacity of learning from possibly incomplete or erroneous data, of guessing optimal strategies and responses from prior knowledge and will extend the system's overall capacity to deal with complex orders. Hardware abstraction will be brought to the next level and standardized via the use of UPnP as a means for robots, even unknown, to expose their available features.

These improvements will make the platform a powerful tool on which to base practical multi-robot applications, another step towards the goal of real symbiotic robotics.

6. References

Bruce, B. (1975). Case systems for natural language. *Artificial Intelligence*, Vol. 6, pp.327-360.

Huntsberger, T.; Pirjanian, P. et al. (2003). CAMPOUT: a control architecture for tightly coupled coordination of multi-robot systems for planetary surface exploration. *IEEE Transactions on Systems, Man and Cybernetics*, Part A, Vol. 33, No. 5, pp.550-559.

Minsky, M. (1974). *A Framework for representing knowledge*, MIT-AI Laboratory Memo 306.

Rowley, H. A.; Baluja, S. and Kanade, T. (1998). Neural network-based face detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, Vol. 20, No. 1, pp.23-38.

Turk, M. A. and Pentland, A. P. (1991). Face recognition using eigenfaces. *Proceedings of Eleventh International Conference on Pattern Recognition*, pp.586-591.

Ueno, H. (2002). A knowledge-based information modeling for autonomous humanoid service robot. *IEICE Transactions on Information and Systems*, Vol. E85-D, No. 4, pp. 657-665.

Yoshida, T.; Ohya, A. and Yuta, S. (2003). Coordinative self-positioning system for multiple mobile robots. *Proceedings of 2003 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, Vol.1, pp.223-227.

Zhang, T. and Ueno, H. (2005). A frame-based knowledge model for heterogeneous multi-robot system. *IEEJ Transactions on Electronics, Information and Systems*, Vol. 125, No. 6, pp.846-855.

http://www.cstr.ed.ac.uk/projects/festival.

http://www.xml.com.

16

Mobile Robot Team Forming for Crystallization of Proteins

Yuan F. Zheng[1,2] and Weidong Chen[2]
[1] *The Ohio State University,* [2] *Shanghai Jiaotong University*
[1]*U.S.A.,* [2]*China*

1. Introduction

Following the completion of human genome sequencing, a major focus of life science has been the determination of the structure of proteins which genome codes for. This is a huge undertaking since there are between 30,000 to 200,000 proteins in the human body, and the structure of proteins is most complicated among all the molecules existing in an organism. The only reliable method for determining the atomic structure of a protein is x-ray crystallography (Luzzati, 1968; Caffrey, 1986) which requires the protein of interest to be purified and solubilized in an appropriate medium and to form a well ordered crystal through incubating. Scientists in biology have long considered crystal growing more an art than science because no theory can completely explain the process (Bergfors, 1999), and appropriate physical and chemical conditions supporting the growth have to be discovered by exploring a huge combinatorial space, which is tedious and time consuming (Cherezov, et al., 2004).

In the area of membrane proteins, for example, an effective method called *in meso* was introduced some years ago by Landau and Rosenbusch (Landau and Rosenbusch, 1996). The basic recipe for growing crystals in meso includes three steps: a. combine 2 parts of protein solution/dispersion with 3 parts of lipid (monoolein), b. overlay with precipitant, and c. incubate at 20 °C. Protein crystals can then form in hours to days. The method involves combining protein with lipid to achieve spontaneous formation of cubic phases, and dispensing the cubic phase into a small container for mixing with precipitant and for incubation (Cheng et al., 1998; Luzzati, 1997). The ratio between the amounts of the protein and the lipid, the volume and type of the precipitant, and the temperature and duration of incubation all affect the quality of the crystal.

The process of protein crystallization poses a challenging question: how do randomly distributed protein molecules move to form the uniform structure? Various theories have been generated aiming to discover optimal physical and chemical conditions which are most conducive to protein crystallization (Caffrey, 2000; ten Wolde and Frenkel, 1997; Talanquer and Oxtoby, 1998). Researchers are still curious about motion trajectory which individual protein molecules take to eventually form the crystal. Hope such an outstanding can help scientists reduce the time and effort in searching for an optimal condition.

In this chapter, we propose to study the motions of the proteins by a completely new approach which is robotic team forming by proper motion trajectories of mobile robots. The

idea is inspired by research activities of two different disciplines: biology and robotics. In biology, scientists are amused by important functions which proteins play in a completely autonomous way which is similar to autonomous robots working collaboratively to perform useful functions. In this regard, biologists even call proteins nature's robots in a recent book (Tanford and Reynolds, 2004). In robotics, robot team forming is for multiple mobile robots to establish a robotic pattern which is optimal for performing a given task (Parker et al., 2005; Ceng et al., 2005). By combining the two perspectives, we consider each protein an autonomous robot, and protein crystallization a process of robot team forming. This analogy is reasonable because crystal is a set of orderly connected proteins, and crystallization needs the proteins to form a symmetrical pattern. Our goal is to give the crystallization process a mathematical description similar to that of path planning for mobile robots such that the process has more flavor of science.

Robot team forming for performing robotic tasks has been studied by many works in recent years. Passino, Liu, and Gazi investigated control strategies for cooperative uninhabited autonomous vehicles (UAV) by modeling them as social forging swarms, and developed conditions which led multiple UAVs to cohesive foraging (Gazi and Passino, 2004; Liu and Passino, 2004). Chio and Tarn developed rules and control strategies for multiple robots to move in hierarchical formation based on a so-called four-tier hybrid architecture (Chio and Tarn, 2003). Some other works have used a leader-follower approach (Desai et al., 1998; Leonard and Fiorelli, 2001; Sweeney et al., 2003) to control the team forming in which the motion is specified for only one robot, called leader, and the others just follow. In (Fregene et al., 2003), Fregene et al. developed a pursuit-evasion scheme to coordinate multiple autonomous vehicles by modeling them as classes of hybrid agents with certain level of intelligence.

