The Journal of Supercomputing
https://doi.org/10.1007/s11227-023-05192-5

The music demixing machine: toward real-time remixing of classical music

Pablo Cabañas-Molero[1] · Antonio J. Muñoz-Montoro[2] · Pedro Vera-Candeas[1] · José Ranilla[2]

Accepted: 12 March 2023

Abstract

Classical music, unlike popular music, is usually recorded live with close microphone techniques. For this reason, isolated tracks are not available to create the final mixture/stream, and so the mixing process requires greater effort. Source separation methods are a potential solution to this problem. However, current algorithms are not fast enough to yield real-time separation in professional setups with dozens of microphones and sources. In this paper, we propose a fast approach consisting of a panning-based multichannel non-negative matrix factorization model to separate classical music. We tested the system on real professional recordings, where we were able to reach real-time with very low latency and promising quality.

Keywords Music source separation · Real time · Multichannel NMF · Classical music · High performance computing

These authors contributed equally to this work.

✉ Pablo Cabañas-Molero
pcabanas@ujaen.es

Antonio J. Muñoz-Montoro
munozantonio@uniovi.es

Pedro Vera-Candeas
pvera@ujaen.es

José Ranilla
ranilla@uniovi.es

[1] Department of Telecommunication Engineering, University of Jaen, Campus Cientifico-Tecnologico de Linares, 23700 Linares, Jaen, Spain

[2] Department of Computer Science, University of Oviedo, Campus de Gijón, 33204 Gijón, Asturias, Spain

Published online: 06 April 2023

1 Introduction

Music source separation (SS) is the process of segregating a music mixture into its constituting individual instruments. This is a task of great interest for multiple musical applications, including music remixing, equalization, professional production of records/streams, music education and recreational purposes (e.g., karaoke).

In the last years, thanks to the emergence of deep neural networks (DNN), many competent systems have been proposed to perform SS on popular music. Typically, convolutional neural networks (CNN) or long short-term memory (LSTM) networks are trained on large datasets containing mixture-source pairs. Since popular music is usually created by recording each instrument separately (before the audio engineer mixes all stems to produce the final mixture), reasonably large datasets are available for training, specially thanks to artists sharing their stems. Among the most popular deep learning models, we can cite Open-Unmix [1], Spleeter [2] or Demucs [3].

Classical music, unlike popular music, is usually recorded with all musicians performing at the same time in the same acoustic space. For this reason, datasets with isolated sources are scarce in classical music, which is a great impediment for training data-driven SS models. In a typical recording session, each instrument is recorded with one or more close microphones, whereas the whole orchestra/ensemble is captured with two main microphones (stereo pair) [4]. Additional channels can be used to pick up ambient sounds. The final recording (or stream) is created by boosting the stereo pair with the close-mic and ambient signals, selecting appropriate gain levels and compensating the inter-channel time differences (ITD). Usually, before the concert, each musician is asked to play alone for a few seconds to allow the audio engineer to fine-tune some of these parameters. However, because the instruments are not completely isolated in each close channel during the concert, the mixing process still requires great effort and expertise, with numerous tricks and adjustments to mask microphone leakage (e.g., artificial reverberation) [5]. An SS algorithm able to yield high-quality isolated stems for mixing could facilitate this labor significantly. Here, real-time is a desirable requirement, in particular when the concert is to be transmitted via streaming.

The lack of training material restricts SS in classical music to heuristic solutions, such as spectrogram decomposition with non-negative matrix factorization (NMF) or similar [6–8]. Even though these methods do not achieve better quality than machine learning, they can deliver reasonably good results when prior information is known, such as instrumentation, score of the piece, or inter-channel level/phase differences for each instrument. Recently, systems based on multichannel NMF (MNMF) have been proposed with promising results [9–11]. However, current approaches are too complex for real-time delivery in a realistic recording setup, which usually comprises dozens of instruments and channels. Besides, some methods rely on information that may not be available during the recording, such as microphone positions or the piece score (as in [9]).

In this paper, we propose a MNMF system able to reach real-time SS in classical music with a high number of channels and instruments. The MNMF model is

similar to the version described in [12], based on a panning matrix that encodes inter-channel level differences (ILD) for each instrument, but with pre-trained instruments bases. This model was chosen for efficiency, simplicity and flexibility reasons. Even though it does not represent accurately how the sources are mixed in the air, it exploits effectively multichannel information. The panning matrix can be learnt in a preliminary stage, asking each performer to play alone for a few seconds before the concert, which is a common practice. The method does not need any other additional prior information. During the separation, only the instrument time-varying amplitudes have to be estimated. The system is able to produce hundreds of stems in real time with reasonable quality. Our approach provides the basis to create a real-time music demixing machine for classical music able to work in realistic conditions.

We organize this document as follows. In Sect. 2, we describe the implemented SS system. In Sect. 3, we test the algorithm on professional orchestra recordings and synthetic mixtures and provide objective and subjective SS measures. A detailed experimentation is conducted to analyze the execution time and efficiency of the algorithm. Section 4 concludes the paper.

