Noname manuscript No.
(will be inserted by the editor)

Parallel Multichannel Music Source Separation System

A.J. Muñoz-Montoro · D. Suarez-Dou · J.J. Carabias-Orti · F.J. Canadas-Quesada · J. Ranilla

Received: date / Accepted: date

Abstract This paper presents a parallel low latency multichannel source separation system designed to recover the original signals of the instruments that compound a multichannel music recording. Our approach is suitable for many applications based on interactive live broadcast classical music, where a latency of a few seconds can be assumed by online users. The obtained results show that it is possible to reach real-time in the tested scenarios assuming a low latency and combining multi-core architectures with parallel and high performance techniques.

Keywords Source Separation · Real-time · Audio-to-score aligment · Parallel computing · NMF · DTW

A.J. Muñoz-Montoro
Department of Telecommunication Engineering, Universidad de Jaén, Spain
E-mail: jmontoro@ujaen.es

D. Suarez-Dou
Department of Computer Science, Universidad de Oviedo, Spain
E-mail: suarezddavid@uniovi.es

J.J. Carabias-Orti
Department of Telecommunication Engineering, Universidad de Jaén, Spain
E-mail: carabias@ujaen.es

F.J. Canadas-Quesada
Department of Telecommunication Engineering, Universidad de Jaén, Spain
E-mail: fcanadas@ujaen.es

J. Ranilla
Department of Computer Science, Universidad de Oviedo, Spain
E-mail: ranilla@uniovi.es

1 Introduction

Music source separation (SS) is the task of recovering each original instrument from a polyphonic mixture of multiple musical instruments. Currently, music source separation methods are mainly used in research for several audio processing task, such as music remixing [37], automatic karaoke [21], instrument-wise equalization [22], music information retrieval systems [16], audio compression [35], etc. Nevertheless, these methods have been rarely exploited in real-world applications, because of their high computational cost and slow execution times that prevent their use in a wide spectrum of practical situations.

During the last decade, due to the growing demand for low cost computing devices and the desire to enjoy music content in streaming mode on these embedded devices, real-time audio processing is becoming an important research topic in the field of music. In this way, SS is used for transcription applications [12], sound event detection [13] or interactive music editing applications (Celemony Software GmbH[1]). On the other hand, there are many music SS applications where response time is not a restriction. These are often called *offline* applications and require the whole audio signal to estimate the sources, and hence cannot be implemented in real-time. In these cases, the main goal is to perform the separation with the highest possible perceptual quality results while keeping interference between the sources to a minimum. These algorithms are useful as a preprocessing stage for other tasks, such as structured coding [36], beat tracking [10], audio restoration [6], etc. Note that real-time systems perform the separation as the audio is acquired, using only past and present information, meanwhile in the offline approach the audio performance to be processed is available as a whole. This fact makes real-time performance less accurate than offline performance. In this context, many recent efforts in the field have been focused on improving the robustness and speed of music real-time systems, making them appropriate for mobile devices and for a wide range of real-life contexts [2,1,28].

Nowadays, most commercial music recordings are available in multichannel format, such as stereo and 5.1. The number of channels available to perform music separation is a key factor that can be exploited when designing music SS models. Multichannel mixtures allow spatial positioning of the music sources in the sound field. This spatial positioning models the contribution of each musical source in each channel. However, high SS quality results are not possible to reach using only the multichannel spatial property [33]. In that sense, the knowledge of additional prior information could be exploited to improve the separation, such as the instrumentation [24], musical score information [25,20,14], spectral information about the sources [30], information about the recording/mixing conditions [15], etc.

In this paper, we propose a SS framework for multichannel audio signals of high polyphonic complexity providing a high SS quality by developing a low latency multichannel system. In this context, the tested scenarios show

[1] `www.celemony.com`

that it is possible to reach real-time combining parallel and high performance techniques with multi-core architectures, the most common architecture in today's computer systems.

The proposed approach is suitable for live broadcast platforms of classical music such as MyOperaPlayer[2] or Medici.TV[3], since a latency of a few seconds can be assumed by online users in exchange for an interactive experience, where they can select their favorite personal mix switching between enjoying the full performance or concentrating on a specific instrument. For this purpose, we decompose this problem in three stages: (1) the audio-to-score alignment, (2) the separation process and (3) the source reconstruction stage. The first stage is based on the ReMAS [1] system: a parallel and efficient Real-time Musical Alignment System. In the second, the estimation of each source spectrogram for each channel is carried out using a non-negative matrix factorization (NMF) approach. In the final stage, a Wiener filtering method is computed for modeling the energy contribution of each instrument of the audio mixture.