Robot team forming has its root in the swarm by large number of small animals such as bees and birds, which continues to be a topic of study in recent years. For example, Li et al. (Li et al., 2005) studied stable flock of swarms using local information by which a theory of decentralized controller is proposed to explain the behavior of self-organized swarms of natural creatures. Secrest and Lamont used a Gaussian method to visualize particle swarm optimization (Secrest and Lamont, 2003). Liang and Suganthan (Liang and Suganthan, 2005) developed a new swarm algorithm to divide the whole population of individual entities into small swarms which regroup frequently for information exchange between the groups. For the latter purpose, Das, et al. (Das et al., 2005) developed an efficient communication protocol between mobile robots in motion to form a team of desired pattern.

In reviewing the literature, we found that existing strategies for robot team forming are very complicated which employ sophisticated schemes such as the Lyapunov stability theory (Gazi and Passino, 2004; Liu and Passino, 2004) to control the motions of mobile robots or to explain the swarm of living agents. These strategies are not suitable for solving the current problem because protein molecules are not able to plan complicated motions. In addition, all the proteins in the medium are equal and cannot have different rules of motion. These two practical issues impose significant constraints to any theory which can explain the crystallization process as well as be acceptable to the biology community. In this chapter, we attempt to present such a theory which meets the two constraints.

We borrow the technique from robotics to develop the theory in two steps. The first step is path planning which defines the motion trajectories of the protein robots for forming the team (crystal). The second is robot-servoing which drives the protein robots to follow the

planned path. In the first step we plan a simple and local path which is realistic for the protein robots to take, while in the second we define a control law which governs the planned motion of the protein robots, and further prove the stability of the motion. To guarantee the control law, we find a natural force, the van der Waals force, to drive the protein robots. Our major contribution is to define a set of rules which is realistic yet effective for the protein robots, and prove that such a trajectory is naturally possible.
This chapter is organized as follows. In the next section, we develop an effective path which is able to guide the proteins to form the crystal. In the third section, we examine the protein dynamics to see how the proposed motion is physically possible and stable from the control theory point of view. Simulation and experimental results are presented in the fourth section to verify our theory, which is followed by the section of conclusions.

2. Protein Path Planning for Crystallization

We first need to define the rules which govern the trajectory of the protein robots in the process of team forming, and then analyze the stability of the motion as well as the shape of the team formed by the protein robots.

2.1 The Model of the Motion

Path planning in robotics is to plan the motions of individual robots such that a particular pattern of the team can be formed. Using the same technique we define the path that each protein robot should take to eventually form the crystal. To solve this problem, we need to understand the intrinsic structure of crystal. A crystal is a substance which has ordered connections of its composing elements such as atom, molecule, or ion which form symmetrical 3D lattice in its structure (Pollock, 1995). Depending on the way and extent of symmetry, there are many kinds of systems in crystal. The most symmetrical possible structure is called isometric system which comprises three crystallographic axes of equal length and at right angles to each other. Other systems are symmetrical in different ways, but all has a uniform structure which is the key element for the success of x-ray crystallography.
Each atom or molecule has its own structure which is isometric in many cases such as diamond and most mineral crystals. Proteins on the other hand are structurally complicated molecules, each type of which has a unique 3D shape consisting of amino acids folded or coiled into specific conformations (Campbell and Reece, 2004), which is not isometric, i.e., not symmetric in its structure. Thus, both position and orientation of the protein robots must be right in the team forming process; otherwise, symmetrical structure of the crystal is not possible. For convenience we model each protein as a rigid body shown in Fig. 1 and limit our study to 2D. The result, however, can be extended to 3D without much difficulty.

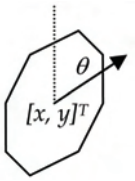


Figure 1. Modeling the position and orientation of a protein

We further use three parameters to describe the state of a protein robot, two for the position $[x, y]^T$ and one for the orientation θ of the protein robot. The mass and inertia of the protein are not relevant when we discuss the kinematics of the robot, which however will be defined later when the dynamics of the protein robot is to be studied. The detailed structure of the protein, however, is not considered here.

As mentioned in Introduction, *simplicity* and *locality* are two key features for the motion of protein robots, for which we propose three rules to dictate the process of path planning of the protein robots.

***Rule 1*: Each protein searches for its nearest neighbor**. To form a team, each protein has to position itself in a correct way with respect to its team members. A protein robot is not intelligent enough to determine where many other proteins are but is assumed able to find its nearest neighbor. This assumption is reasonable because a protein robot will always be attracted to its nearest team member by the natural force existing between them. This natural force will be further discussed later in this chapter.

***Rule 2*: Once the nearest neighbor is found, the protein moves towards it by taking the shortest path until a predefined distance is reached.** This rule is reasonable again as the natural force mentioned in Rule 1 drives the two molecules until it becomes null.

***Rule 3*: Multiple proteins may form a super-protein which eventually becomes crystal.** As protein robots move together following the first two rules, more and more proteins are bound together to form what we call super-protein. The super-proteins will continue attract proteins according to Rules 1 and 2 until no free proteins are available in the medium when the super-proteins become crystal. The super-proteins, however, may or may not attract each other to form a larger super-protein depending on the size of the super-protein.

The above three rules generate straightforward trajectory for the protein robots which is the shortest path towards the nearest neighbor. By such a motion, each protein robot does not move according to a *global* strategy, but only a *local* path to reach its destination until all the "natural" forces disappear and the crystal is formed. Now the question is whether the simple and local motions can make the protein robots form the desired shape of the team, i.e., the crystal. For that purpose, the following analysis is in order.