2 Proposed separation method

2.1 Problem statement and assumptions

The problem addressed in this work is to separate each source instrument from a live music performance recorded with N microphones. The input of the system is an N-channel signal x_n, with $n \in [1, N]$, and the outputs are the $j \in [1, J]$ source instrument signals at each microphone, which give us a total of $J \cdot N$ output waveforms $\hat{y}_{jn}$. In practice, it is not necessary to get so many outputs, because most channels correspond to close microphones where only one instrument is desired.

In the frequency domain, we denote the input signals at time frame t as $\mathbf{x}_{nt} = [x_{1,nt}, \ldots, x_{F,nt}]^\intercal \in \mathbb{R}_+^F$ and the original sources as $\mathbf{y}_{jt} = [y_{1,jt}, \ldots, y_{F,jt}]^\intercal \in \mathbb{R}_+^F$, where F is the number of bins. In this work, we use the same frequency representation as in [9], where the signal is windowed with Hanning windows and the dimensionality of the magnitude STFT is reduced to a musically meaningful resolution [13].

We assume that the instruments are mixed in each microphone following an instantaneous mixture model,

$$\mathbf{x}_{\mathrm{nt}} = \sum_{j=1}^{J} m_{\mathrm{nj}} \mathbf{y}_{\mathrm{jt}}, \tag{1}$$

where $m_{\mathrm{nj}} \in \mathbb{R}_+$ is the gain contribution of instrument j to channel n (panning coefficient). Clearly, this model is a simplification of the way signals are actually mixed. The actual mixing is convolutive in nature and depends on the source-to-microphone paths and room acoustics. However, in close mixing scenarios, this model is useful enough to exploit effectively spatial information, because in that case, ILDs are

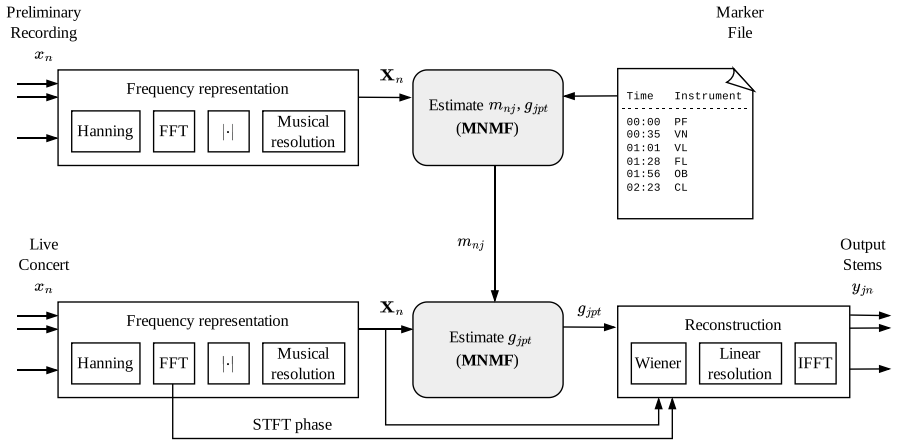


Fig. 1 Block diagram of the proposed separation system

significant. Coefficients m_{nj} are also supposed to be time-invariant (i.e., sources and instruments do not change placement).

In addition, we assume that each instrument source signal can be modelled as a linear combination of P spectral patterns,

$$\mathbf{y}_{\mathrm{jt}} = \sum_{p=1}^{P} g_{\mathrm{jpt}} \mathbf{b}_{\mathrm{jp}}, \tag{2}$$

where $\mathbf{b}_{\mathrm{jp}} \in \mathbb{R}_{+}^{F}$ is the p-th spectral pattern of instrument j, and $g_{\mathrm{jpt}} \in \mathbb{R}_{+}$ are their corresponding time-varying gains. In our approach, the instrumentation of the piece is known in advance. The spectral patterns are picked from a dictionary that contains pre-learned spectral patterns for each classical instrument, with each pattern representing the shape of a single pitch/note. Further information about how to generate this dictionary can be found in [13].

2.2 System overview

Our approach operates in two stages. The first one is dedicated to estimate the panning coefficients m_{nj}. To this end, the system works on the preliminary recordings that audio engineers make before the concert. Typically, in these recordings, each performer plays alone for a short time, allowing the engineer to find the best mixing settings (gain, delay, reverb) for each channel in the absence of interferences. The time interval during which each instrument is playing is annotated manually in a text file in the form of time markers (see Fig. 1). Our system uses these annotated recordings to accurately determine the panning coefficients for each instrument, exploiting the fact that each source is free of interference.

In the second stage, dedicated to source separation during the concert, the system estimates the time-varying gains g_{jpt}. The panning coefficients m_{nj} (estimated in the previous stage) and the instrumental patterns $\mathbf{b}_{\text{jp}}$ are kept fixed. Unlike the previous stage, this one has to work in real time, and therefore, the inference of g_{jpt} must be heavily optimized. Here, the signal is processed in a block-wise fashion in order to keep latency to a minimum. A block diagram of the proposed approach is shown in Fig. 1.