In this work, significant improvements have been carried out that clearly differentiate it from our previous work [28]. First, note that ReMAS, which was optimized for low-power processors such as ARM processors, applies the alignment process frame-by-frame. In the new context, the frame-by-frame mechanism is not proper, having been necessary to redesign ReMAS for addressing a block-by-block approach. Here, we propose to split up the input signal in consecutive audio segments (i.e. audio blocks) with the aim of reducing the memory consumption in long musical pieces. Then, the alignment process is performed over each segment consecutively. This alignment process has been enriched by the incorporation of a traceback stage which improves the alignment results at the expense of a complexity increment. Second, a novel multichannel signal model which encodes the score information within the model is proposed. In this sense, the separation stage has been implemented using a different non-negative matrix factorization (NMF) approach. Therefore, the multiplicative update rules have had to be analysed and efficiently implemented according to the proposed model. Finally, the reconstruction stage has been also redesigned for addressing the complexity increment of the novel signal model.

According to the best of our knowledge, there has not yet been presented a holistic, flexible and free system that addresses this problem on parallel shared memory systems. As a proof of concept, some experiments are carried out on a dataset of chamber and orchestral music, showing reliable results in terms of sound quality.

The structure of the article is organized as follows. In Sect. 2, we present the proposed framework and describe the main function and methodology of each of the stages that compose it. In Sect. 3, the evaluation set-up is presented

[2] `www.myoperaplayer.com`

[3] `www.medici.tv`

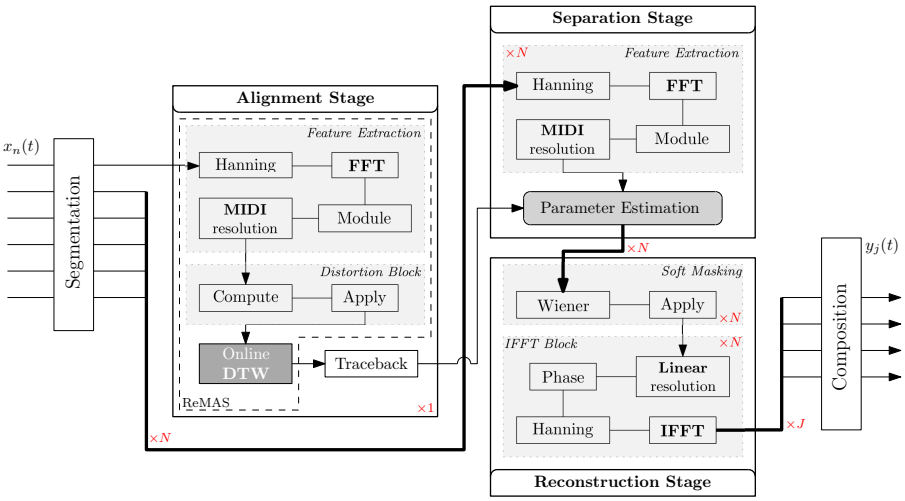


Fig. 1: Block diagram of the proposed framework.

and the experimental results are shown. Finally, conclusions are summarized in Sect. 4.

2 Framework description

2.1 System overview

In this work, we propose the development of a low latency multichannel SS system for musical pieces of high polyphonic complexity. In particular, we present a framework capable of aligning the musical score and the performance first, and then using this alignment to guide the separation process of all the instruments which compound the original mixture.

Unlike real-time systems in which each audio frame is processed in series, in the offline applications the entire recordings of the performance are available in advance for processing. However, processing long recordings directly is not possible due to the great impact that the duration of these recordings has over the system in terms of memory resources. To mitigate this problem, we propose to split up the input audio signals into blocks of a configurable duration. These blocks are processed in series providing a hybrid behaviour. In this way, our system performs the SS in real-time with a latency equal to the duration of these blocks.

The block diagram of the proposed framework is displayed in Fig. 1. As can be observed, the issue has been decomposed into three main stages: alignment, separation and reconstruction. The following subsections detail the main function and the procedure of each stage.

2.2 Alignment stage

The main aim of this stage is to synchronize the audio signal of the musical performance with its corresponding score to guide the separation process. In this way, each temporal frame t of the input audio signal is matched with each MIDI score event s. For this purpose, we have redesigned our real-time algorithm ReMAS [1] and incorporated a traceback module (see Fig. 1).