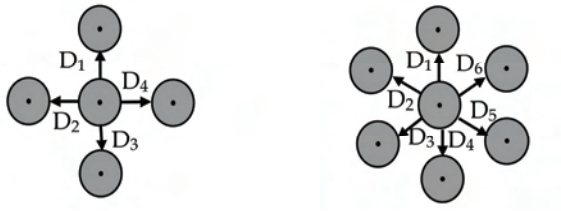
The simple and local motion of the protein robots can be described by the following equation:

$$\dot{X}_i = C[(X_{in} - X_i) - D_p] \tag{1}$$

where $X_i = [x \ y \ \theta]^T$ represents the position and orientation of the current protein, X_{in} represents the same of the nearest neighbor, and C is a coefficient which is a function of time, i.e., it varies as the two protein robots getting closer. Equation (1) reveals that when the distance between the current protein robot and its nearest neighbor is not equal to D_p, which is the desired distance between any two proteins in the crystals, the robot will keep moving until that is true. The velocity of the robot is proportional to the difference between the actual and desired distances. Note again that the mass and inertia of the protein are not involved in (1) because the analysis here is on kinematics of the robots.

The value of D_p with $p = 1, 2, 3 .. n$ depends on the structure of the crystal and so does n. For example, if the crystal has a rectangular grid as shown in Fig. 2a, $n = 4$ and each D_p is shown as a vector. For the structure as shown in Fig. 2b, $n=6$ and the D_p is significantly different from those in Fig. 2a. It is unknown in which pattern of the grid the proteins choose to form the crystal, but the x-ray crystallography will reveal the structure of the protein so long

as the grid is uniform in both position and orientation (Taylor and Lipson, 1965). Note that in Fig. 2, only the positions of the protein robots are shown, but the orientation of all the proteins must be uniform for forming the crystal.



a) b)

Figure 2. Structures of protein crystal with different D_p

From the above discussion, one can see that each protein will have to select a D_p for its local motion which appears to violate the simplicity rule. Fortunately it is not a complicated matter because D_p is always determined by where the nearest neighbor is. That is, once the nearest neighbor is determined, D_p is automatically directed from the position of the current protein robot to that of the neighbor.

Not all the proteins move simultaneously together to form a single large crystal at once. Some of the proteins will come together to form a small group earlier than others. We call these groups super-proteins. The super-proteins will continue to attract its nearest proteins (or to attract other super-protein) to form an even large group. Such a process will continue until no free proteins are available. A super-protein may or may not attract other super-proteins to form a larger super-protein, depending on the sizes of the two super-proteins when they meet which will be further explained later.

2.2 Stability and Shape Analysis

The motion of the protein robots as defined by (1) is extremely simple. The question is if that motion is stable to generate the desired structure of the crystal, for which we need to perform stability analysis by which we want to prove that all the protein robots come to rest at constant positions based on the motion trajectory defined by (1). To do so, we take the derivative of (1) with respect to time and get the following differential equation:

$$\ddot{X}_i = \dot{C}[(X_{in} - X_i) - D_p] + C(\dot{X}_{in} - \dot{X}_i). \tag{2}$$

Without loss of generality consider X_{in} to be constant. Equation (2) can be expressed as:

$$\ddot{X}_i + C\dot{X}_i + \dot{C}X_i = \dot{C}(X_{in} - D_p). \tag{3}$$

Equation (3) now describes the motion of the protein robot assuming it takes the trajectory defined by (1) which on the other hand is hypothesized according to the three rules.

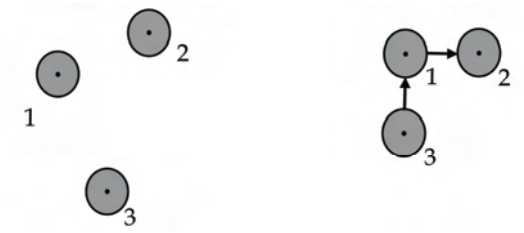
From the classical control theory (D'Azzo et al., 2003), equation (3) represents a second order system which is unconditionally *stable* provided that $\dot{C}$ and C are positive definite. That condition can be satisfied by the inter-molecular forces, which will be shown true when we discuss the dynamics of the protein robot.

Equation (3) shows that two protein robots move towards each other to eventually bind at the desired distance D_p, which has been proved stable. It is still not convincing that massive amount of proteins taking the same type of motions will bind and eventually form the crystal. Here we will use a technique called *induction proof* to show that multiple protein robots can bind to form crystals by performing the *local* motion of (1). Induction is a powerful tool in discrete mathematics which is suitable for proving properties associated with natural numbers, *0, 1, …* (Meyer and Nagpal, 2002). If a property is true for *0* or *1*, one can assume that it is true for any number *k*. Then the induction is to show that the same property is true for number *k+1*. If the latter is successful, one can claim that the proof of the property is completed. Using the method of induction proof, we propose and prove the following theorem:

Theorem*: a number of proteins move with the local motion as defined by (1) will come together to form one or more crystals with the desired structure.*

Proof: The induction proof takes the following three steps:

Step 1. Let the first protein be numbered n_p=*1*, and the second n_p =*2*. According to (3), we obtain a single pair of proteins which are bond together. Now let n_p =*3*, i.e., there are three proteins as shown in Fig. 3a. According to the rules of motion, two closest proteins in the medium move towards each other until the desired distance (relative position and orientation) is reached. Suppose that the distance between proteins *1* and *2* is shortest, and that between *1* and *3* is next. Proteins *1* and *2* will move together regardless of the position of protein *3*, and proteins *1* and *3* will move together regardless of *2*. Consequently, the three proteins will form the group as shown in Fig. 3b.



a) b)

Figure 3. Three proteins come together to form the team

Step 2. Assume that *k* proteins form a super-protein.

Step 3. Let the largest n_p = *k+1*. Assume *k* of the *k+1* proteins form a super-protein before the $(k+1)^{th}$ protein. The last protein will still go to the super-protein formed by the first *k* proteins until it is bonded with the super-protein, which is also defined by the rules of the motion. It should be mentioned that multiple bindings could occur simultaneously, and all the bindings continue until protein molecules in the medium are exhausted. Therefore, multiple pieces of crystal could form at the end of the process.