2.3 MNMF instantaneous model

Let $\mathbf{X}_n = [\mathbf{x}_{n1}, \ldots, \mathbf{x}_{nT}] \in \mathbb{R}_+^{F\times T}$ be the frequency representation of a small block of input signal at channel n, where T is the block length in frames. Following assumptions (1) and (2), our MNMF decomposition model $\hat{\mathbf{X}}_n$ can be formulated as

$$\mathbf{X}_n \approx \hat{\mathbf{X}}_n = \sum_{j=1}^{J} m_{\text{nj}} \mathbf{B}_j \mathbf{G}_j, \tag{3}$$

where $\mathbf{B}_j = [\mathbf{b}_{j1}, \ldots, \mathbf{b}_{jP}] \in \mathbb{R}_+^{F\times P}$ is the matrix that holds the spectral patterns for instrument j, and $\mathbf{G}_j \in \mathbb{R}_+^{P\times T}$ holds their corresponding time-varying gains g_{jpt} across the block. This model is just a panning-based extension of NMF, similar to [12], but with pre-computed and fixed bases $\mathbf{B}_j$. The unknowns of the model are the panning coefficients m_{nj} and the gains $\mathbf{G}_j$.

The model parameters are found by minimizing the β-divergence between the observed input $\mathbf{X}_n$ and its modeled form $\hat{\mathbf{X}}_n$. In order to make the algorithm independent from the signal amplitude, the input matrix $\mathbf{X}_n$ is first normalized to have β-norm equal to 1 as follows:

$$\mathbf{X}_n \leftarrow \frac{\mathbf{X}_n}{\left(\sum_{fnt} \left(x_{fnt}\right)^{\beta}\right)^{1/\beta}}. \tag{4}$$

The β-divergence is minimized using the well-known multiplicative gradient approach [14]. First, the variables $\mathbf{G}_j$ and m_{nj} are initialized with non-negative values. Then, they are updated iteratively using multiplicative update rules in which each element is multiplied by a positive gradient. These rules guarantee that the β -divergence is reduced at each iteration, while preserving non-negativity of the variables. For our model in (3), the update rules are

$$\mathbf{G}_j \leftarrow \mathbf{G}_j \odot \frac{\sum_n m_{\text{nj}} \cdot \mathbf{B}_j^{\intercal} \cdot \left(\hat{\mathbf{X}}_n^{\beta-2} \odot \mathbf{X}_n\right)}{\sum_n m_{\text{nj}} \cdot \mathbf{B}_j^{\intercal} \cdot \hat{\mathbf{X}}_n^{\beta-1}}, \tag{5}$$

$$m_{\mathrm{nj}} \leftarrow m_{\mathrm{nj}} \cdot \frac{\sum_{\mathrm{ft}} \left[\mathbf{B}_j \mathbf{G}_j\right]_{\mathrm{ft}} \cdot \left[\hat{\mathbf{X}}_n^{\beta-2} \odot \mathbf{X}_n\right]_{\mathrm{ft}}}{\sum_{\mathrm{ft}} \left[\mathbf{B}_j \mathbf{G}_j\right]_{\mathrm{ft}} \cdot \left[\hat{\mathbf{X}}_n^{\beta-1}\right]_{\mathrm{ft}}}, \tag{6}$$

where divisions, exponentials and multiplications denoted by $\odot$ are element-wise operations, and $[\cdot]_{\mathrm{ft}}$ means matrix indexing. We run the algorithm with a fixed number of iterations, and set $\beta = 1.5$, which has been found to perform well with music signals [9].

In the first stage of the system, $\mathbf{X}_n$ represents the preliminary test recording, which is a few minutes long at most. Since we know the interval where each instrument is playing, $\mathbf{G}_j$ is initialized to 1 in the frames where instrument j is active, and 0 in the rest. Coefficients m_{nj} are initialized to 1. Both update rules (5 and 6) are applied in this stage, and the resulting m_{nj} values are stored for later use. Real time is not a requirement in this stage.

In the second stage, $\mathbf{X}_n$ represents a block of the live audio signal. To ensure a low latency for streaming applications, the block size is set to a small number of frames. Coefficients m_{nj} are initialized to the values obtained in the previous stage and kept fixed, so that only rule (5) is applied to estimate $\mathbf{G}_j$, whose elements are initialized to 1. Implementation of equation (5) is then critical, because in real conditions, recordings can be made with dozens of instruments and channels. Thereby, efficient techniques including parallel and high performance computing will be used. The estimated gains $\mathbf{G}_j$ are used to reconstruct the sources in each channel for the current block.

2.4 Source reconstruction

The J instrument signals in each channel are reconstructed from the parameters estimated in the MNMF decomposition. To do this, we make use of the well-known Wiener filtering strategy. Observe that according to our model in (3), the j-th spectrogram in channel n can be constructed as

$$\hat{\mathbf{S}}_{\mathrm{nj}} = m_{\mathrm{nj}} \mathbf{B}_j \mathbf{G}_j, \tag{7}$$

with $\hat{\mathbf{S}}_{\mathrm{nj}} \in \mathbb{R}_+^{F \times T}$. A single-channel Wiener mask for a certain source j represents the relative energy contribution of source j with respect to the total energy of the mixture. In other words, the Wiener mask $\mathbf{V}_{\mathrm{nj}} \in \mathbb{R}_+^{F \times T}$ for source j and channel n can be computed as

$$\mathbf{V}_{\mathrm{nj}} = \frac{|\hat{\mathbf{S}}_{\mathrm{nj}}|^2}{\sum_j |\hat{\mathbf{S}}_{\mathrm{nj}}|^2}. \tag{8}$$