In order to reduce the impact that long recordings have over the system in terms of memory resources, we propose to divide the signal into blocks of a fixed duration. Then, each block is processed in series by the proposed framework.

For each new audio segment that arrives for being processed, we perform a novel extension of the audio-to-score alignment presented in [8] and implemented in [1]. In particular, Carabias et al. [8] proposed to compute a distortion matrix $\mathbf{D} \in \mathbb{R}_{+}^{S\times T}$ as the β-divergence between two features sequences related to the score and the input audio frames. Firstly, the score information is arranged into score units (i.e. combinations of notes annotated in the score) and then, a single spectral pattern for each of them is learnt. These spectral patterns compound the score features sequence. Afterward, a vector of features per each frame t is computed as its conversion to the frequency domain. A deep description of this theoretical procedure can be found in [8,1].

Once the distortion matrix $\mathbf{D}$ is generated for each frame t in the block and each instant s in the score, the general alignment is computed using dynamic time warping (DTW). In this way, an accumulated cost matrix $\mathbf{C} \in \mathbb{R}_{+}^{S\times T}$ is defined using the following recursion,

$$\mathbf{C}(s,t) = \min_{c_s,c_t} \left\{ \begin{array}{l} \mathbf{C}(s-1, t-c_t) + \mathbf{D}(s,t)\sigma_{1,c_t} \\ \mathbf{C}(s-c_s, t-1) + \mathbf{D}(s,t)\sigma_{c_s,1} \end{array} \right\} \tag{1}$$

where c_s and c_t are the step sizes at each dimension, whose values are the integers in the range $c_s \in [1, C_s]$ and $c_t \in [1, C_t]$. Scalars C_s and C_t are then the maximum allowed step sizes in the score and in the performance, respectively. The weights σ control the bias toward diagonal steps, being set to $\sigma_{x,y} = \sqrt{x^2+y^2}$. Observe that $\mathbf{C}(s,t)$ is the accumulated cost matrix of the minimum-cost path from $(1,1)$ to (s,t), and that $\mathbf{C}(1,1)$ is initialized to $\mathbf{D}(1,1)$, because the alignment result is constrained to include the point $(1,1)$ of the audio block. Note that given a time frame t, Eq. 1 is applied independently for each score unit s in order to consider all possible paths throughout the score up to the instant t. In this regard, a parallel implementation of Eq. 1 has been carried out to reduce the impact of long musical compositions. Observe that the number of paths to be computed grows when the score increases (i.e. when the number of score units rises).

ReMAS [1] simply returns the score position associated to the best forward path for each time t, that is, $s_{\texttt{out}} = \arg\min_s \mathbf{C}(s,t)$. However, here we have incorporated a traceback stage to improve the alignment results. This stage delivers the alignment result corresponding to all the audio frame of the block, obtained by computing the forward path up to the end of the audio block and

following the recursion b steps backward in the x-axis (performance), where b is the total number of frames of the block.

In order to compute a backward path with b frames, the algorithm must store the best step sizes for each (s,t) across the b frames. A matrix $\mathbf{P} \in \mathbb{R}_{+}^{S \times T}$ of step sizes is constructed as follows,

$$\mathbf{P}(s,t) = \arg\min_{c_s,c_t} \left\{ \begin{array}{l} \mathbf{C}(s-1,t-c_t) + \mathbf{D}(s,t)\sigma_{1,c_t} \\ \mathbf{C}(s-c_s,t-1) + \mathbf{D}(s,t)\sigma_{c_s,1} \end{array} \right\}, \tag{2}$$

where each element $\mathbf{P}(s,t)$ stores the pair $\{c_s, c_t\}$ corresponding to the last step of the minimum cost path up to (s,t). Since b is the temporal size of the block, $\mathbf{P}$ can be implemented in the computer as a matrix with $S \times b$ elements. Consequently, the traceback stage increases the memory consumption in comparison to online DTW.

At the last time frame T of the block, the score position corresponding to the best forward path is selected as $s_{\texttt{min}} = \arg\min_s \mathbf{D}(s,T)$, and the backward path is constructed step-by-step from $(s_{\texttt{min}}, T)$ by reading the step sizes stored in $\mathbf{P}$. Note that when a new audio block arrives, a new accumulated cost matrix $\mathbf{D}$ has to be computed considering that $\mathbf{D}(1,1)$ has to be initialized to $\mathbf{C}(s_{min}, 1)$, because the path has to continue from the last score position selected by the previous block.