The above proof only shows that the proteins can form crystal with a uniform *structure*. It leaves the *shape* of the crystal undetermined. By using equation (1), the final shape of the crystal will depend on the distribution of the proteins in the medium space before the crystallization process starts. For example, if the proteins are evenly distributed within a circular area, the final shape is likely to be a circle or a polygon as shown in Fig. 4a. If the proteins are distributed over a long and narrow area as shown in Fig. 4b, the final shape is

likely to be a stick. Then a challenging question follows: can the simple and local motions of the proteins form any shape of crystal by positioning the proteins accordingly if possible? The answer to that question is not that straightforward. We argue that some, not any, shapes can be formed by the protein robots. Consider a group of robots which are moving in the direction N as shown in Fig. 5a. If we limit D_p to just one possibility (Fig. 5b) and modify the three rules of motion to the following, the robots will form a line as shown in Fig. 5b.

Rule 1: Each robot searches for its nearest neighbor within its field of view, and becomes its follower once found, and each robot has only one follower.

Rule 2: Once the nearest neighbor is found, the robot moves towards it taking the shortest path until the desired distance is reached. The robot will then maintain the distance until the team needs to reform for a new pattern.

Rule 3: Multiple robots may form a sub-team which may continue to attract still isolated robots until none is available.

Comparing the new set of rules with the earlier set, one may find that the motion of the robot has become more complicated. The nearest neighbor is no longer chosen blindly but under certain constraints which can only be achieved by more "intelligent" organisms. Protein molecules as mentioned earlier do not possess such intelligence; therefore, the shape can only be random, depending on how the proteins are distributed in the medium. Recall the three steps for crystallizing membrane proteins which involve combining 2 parts of protein solution/dispersion with 3 parts of lipid (monoolein). How well the protein solution is mixed may affect the distribution of the proteins, and eventually the shape of the protein crystal.

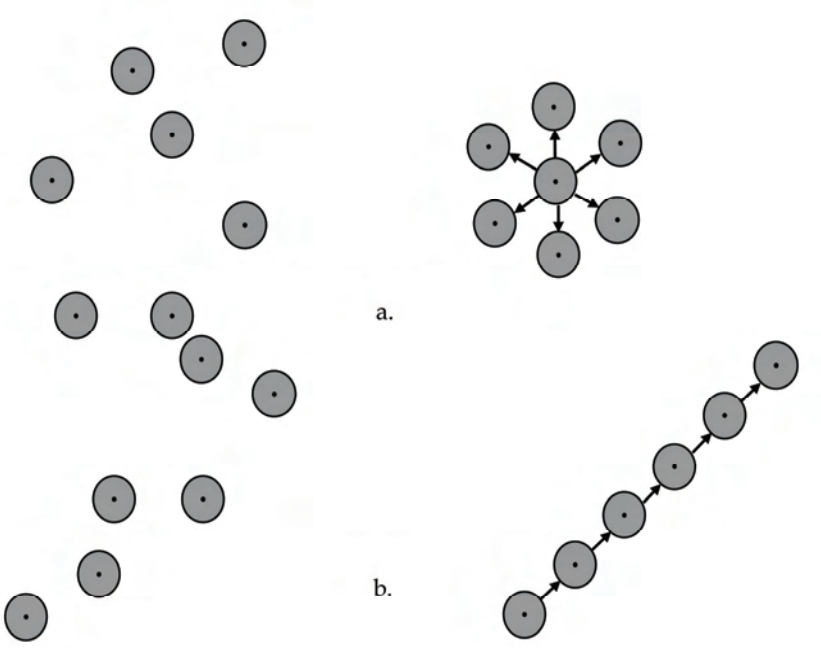
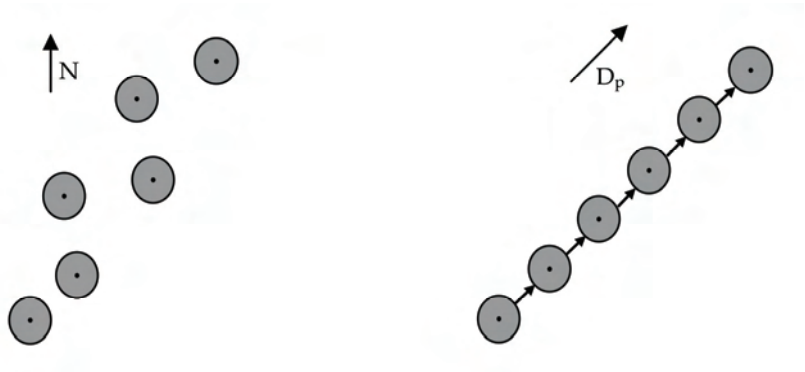


Figure 4. Global shape formation of the protein robots



a) b)

Figure 5. Limited local motion forms a straight line

3. Driving Force of the Protein Robots in Crystallization

We have proved that simple and local motions of individual proteins can move together to form crystals of uniform structure. The idea is borrowed from the strategy of path planning in robotics. Robot path planning is an issue of kinematics which involves no mass and inertia of the robots. To guarantee the proposed motion, regardless how simple and local it could be, we still need to study the dynamics of the protein robots. In robot team forming, the driving force is regulated by some control laws reacting to the position and velocity of the robot. In protein crystallization, we can only rely on natural forces which drive the protein robots to follow the planned path. Ultimately, we need to understand how the proposed motion is achieved physically in the medium, for which the key is the interactive forces between the protein molecules. At the molecule level, the van der Waals force is the only natural force which can play such a role.