Then, we can extract the spectrogram of source j from channel n by applying the filter $\mathbf{V}_{\mathrm{nj}}$ over the input mixture:

$$\hat{\mathbf{Y}}_{\mathrm{nj}} = \sqrt{\mathbf{V}_{\mathrm{nj}}} \odot \mathbf{X}_n. \tag{9}$$

Before transforming the spectrogram $\hat{\mathbf{Y}}_{\mathrm{nj}}$ to the time domain, it is necessary to perform two steps. First, since $\hat{\mathbf{Y}}_{\mathrm{nj}}$ is expressed in musical resolution, it must be converted to lineal frequency resolution. This is done by replicating the value of each band f to all its corresponding linear frequency bins. Second, since $\hat{\mathbf{Y}}_{\mathrm{nj}}$ only contains magnitude information, the phase must be provided to obtain a complex-valued spectrogram. Here, for all sources, we use the phase of the STFT of the input signal x_n. Finally, the inverse STFT is employed to yield the output stems $\hat{y}_{\mathrm{jn}}$. For the reconstruction stage, we use the same software developed in [9].

The separation system proposed in this study is summarized by Algorithms 1 and 2. Algorithm 1 focuses on the first stage of the system, which involves the estimation of panning coefficients m_{nj}. This estimation is performed using preliminary recordings without the constraint of real-time processing. By contrast, Algorithm 2 involves the estimation of the time-varying gains g_{jpt}. This stage must be performed in real time, demanding the use of parallel and high-performance computing (HPC) techniques in its design. Specifically, parallel OpenMP directives were used to buffer input audio frames (see line 4 of Algorithm 2). Next, frames within the same audio block $\mathbf{X}_n$ were processed in parallel using OpenMP directives (see line 5 of Algorithm 2). Later, the norm of $\mathbf{X}_n$ was calculated in parallel using reduction clauses and directives of OpenMP. The matrix is then normalized using Level 3 BLAS calls (see line 6 of Algorithm 2). Finally, the operations involved in the NMF update (see line 9 of Algorithm 2) were parallelized using BLAS calls for matrix–matrix and vector–vector products and OpenMP directives for non-BLAS operations. Notably, for three-dimensional matrix products, several versions were implemented using the *batched* variants available in Intel oneAPI.

Algorithm 1 Proposed MNMF separation algorithm (panning estimation)

1: Load recording x_n and its marker file
2: Compute $\mathbf{X}_n$ (Hanning window, magnitude FFT, musical resolution) and save original FFT phase
3: Normalize $\mathbf{X}_n$ by (4)
4: Load instrument bases $\mathbf{B}_j$
5: Initialize gains $\mathbf{G}_j$ according to marker file
6: Initialize panning coefficients m_{nj} to 1
7: **for** $i = 1$ to # of iter **do**
8: Update $\mathbf{G}_j$ by (5)
9: Update m_{nj} by (6)
10: **end for**
11: Save m_{nj}

Algorithm 2 Proposed MNMF separation algorithm (separation stage)

Load instrument bases $\mathbf{B}_j$
Load panning coefficients m_{nj}
while audio keeps coming **do**
    Buffer T frames of audio x_n
    Compute $\mathbf{X}_n$ (Hanning window, magnitude FFT, musical resolution) and save original FFT phase
    Normalize $\mathbf{X}_n$ by (4)
    Initialize gains $\mathbf{G}_j$ to 1
    for $i = 1$ to # of iter **do**
        Update $\mathbf{G}_j$ by (5)
    end for
    Compute $\mathbf{B}_j\mathbf{G}_j$
    for $n = 1$ to N **do**
        for $j = 1$ to J **do**
            Compute $\hat{\mathbf{Y}}_{nj}$ by (7), (8) and (9)
            Change to linear resolution and add phase
            Inverse FFT
            Output $\hat{y}_{jn}$
        end for
    end for
end while

3 Experiments

3.1 Experimental setup

We evaluated the proposed method using two real professional recordings created by a record label. These recordings contain performances of two classical pieces composed by Brahms and Webern, played with piano and orchestra in a concert hall. Each instrument was recorded with one or two close microphones, and the whole scene was captured with a stereo pair and several ambient sensors. Both recordings contain a preliminary segment where each musician plays alone for a few seconds. These segments were annotated by the audio engineer in a marker file. We used these annotated sections to estimate our panning matrix.

In addition, to give objective separation measures, we employed a subset of the University of Rochester Multimodal Music Performance (URMP) database [15], which provides separated stems. The chosen subset is composed of four pieces of chamber music (two quintets and two quarters) played by wind and stringed instruments. To generate multichannel mixtures, we simulated the spatial position of the sources and microphones using a room simulator based on the image method [16]. The simulated room is a rectangular prism of dimensions $22 \times 16 \times 5$ m^3 and reverberation time RT_{60} equal to 1 s. The room layout is illustrated in Fig. 2, where the

Table 1 Dataset used to evaluate our music separation system

Composer	Piece name	Duration	Channels	Instr	Room
Allegri	Miserere mei deus	00:40	5	5	Simulated
Mozart	String quintet K515	03:45	5	5	Simulated
Purcell	Rondeau from abdelazer	02:08	4	4	Simulated
Dvořák	Slavonic dance	01:22	4	4	Simulated
Brahms	Piano concerto No. 2	26:14	30	18	Real
Webern	Concerto for 9 instruments	11:40	20	9	Real