However, this block-by-block approach is less accurate than analyzing the input signal as a whole. Observe that while the first frames of a block are aligned using the future information corresponding to the last frames of the block, the alignment decision for these last frames is made without the information of the future blocks. To mitigate this, we propose to overlap the blocks, that is, the first frames which compound a specific block belong to the previous block. In this way, these special frames will be aligned using the information of the following block.

Fig. 2 illustrates the proposed traceback process with an example. As can be observed, the backward path (displayed as a black line) is constructed from s_{min} for each aligned block (displayed as a rectangle of dashed red lines). In this example, the last two frames of one block are the fist two frames of the following block (represented in gray). As a result, the global path (displayed as a green line) differs from the local paths corresponding to each block. The output per analyzed block of this alignment stage is the path corresponding to the interval of frames $[1, T-m]$, where m is the number of overlapped frames ($m = 2$ in this example). This aligned path is used for the following stages to perform the SS of this block of size $T - m$.

The computational complexity of the overall system increases as the number of overlapped frames increases, since more blocks have to be analyzed. However, the higher number of overlapped frames, the better the alignment. In this regard, Cabañas et al. [4] analyzed the effect of the latency in the alignment accuracy achieved by ReMAS. Their experiments showed that, beyond a delay of 2 seconds, the alignment results are almost equal to the offline case. Therefore, in the development of our system we have considered this 2 seconds to compute the number of overlapped frames m.

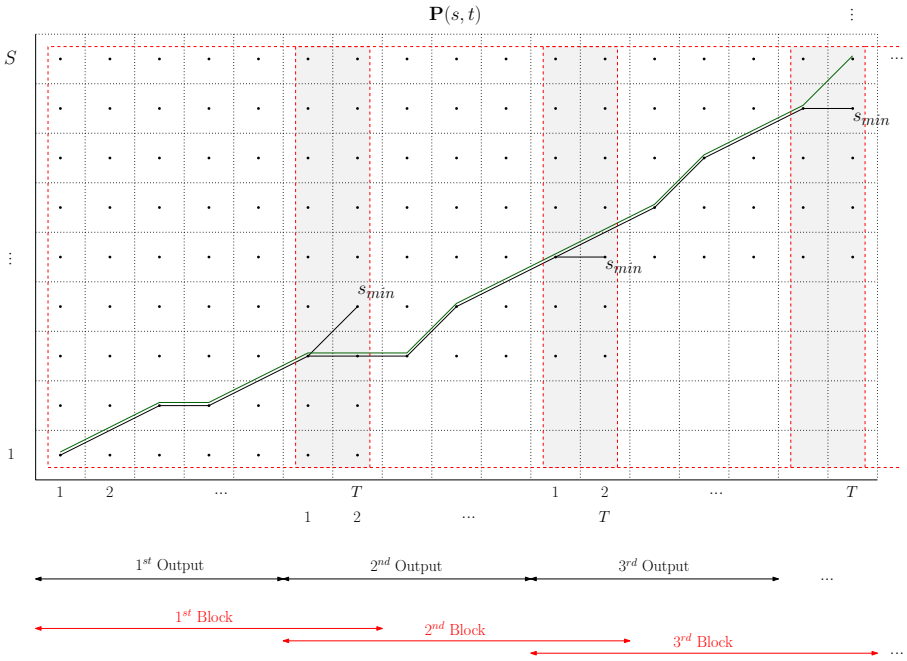


Fig. 2: The proposed block-by-block approach for the traceback stage.

For this alignment stage, the computational complexity is mainly determined by ReMAS. According to [2], where a study of the theoretical computational cost of ReMAS was analyzed, the complexity per frame of the sequential version is defined by,

$$O\Big(L_T + L_F \log_2(L_F) + PK + S\Big) \tag{3}$$

where L_F is the fast Fourier transform length, L_T is the frame length, P is the number of notes in MIDI scale, S is the number of states and K is the number of score units. For the time-frequency representation of the input signal, we have used a frame length L_T of 5700 samples ($\sim$ 129 ms) and a hop size of 570 samples ($\sim 12,9$ ms) in order to have enough temporal resolution, being the sampling rate equal to 44,1 kHz. To have enough frequency resolution for low frequency sounds, $L_F = 16384$ bins has been chosen as in [31]. P is selected as 114 that corresponds to a range of notes of 9.5-octaves in one sample per semitone of MIDI resolution. Remark that K and S rely on the musical composition and are obtained from the processing of the MIDI score.