3.1 The van der Waals Force and the Local Motion

According to the basic properties of matters, we understand that there exist physical forces of attraction and repulsion between molecules which are responsible for the motion and cohesion of molecules in the forming of crystals (Pollock, 1995). These forces are called van der Waals forces and act only over a relatively short distance between two molecules. The combined effect of the attraction and repulsion forces generates a potential which can be expressed by the following equation:

$$E_p = -\frac{A}{r_{ij}^m} + \frac{B}{r_{ij}^n} \tag{4}$$

where r_{ij} represents the distance between the two molecules, m, n, A, and B are parameters depending on the properties of the proteins and the medium which the proteins are in. It should be mentioned that the first term on the right-hand side represents the attraction potential E_{att}, and the second term the repulsion potential E_{re}. Also m is smaller than n. The combined potential of E_{att} and E_{re} can be seen in Fig. 6 which shows little repulsion force when two molecules are not close.

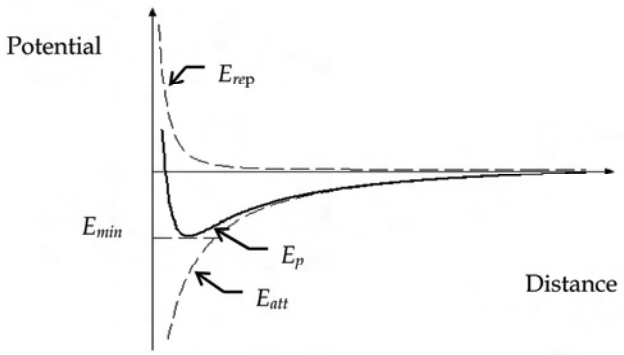


Figure 6. van der Waals forces generate potential between two protein molecules

In the combined curve of E_p, there is a point of minimal potential E_{min} which is called dissociation energy, and the distance between the two proteins at that point is called the length of bond. Since the potential is minimal, the molecules are forced to stay at that distance, which is similar to a spring connecting two small balls together. When the two balls move away or towards each other, the force of the spring will bring them back. Only when the potential between the two balls is minimal, i.e., at the distance which the spring produces no force, the balls remain stationary. In the protein crystallization process, protein molecules are far apart to start with, and the attraction potential dominates to bring the molecules together. The two molecules have no chance to be closer than the length of bond since the repulsion potential increases much faster than the attraction potential. Consequently, we can claim that the length of bond r_0 is identical to D_p, i.e.

$$r_0 = D_p. \tag{5}$$

Using D_p as the equilibrium point, the interaction force between the two protein molecules as described in Fig. 6 can be expressed as

$$F = C_{fp}[(X_{in} - X_i) - D_p] \tag{6}$$

where C_{fp} is positive definite because F monotonically increases in the positive direction when $(X_{in}\text{-}X_i)$ is greater than D_p and negative otherwise. Taking into consideration the nonlinearity of the relationship, it is reasonable to add a second order term to (6), i.e.,

$$F = C_{fp}[(X_{in} - X_i) - D_p] + C_{fv}(\dot{X}_{in} - \dot{X}_i) \tag{7}$$

where C_{fv} is also positive definite. By applying the Newton dynamic equation to the protein robot, one obtains:

$$\ddot{X}_i = m^{-1}F = m^{-1}C_{fp}[(X_{in} - X_i) - D_p] + m^{-1}C_{fv}(\dot{X}_{in} - \dot{X}_i) \tag{8}$$

where m is the mass and inertia of the protein molecule. One can see that equation (8) is equivalent to equation (2). That is, in (2) the derivative of C is replaced by $m^{-1}C_{fp}$, and C is replaced by $m^{-1}C_{fv}$. Consequently, the van der Waals forces warrant a stable process of crystallization. Since protein molecules are separated and distributed in the chemical

medium, only the attraction forces are effective when the crystallization process starts. Clearly, if the protein molecules are too few and too far apart, the van der Waals forces may not be large enough to bring them together. Therefore the density of the protein molecules in the medium is an important factor for successful crystallization.

It should be noted that there exists the van der Waals force between any pair of proteins so long as they are close to each other. For simplicity, we consider all the forces to be insignificant except the one from the nearest neighbor. This assumption is reasonable because the van der Waals force vanishes quickly as the distance between two proteins becomes large. Another interesting issue is what happens when two neighboring protein molecules are exactly the same distance from the current molecule. According to the rules of the motions, the current protein robot will have to make a random choice if the path is artificially planned. Naturally, two equal van der Waals forces will be applied to the protein simultaneously, and the protein will be moved by the combined force, not towards any of the two proteins. However, if there is a random factor in the medium which makes the protein a little closer to one of the two proteins, the van der Waals force from the closer one will dominate, and the one neighbor rule resumes. Since the probability of same distance is extremely small, it is not considered in the rules of motion.

The property of the van der Waals force can further explain why super-proteins may or may not attract each other. When the size of the two super-proteins is too large, the distance between them prevents a large enough van der Waals force to attract them together. On the other hand, when the size of the super-protein is still comparable with a macromolecule (Campbell and Reece, 2004), the distance allows a large van der Waals force which drives the two super-proteins together for binding. Furthermore, the proteins on the surfaces of the super-proteins have close contact with free proteins in the surrounding medium space and are able to attract them to the super-proteins. As a result, a super-protein is able to grow in two ways so long as free proteins in the chemical medium are not exhausted.

3.2 The Issue of Orientation

We have so far not touched the issue of orientation of the protein robots, which should all be the same for forming uniform and well-ordered structure of the crystal. For the purpose of path planning, we only need to change the second rule of the rules of motion as defined in Section 3 slightly. That is, when the nearest neighbor is found, the protein robot moves towards it taking the shortest path and aligning in the same direction as the neighbor in the three dimensional space. The question is how this can be physically achieved. Our hypothesis is that both the polarization of the protein and the van der Waals force play a role in the alignment.

Proteins are macro molecules consisting of a chain of amino acids which are in turn composed of multiple atoms of hydrogen, carbon, oxygen, and nitrogen, etc. A large number of electrons associated with the atoms move continuously between the atoms of the amino acids. They can pile up on one side of the protein that creates negative polarity on one end, and positive on the other known as polarization as shown in Fig. 7. The negative end attracts the positive end of another protein by the van der Waals forces, and vice versa. These forces create the local motions as discussed in the previous sub-section.