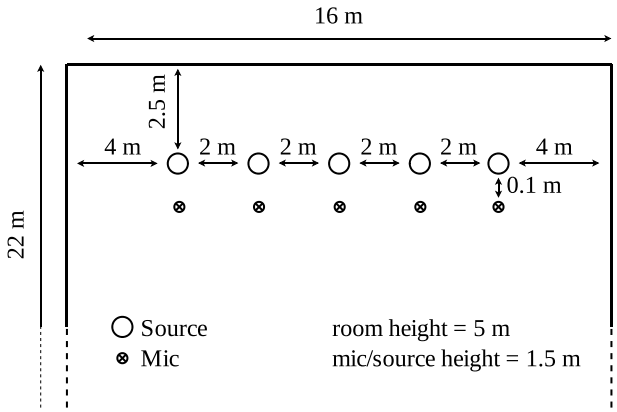


Fig. 2 Simulated room and microphone/source positions (quintet)

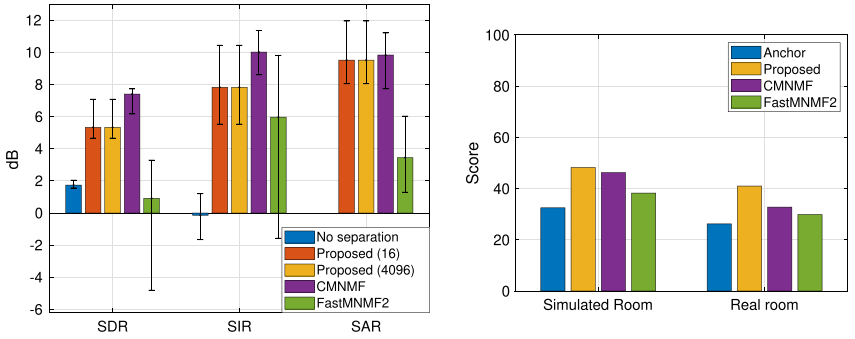
close microphones are omnidirectional. We also generated a preliminary recording for each piece to learn the panning matrix.

Information about our dataset is provided in Table 1. All mixtures are sampled at 44 kHz and were analyzed using a frame size of 5700 samples and a frame shift of 570 samples (13 ms). The number of iterations of the MNMF decomposition was set to 50.

In our experimentation, we used two different systems as a testbed. The first system was an Intel server with two Intel Xeon® Silver 4110 processors, each with 8 cores. The processors operate at a clock speed of 2.10 GHz, and HyperThreading and Turbo Boost were disabled in the experimentation. The second system was the NVIDIA Jetson AGX Xavier development kit, which is a system-on-chip with an 8-core ARM v8.2 64-bit CPU. The Xavier platform was used to simulate the performance of a range of mobile devices, such as smartphones, laptops, and tablets, under controlled conditions.

The Xavier system ran Ubuntu 18.04.6 LTS and uses the OpenBLAS[1] library (release 0.3.20, February 2022) and the FFTW[2] library (release 3.3.10, September

[1] https://www.openblas.net
[2] http://www.fftw.org



(a) Objective measures in simulated room.

(b) Subjective evaluation (MUSHRA) in simulated and real rooms.

Fig. 3 Separation results for sources extracted from their closest microphone

2021). The Intel server ran CentOS Linux 7.9 and the Intel oneAPI Toolkit (release 2022.2). Although the Intel oneAPI Toolkit provides the FFTW interface, the external FFTW library (release 3.3.10, September 2021) was used on the Intel server to accurately compare the performance of the two systems and draw meaningful conclusions from the experimentation.

3.2 Separation results

For the URMP subset, the separation quality was objectively measured using the BSS_eval toolbox [17], which provides three metrics: signal-to-distortion ratio (SDR), signal-to-interference ratio (SIR) and signal-to-artifacts ratio (SAR). For each source, we only considered the separated stem in its corresponding close microphone. We tested two different block sizes: $T = 16$ and $T = 4096$ frames.

We compared our system with two state-of-the-art MNMF-based methods. The first one is the complex-valued multichannel extension (CMNMF) proposed in [18]. In this case, the complex spatial matrix for each instrument was initialized using the preliminary recording, in a similar fashion to our system. Euclidean divergence was used, and all model parameters were updated during the separation until convergence. The second one is FastMNMF2 proposed in [19], using the code provided by the authors. In this case, the method was not initialized with any prior information, so it ran completely blind. Finally, the baseline result without conducting any separation was computed (i.e., we compared the ground-truth stem with the instrument spot signal divided by the number of sources).

Figure 3a illustrates the median values of SDR, SIR and SAR obtained in our experiments, together with 25-th and 75-th percentiles. As shown, our system achieves SDR = 5.34 dB, SIR = 7.83 dB and SAR = 9.52 dB, which are fairly superior to the baseline metrics. Our evaluation shows that the block size does not impact the separation quality. This was expected, as each g_{jpt} is calculated from the same input variables for any T, which only affects the normalization step in (4). In this test, CMNMF was the best method, achieving approximately 2 dB more than our

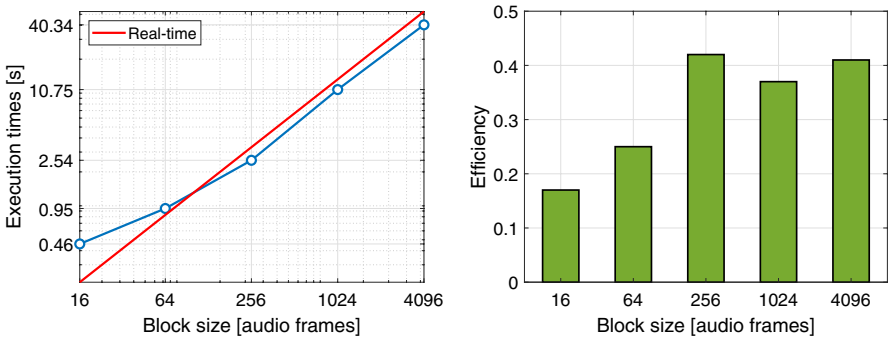
system in terms of SDR and SIR. FastMNMF2 performed poorly in terms of distortion and artifacts, but still managed to achieve a competitive level of SIR (6 dB). We recall FastMNMF2 is operating blindly.

For both real (*Brahms*, *Webern*) and URMP mixtures, we conducted a subjective evaluation using the MUlti Stimulus test with Hidden Reference and Anchor (MUSHRA), which is a well-established approach for evaluating audio quality as specified in ITU-R Recommendation BS.1534-1 [20]. In the MUSHRA test, the participants are given the signals being tested along with a reference signal (typically the clean source) and asked to rate the different signals on a quality scale ranging from 0 to 100. The mixture without separation is also included among the tested signals as an anchor. The algorithm corresponding to each signal is hidden. Since we do not have access to a clean reference for the real recordings, we presented the listeners with the name of the instrument and asked them to provide a low score if the instrument sounded distorted, artifacted or interfered with other sources, and vice versa. Twenty-three listeners ranging in age from their 20 s to their 40 s participated in the test. Figure 3b shows the average subjective measures for all compared methods and the anchor (no separation). As can be observed, listeners preferred our system in both real and simulated recordings. In our opinion, the reason is that phase-based methods such as CMNMF and FastMNMF2 usually perform worse in higher frequencies due to the phase ambiguity problem. Objective measures are biased to very low frequencies just because audio signals usually present higher energy in that range. On the contrary, psycho-acoustic subjective measures are more balanced in all frequency bands, since the auditory system is more sensitive to higher frequencies. In real recordings, CMNMF performed significantly worse, perhaps due to its limitations for modeling real-world mixing. Excerpts used in the test, as well as other separated stems, can be found online.[3]

3.3 Computational results

Concerning the computational results, a specific experimentation was designed and conducted to evaluate the performance of our system. These results cover the entire system except the reconstruction stage described in 2.4. Figure 4 shows the computational results obtained by the Intel server as a function of the audio block size. As can be observed, the block size to process ranges from $T = 16$ to $T = 4096$ audio frames. Figure 4a illustrates the measured execution times, with a red line indicating the limit for real-time processing. If the runtime for a given block size falls above this line, the system will not be able to process the audio blocks as they arrive. However, if the runtime falls below the red line, the system will be able to process the entire block before the next one arrives. For instance, with a block size of $T = 16$ frames, corresponding to 0.21 s of audio, the processing time obtained is 0.46 s. As can be seen in the figure, the system starts to behave in real time for block sizes larger than 256 frames. The efficiency of

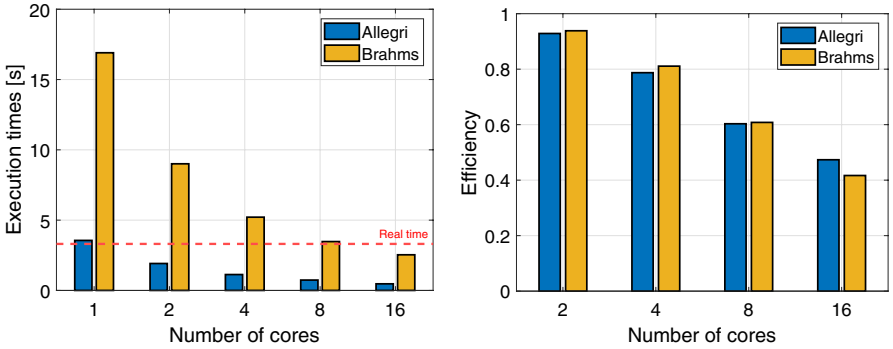
[3] https://gitlab.com/pcabanasmolero/jos-2022



(a) Execution times measured in seconds as a function of audio block size.

(b) Evolution of the efficiency as a function of audio block size.

Fig. 4 Experimental results obtained by the Intel server using all available cores



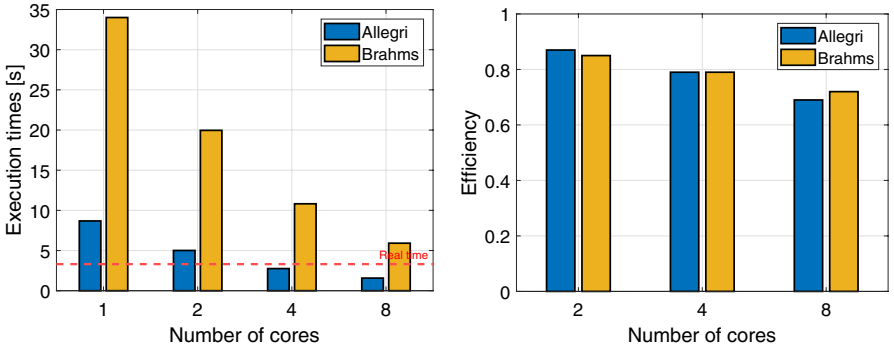
(a) Execution times measured in seconds.