In addition to the attraction force, the dipole of the proteins enables an external electric field to align the proteins in the same direction as the field. Thus, the orientations of all the polarized proteins, as defined by the directions of the vector which goes from the negative

end to the positive as shown in Fig. 8, become identical automatically in a medium which has such an electric field. We hypothesize that the most qualified candidate for generating the electric field in the *in meso* process, for example, is the lipid membrane in which the membrane proteins are embedded at a natural status. Each lipid molecule has a hydrophilic head and a hydrophobic tail, and the heads attach to each other and the tails aligned on one side (Caffrey, 2000). In the crystallization process, the membrane proteins are washed free by certain detergent from the lipid membrane. The lipid molecules in the medium continue to generate a strong and uniform electric field to align the protein molecules in the same direction. Thus a uniform and well-ordered structure of the crystal becomes possible.

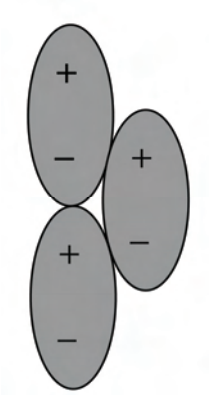


Figure 7. Polarized proteins attract and align with each other

The electric field as just mentioned only forces the polarizing vector to align in the same direction as the electric field. The protein robots can still rotate about the axis of the vector as shown in Fig. 8. One needs another force to lock the protein robot at an angle which is identical to all the proteins. We consider both the structure of the protein and the van der Waals force to be responsible for this alignment. The protein as a macro-molecule has complicated structure in comparison with atoms and small molecules. Thus, when the van der Waals forces attract two proteins together, the energy is not always minimal due to the difference in the angle of rotation, which leads to an unstable binding. Only when the two protein robots are aligned correctly in all the directions, the two proteins are locked into each other to generate the minimal energy possible.

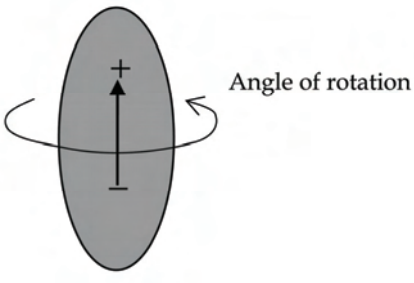


Figure 8. The protein robot can rotate about its axis of polarized vector which is aligned with the electric field

4. Simulation and Experimental Results

We use TeamBots, a multi-robot software, developed by the CMU MultiRobot Lab as an environment for simulation of protein crystallization. Each protein robot is represented by a black circle which is randomly distributed in the medium space. For simplicity, robot orientation is not considered in the simulation run. The protein robot is very small with a radius of 5 nm which is similar to the size of a true protein (Campbell and Reece, 2004), and the length of Dp is 10 nm in the simulation.

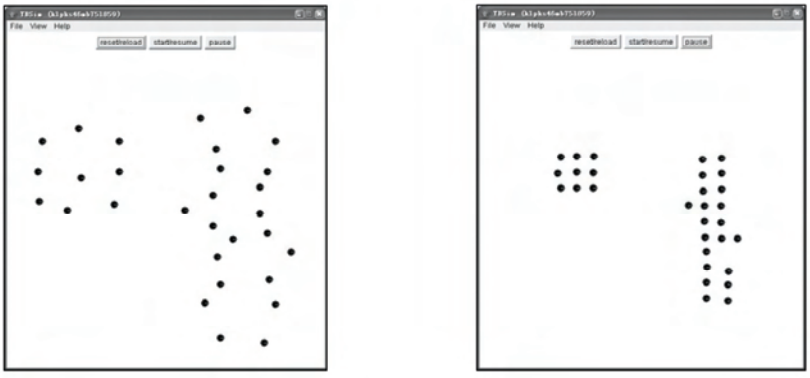
In the first simulation, 30 protein robots are randomly distributed in the mixture of protein solution/dispersion and lipid molecules along with the precipitant/salt which create the medium space. The original distance between the proteins is assumed to be between 15 to 20 nm which is small enough for the van der Waals force to become effective.

Using equation (1), one has to select the parameter C which is related to the potential between two molecules. From Fig. 6, one can see that C increases nonlinearly as the two molecules gets closer until the two reaches the length of bond. From the curve of Fig. 6, we select the following nonlinear equation for *C*:

$$C = N[(X_{in} - X_i) - D_p]^{-3}. \tag{9}$$

Let N = 0.05 nm-3/sec. The proteins will move with a speed of 0.0005 nm/sec when they are 20 nm away from the nearest neighbor until 0.05 nm/sec when only 11 nm apart. The proteins will not move any longer when the bond is formed which is 10 nm. With the above parameters, the two proteins which are 20 nm apart take approximately 5,000 seconds to become just 15 nm. It will take much shorter time to travel the final 5 nm for reaching the length of bond.

In the simulation we assume each time step to be 150 seconds. The 30 protein robots form two pieces of crystal in 45 steps, i.e., 6,750 seconds. That time scale is in line with the actual crystallization experiments (from hours to days). Fig. 9 shows both original and final locations of the protein robots. We can clearly see two groups of protein robots forming in the process. Once the two are formed individually, no sufficiently large force exists to drive them together, and the crystallization process stops to grow even larger crystals.



a) b)

Figure 9. Simulation of team forming of protein robots: a. initial distribution, and b. formation of crystals

To see how the theoretical analysis and simulation results compare with the real crystallization process, we have examined the experimental results of the membrane protein crystallization process. The experiment was performed in the bio-chemistry laboratory at The Ohio State University which is specilized in the crystallization of membrance proteins. The experiment result is shown in Fig. 10 (Muthusubramaniam, 2004). One can see that pieces of the protein crystal (darker pink color) formed at the end, and some are larger than others. The result is in agreement with the theoretical analysis and the simulation results. That is, multiple pieces of protein crystal formed from a large number of protein robots when the latter takes a simple motion strategy as described in this chapter.
We have also conducted a simulation for the second set of rules as mentioned in Section 2. That is, additional constraints are imposed on the simple and local motions of the protein robots. That constraints include the direction of the motion and the field of view in seeking the nearest neighbors. For the given direction of motion, and different angles of views, both shown in Fig. 11, 30 robots form different pieces of lines after 3,200 seconds as shown in Fig. 11. This result shows that with only a slight increase of the level of intelligence, a team of protein robots can form globally different shape of crystal.