(b) Evolution of the efficiency.

Fig. 5 Experimental results obtained by the Intel server as a function of the number of cores used

the proposed system is depicted in Fig. 4b. As shown, the optimal efficiency is achieved when using a block size of 256 audio frames. This result aligns with expectations, as using a block size of $T = 256$ results in matrices that are more square in shape, providing a better performance for the matrix products and divisions of the algorithm (see(4)-(6)). Therefore, for the sake of brevity, we will use 256 frames as the block size in order to assess the performance of the proposed system for the rest of the experimentation. In this sense, note that processing a 256-frame block corresponds to less than 3 s of latency, which is acceptable for streaming applications.

A study was conducted to investigate the impact of the number of cores on the performance of a source separation system. Two application scenarios were considered: chamber music and orchestral music. As already introduced in Table 1, the main difference between these scenarios is the number of instruments that comprise the musical piece and the number of channels or microphones used



(a) Execution times measured in seconds.

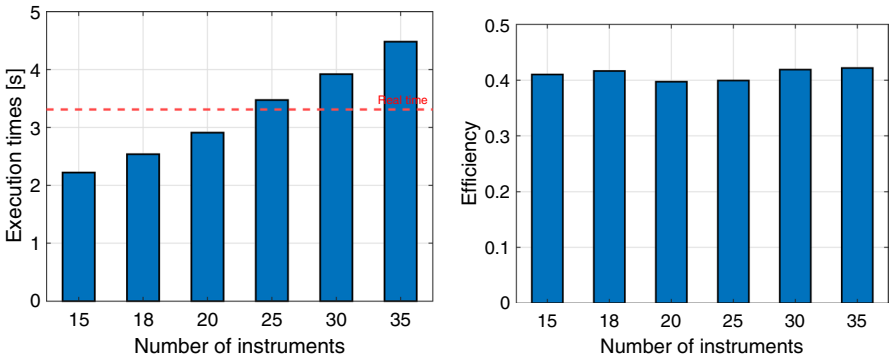
(b) Evolution of the efficiency.

Fig. 6 Experimental results obtained by NVIDIA AGX Xavier as a function of the number of cores used

during the recording process. Obviously, the target task with a large number of instruments and/or channels will involve a higher computational load. Figure 5 shows the results obtained by the Intel server for the *Allegri* and *Brahms* compositions. Thus, the execution times measured are presented in Fig. 5a. As can be observed, the application of parallelism and high performance computing techniques allows for real-time behavior in chamber music settings, characterized by a limited number of instruments and channels. However, in settings with a greater number of instruments and channels, the use of 16 cores is necessary to achieve real-time performance. The efficiency of the system, as shown in Fig. 5b, decreases as the number of cores increases due to the need to divide the workload among a larger number of cores, resulting in less efficient utilization of each individual core.

Figure 6 presents the results of the Xavier platform's performance on the *Allegri* and *Brahms* compositions as a function of the number of cores used. It is noted that real-time processing is only achieved for chamber music when at least 4 cores are utilized (see Fig. 6a). In terms of efficiency, values above 0.7 were obtained in this experiment (see Fig. 6b), indicating that the proposed system scales properly.

Finally, the effect of the number of instruments on the performance of the proposed system was analyzed. Figure 7 summarizes the results obtained by the Intel server when using 16 cores and a block size of 256 frames. The results show that real-time behavior cannot be achieved for musical compositions with more than 25 instruments. This is due to the strong impact of the number of instruments on the operations performed in the MNMF stage. In terms of efficiency, as shown in Fig. 7b, the number of instruments does not significantly affect the performance.



(a) Execution times measured in seconds.

(b) Evolution of the efficiency.

Fig. 7 Experimental results obtained by the Intel server as a function of the number of instruments in the musical composition

4 Conclusions and future work

In this paper, we present a multichannel SS system designed for classical music recordings made in professional environments. Real recordings are usually made using dozens of instruments and close microphones, and real time is often a requirement. Besides, certain information usually needed by previous methods [9] (such as the piece score or source positions) is not always available during the recording. With these requirements in mind, we designed an approach based on panning-based MNMF able to work in real time in typical professional setups without disrupting the workflow of audio engineers. The system can produce separated stems that can be useful to create the final mixture. We tested the system in real orchestra recordings and synthetic chamber mixtures, obtaining promising results in real time with low latency using parallel and high-performance techniques. In a recording with 18 instruments and 30 channels, we achieved real-time separation with less than 3 s of latency in a server with two Intel Xeon® Silver 4110 processors (16 cores).

In future versions, we intend to improve the separation quality by incorporating pyschoacoustic criteria that allow to detect the segments where each instrument is active. Also, we want to estimate certain parameters automatically, such as the ITDs for each instrument, with the goal of creating a complete mixing tool useful to professionals.