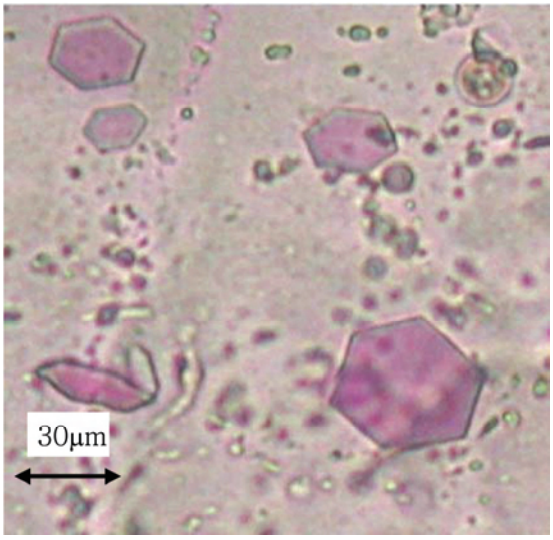


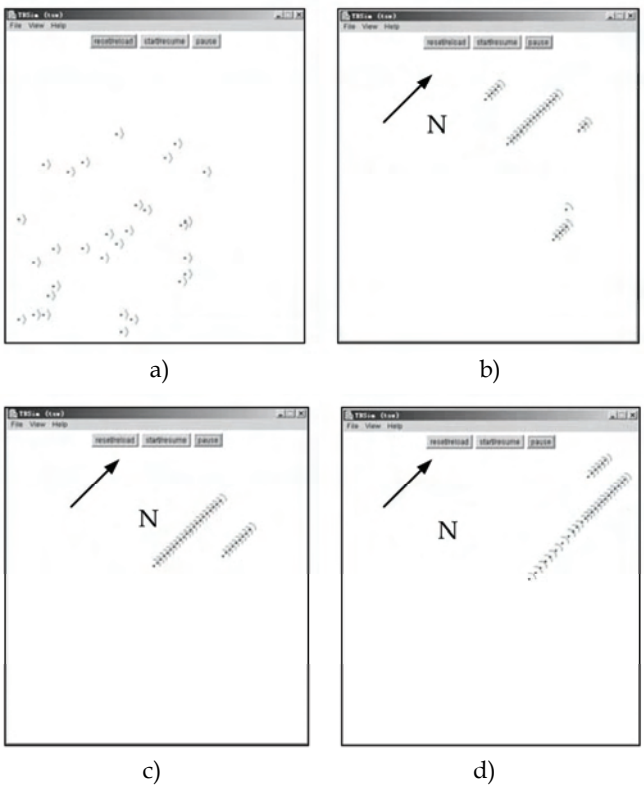
Figure 10. Crystals of a membrane protein, bacteriorhodopsin, obtained after 3 days using 25 nL of cubic phase (100 ng of protein) in 1 μL of precipitant solution at 20 °C (Muthusubramaniam, 2004)

5. Conclusions

In this chapter, we have developed a new theory to explain the growth of protein crystal. The theory is based on the strategy of robot path planning for team forming by multiple mobile robots. Simplicity and locality are two principles in planning the path of the protein robots. The two principles are necessary since we cannot assume proteins to have a high level of intelligence. Based on the two principles, we further have proposed a set of rules which govern the motion trajectory of the protein robot in the process of crystal forming.

Mathematical analysis proves that using the proposed motion strategy, proteins are attracted to its nearest neighbor until the distance of bond is reached. Physical analysis based on the van der Waals forces shows that the proposed motion is physically possible as well as stable. In addition, uniform orientation of the proteins which is a must condition for well-ordered structure can by realized by the electric field, generated by bio-chemical medium involved in the crystallization process, and the van der Waals forces collectively. The structure of the protein is also a factor of the uniform orientation.

Simulation results verify that the proposed motions of the protein robots can form crystal with a globally uniform structure. The results are in agreement with the experiments which we conducted in our laboratory in which pieces of crystals are formed from individual proteins after a period of incubation in an environment ideal for the crystal growth. We further propose that the shape of the robot team or the crystal can be controlled to certain extent when some constraints are imposed to the local motions of the proteins. The latter however requires a higher level of intelligence which may not be easily achieved by alternating the medium space. Our future study should explore more the relationship between the local motions of individual components and the global shape of the organisms which the components build.



a) b)

c) d)

Figure 11. Simulation of team forming of protein robots with constraints: a. initial distribution, b. angle of view=60 degrees, c. angle of view=120 degrees, and d. angle of view=180 degrees

This work represents a new attempt to study the growth of protein crystal using the robotic team forming technique. The key of the technique is the two steps involved, i.e., planning the motion and proving the planned motion governed by a control law is stable. We hope this method can be extended to other biological agents such as cells. Those agents move autonomously as a mobile robot, but the motion trajectories could be much more complicated than the protein robots, and the force of control may not be as simple as a single van der Waals force between units only. The two-step approach, however, is still a valid tool to analyze the trajectories and cause of the motions.

6. References

Caffrey, M. (1986). X-ray diffraction as a technique for studying the mesomorphic phase properties of lipids, In: *Membranes, Metabolism and Dry Organisms*, A. Leopold (Ed.), pp. 350-357, Cornell University Press, Itacha, NY.