Acknowledgements We thank John Anderson from Odratek Records for providing the orchestra recordings used in our experiments.

Author contributions All authors contributed equally to this work.

Funding Funding for open access publishing: Universidad de Jaén/CBUA. This work was supported in part by the Regional Government of Andalucia under the "Proyectos de I+D+i del Plan Andaluz de Investigacion, Desarrollo e Innovacion (PAIDI 2020)" Framework Programme through grant P18-TPJ-4864, and "Ministerio de Ciencia e Innovacion" of Spain under the projects PID2020-119082RB-C21,C22.

Data availability Data used/generated during the current study are available from the corresponding author on reasonable request.

Compliance with ethical standards

Conflicts of interest The authors have no competing interests as defined by Springer, or other interests that might be perceived to influence the results and/or discussion reported in this paper.

Ethical approval Not applicable.

References

1. Stöter FR, Uhlich S, Liutkus A, Mitsufuji Y (2019) Open-unmix - a reference implementation for music source separation. J Open Sour Softw 4(41):1667. https://doi.org/10.21105/joss.01667
2. Hennequin R, Khlif A, Voituret F, Moussallam M (2020) Spleeter: a fast and efficient music source separation tool with pre-trained models. J Open Sour Softw 5(50):2154
3. Défossez A, Usunier N, Bottou L, Bach F Demucs: deep extractor for music sources with extra unlabeled data remixed. Available from: arXiv:1909.01174
4. Huber DM, Runstein R (2013) Modern recording techniques. Routledge;
5. Kokkinis EK, Mourjopoulos J (2010) Unmixing acoustic sources in real reverberant environments for close-microphone applications. J Audio Eng Soc 58(11):907–922
6. Cano E, FitzGerald D, Liutkus A, Plumbley MD, Stöter FR (2019) Musical source separation: an introduction. IEEE Signal Process Mag 36(1):31–40. https://doi.org/10.1109/MSP.2018.2874719
7. Févotte C, Vincent E, Ozerov A (2018) Single-channel audio source separation with NMF: divergences, constraints and algorithms. Makino S, editor. Cham: Springer International Publishing; Available from: https://doi.org/10.1007/978-3-319-73031-8_1
8. Ozerov A, Févotte C, Vincent E An introduction to multichannel NMF for audio source separation. Makino S, editor. Cham: Springer International Publishing; 2018. Available from: https://doi.org/10.1007/978-3-319-73031-8_4
9. Muñoz-Montoro AJ, Suarez-Dou D, Carabias-Orti JJ, Cañadas-Quesada FJ, Ranilla J (2021) Parallel multichannel music source separation system. J Supercomput 77(1):619–637
10. Muñoz-Montoro AJ, Carabias-Orti JJ, Cabañas-Molero P, Cañadas-Quesada FJ, Ruiz-Reyes N (2022) Multichannel blind music source separation using directivity-aware MNMF with harmonicity constraints. IEEE Access. 10:17781–17795
11. Carabias-Orti JJ, Nikunen J, Virtanen T, Vera-Candeas P (2018) Multichannel blind sound source separation using spatial covariance model with level and time differences and nonnegative matrix factorization. IEEE/ACM Trans Audio Speech Language Process 26(9):1512–1527
12. Carabias-Orti JJ, Cobos M, Vera-Candeas P, Rodríguez-Serrano FJ (2013) Nonnegative signal factorization with learnt instrument models for sound source separation in close-microphone recordings. EURASIP J Adv Signal Process 2013(1):1–16
13. Rodriguez-Serrano FJ, Duan Z, Vera-Candeas P, Pardo B, Carabias-Orti JJ (2015) Online score-informed source separation with adaptive instrument models. J New Music Res 44(2):83–96. https://doi.org/10.1080/09298215.2014.989174
14. Févotte C, Idier J (2011) Algorithms for nonnegative matrix factorization with the β-divergence. Neural Comput 23(9):2421–2456

15. Li B, Liu X, Dinesh K, Duan Z, Sharma G (2018) Creating a multitrack classical music performance dataset for multimodal music analysis: challenges, insights, and applications. IEEE Trans Multimedia 21(2):522–535
16. Campbell D, Palomaki K, Brown G (2005) A Matlab simulation of" shoebox" room acoustics for use in research and teaching. Comput Inf Syst 9(3):48
17. Vincent E, Gribonval R, Févotte C (2006) Performance measurement in blind audio source separation. IEEE Trans Audio Speech Lang Process 14(4):1462–1469
18. Sawada H, Kameoka H, Araki S, Ueda N (2013) Multichannel extensions of non-negative matrix factorization with complex-valued data. IEEE Trans Audio Speech Lang Process 21(5):971–982. https://doi.org/10.1109/TASL.2013.2239990
19. Sekiguchi K, Bando Y, Nugraha AA, Yoshii K, Kawahara T (2020) Fast multichannel nonnegative matrix factorization with directivity-aware jointly-diagonalizable spatial covariance matrices for blind source separation. IEEE/ACM Trans on Audio Speech Language Process 28:2610–2625. https://doi.org/10.1109/TASLP.2020.3019181
20. ITU-R BS 1534-3 (2015) Method for the subjective assessment of intermediate quality level of audio systems. International Telecommunication Union;