Caffrey, M. (2000). A lipid's eye view of membrane protein crystallization in mesophases, *Curr. Opin. Struct. Biol.*, Vol. 10, pp. 486-497.

Campbell, N. and Reece, J. (2004). Biology, Seventh Edition, Addison Wesley Longman, Inc., New York, NY.

Bergfors, T. (Ed.) (1999). Protein Crystallization, Techniques, Strategies, and Tips, *International University Line*, La Jolla, CA.

Ceng, X., Li, S. and Xia, D. (2005). Study of self-organization model of multiple mobile robots, *Int. J. of Advanced Robotic Systems*, Vol. 2, pp. 235-238.

Cheng, A., Hummel, B., Qiu, H., and Caffrey, M. (1998). A Simple mechanical mixer for small viscous lipid-containing samples, *Chem. Phys. Lipids*, Vol. 95, pp. 11-21.

Cherezov, V., Peddi, A., Muthusubramaniam, L., Zheng, Y., and Caffrey, M. (2004). A robotic system for crystallizing membrane and soluble proteins in lipidic meso phases, *Acta. Cryst. D*, D60, pp. 1795-1807.

Chio, T.-S. and Tarn, T. (2003). Rules and control strategies of multi-robot team moving in hierarchical formation, *Proceedings IEEE Int. Conf. on Robotics and Automation*, Taipei, Taiwan, Sep. 2003, pp. 2701-2706.

Das, S., Hu, Y., Lee, C., and Lu, Y.-H. (2005). An efficient group communication protocol for mobile robots, *Proceedings IEEE Int. Conf. on Robotics and Automation*, Barcelona, Spain, April 2005, pp. 87-92.

D'Azzo, J., Houpis, C., and Sheldon, S. (2003). Linear Control System Analysis and Design with MatLab, Marcel Dekker, Inc., New York, NY.

Desai, J., Ostrowski, J., and Kumar, V. (1998). Controlling formations of multiple mobile robots, *Proceedings IEEE Int. Conf. on Robotics and Automation*, Leuven, Belgium, May 1998, pp. 2864-2869.

Fregene, K., Kennedy, D., and Wang, D. (2003). Multi-vehicle pursuit-evasion: an agent-based framework, *Proceedings IEEE Int. Conf. on Robotics and Automation*, Taipei, Taiwan, Sep., 2003, pp. 2707-2713.

Gazi, V. and Passino, K. (2004). Stability analysis of social foraging swarms," *IEEE Trans. on Systems, Man, and Cybernetics-Part B: Cybernetics*, Vol. 34, No. 1, pp. 539-557.

Landau, E. and Rosenbusch, J. (1996). Lipidic cubic phases: a novel concept for the crystallization of membrane proteins, *Proceedings National Academy of Science*, USA, Vol. 93, pp. 14532-14535.

Leonard, N. and Fiorelli, E. (2001). Virtual leaders, artificial potentials and coordinated control of groups, *Proceedings IEEE Conf. on Decision and Control*, Orlando, FL, Dec. 2001, pp. 2968-2973.

Li, X., Xiao, J., and Cai, Z. (2005). Stable flocking of swarms using local information, *Proceedings IEEE Int. Conf. on Systems, Man and Cybernetics*, Oct. 2005, pp. 3921-3926.

Liang, J. and Suganthan, P. (2005). Dynamic multi-swarm particle swarm optimizer, *Proceedings IEEE Swarm Intelligence Symposium*, Pasadena, CA, June 2005, pp. 124-129.

Liu, Y. and Passino, K. (2004). Stable social foraging swarming in a noisy environment, *IEEE Trans. on Automatic Control*, Vol. 49, pp. 30-44.

Luzzati, V. (1968). X-ray diffraction studies of lipid-water systems, *Biological Membranes, Physical Facts and Functions*, *Vol. 1*, D. Chapman, D. (Ed.), pp. 71-123, Academic Press, New York, NY.

Luzzati, V. (1997). Biological significance of lipid polymorphism-the cubic phases, *Curr. Opin. Struct. Biol.*, Vol. 5, pp. 661-668.

Meyer, A. and Nagpal, R. (2002). Course Notes 2: Induction, from *Fall'02: Mathematics for Computer Science*, MIT, Boston, MA.

Muthusubramaniam, L., Peddi, A., Zheng, Y., Cherezov, V., and Caffrey, M. (2004). Automating crystallization of membrane proteins by robot with soft coordinate measuring, *Proceedings IEEE Int. Conference on Robotics and Automation*, New Orleans, LA, Apr. 2004, pp. 1450-1455.

Parker L., Schnider, F., and Schultz, A. (Ed.) (2005). Multi-Robot Systems: from Swarms to Intelligent Automata, Springer, New York, NY.

Pollock, T. (1995). Properties of Matter, McGraw-Hill, Inc., New York, NY.

Secrest, B. and Lamont, G. (2003). Visualizing particle swarm optimization – Gaussian particle swarm optimization, *Proceedings IEEE Swarm Intelligence Symposium*, Indianapolis, IN, April 2003, pp. 198-204.

Sweeney, J., Li, H., Grupen, R., and Ramamritham, K. (2003). Scalability and schedulability in large, coordinated, distributed robot systems, *Proceedings IEEE Int. Conf. on Robotics and Automation*, Taipei, Taiwan, Sep. 2003, pp. 4074-4079.

Talanquer, V. and Oxtoby, D. (1998). Crystal nucleation in the presence of a metastable critical point, *J. of Chemical Physics*, Vol. 109, pp. 223-227.

Tanford, C. and Reynolds, J. (2004). Nature's Robots: a History of Proteins, Oxford University Press, Oxford, UK.

Taylor, C. and Lipson, H. (1965). Optical Transforms, Their Preparation and Application to X-Ray Diffraction Problems, Cornell University Press, Itacha, NY.

ten Wolde, P. and Frenkel, D. (1997). Enhancement of protein crystal nucleation by critical density fluctuations, *Science*, Vol. 277, pp. 1975-1978.