As can be observed in Fig. 1, only an audio channel is required to perform the alignment to the MIDI score. Moreover, the traceback process is applied once the block alignment is done. Thereby, the theoretical computational complexity per block can be expressed as,

$$O\Big(T\left(L_T + L_F\log_2(L_F) + PK + S\right) + T\Big) \tag{4}$$

where T is the block size in terms of the number of frames.

Excluding the sequential implementation of the traceback process, the rest of the modules of the alignment stage have been implemented based on the following parallelization strategies: (1) using parallel routines for shared-memory of the FFTW package [17], (2) calling BLAS/LAPACK [3] Level 1 routines for vector and vector-vector operations, and (3) using parallel and worksharing constructors of OpenMP [11]. Moreover, in order to compute the minimum score unit (i.e. s_{min}) and its position among the vector of score units associated to each instant t (see Eq. 1), a parallel reduction function has been implemented for those cases when the OpenMP version is less than 4.0 or the relationship between the search space and the number of cores is not adequate. In this sense, the parallel complexity of this stage is given by,

$$O\left(T\left(\frac{L_T}{c} + L_F\log_2\left(\frac{L_F}{c}\right) + \frac{PK}{c} + \frac{c\log_2(c)+S}{c}\right) + T\right) \tag{5}$$

where c is the number of processors or cores used. As expected, the new complexity includes a minimal overhead corresponding to the built-in sequential traceback module.

2.3 Separation stage

In this stage, the magnitude spectrograms of each audio channel are estimated using a NMF approach. For this aim, we propose a multichannel extension of the signal model presented in [27]. Muñoz et al. proposed a signal model where the score information is encoded in the form of score units, so that each basis represents a unique combination of notes defined in the score. Consequently, during the factorization, this model only allows the activation of concurrent notes that are defined in the score and happen in the performance.

In this regard, we propose a novel multichannel decomposition of the input audio spectrograms $\mathbf{X} \in \mathbb{R}_+^{F\times T\times N}$ into the product of four nonnegative matrices as follows,

$$\mathbf{X}_n(f,t) \approx \hat{\mathbf{X}}_n(f,t) = \sum_{p,j} \mathbf{M}_{n,j}\mathbf{B}_{p,j}(f) \underbrace{\sum_k \mathbf{E}_{k,p,j}\mathbf{A}_k(t)}_{\mathbf{G}_{p,j}(t)} \tag{6}$$

where $\hat{\mathbf{X}} \in \mathbb{R}_+^{F\times T\times N}$ is the estimation of the magnitude spectrogram at each time-frequency (f,t) point and per each channel n; $\mathbf{B} \in \mathbb{R}_+^{F\times P\times J}$ is the basis matrix of spectral patterns for each note p and instrument j of the composition; $\mathbf{E} \in \mathbb{R}_+^{K\times P\times J}$ is the matrix of score units that encodes the active notes and instruments in each score unit k; $\mathbf{A} \in \mathbb{R}_+^{K\times T}$ is the time-varying gains matrix; $\mathbf{G} \in \mathbb{R}_+^{P\times J\times T}$ is the matrix which holds the gains of the basis $\mathbf{B}$ corresponding

to the note p and instrument j at frame t; and $\mathbf{M} \in \mathbb{R}_{+}^{N \times J}$ is the panning matrix which models the contribution of each instrument j in each channel n.

In this approach, the spectral patterns $\mathbf{B}$ are obtained in a preprocessing stage where the amplitudes of each note of the musical instruments are learned in advance by using the Real World Computing (RWC) music database [19, 18]. These pre-learned instrument models are an approximation to the real spectral patterns occurring in the mixture. Note that sometimes certain dissimilarity between training and testing instruments can be occurred due to the physical differences between the training and testing instruments, the performing style difference of the performers and the acoustical difference of the recording environments. In the literature, some authors [9,31] have proposed to adapt the learned models to the instruments in the mixed signal during the factorization process in order to fit them better with the recorded signal.

Regarding the rest of the model parameters, the score units matrix $\mathbf{E}$ is also defined in that preprocessing stage by the score. In this work both parameters $\mathbf{B}$ and $\mathbf{E}$ are fixed during the separation process. The gain matrix $\mathbf{A}$ is initialized to the output path of the alignment stage. This path indicates the score units played at each frame t.

Contrary to the alignment stage, here all the channels are used to perform the spectrograms estimation. In this sense, a Hanning-windowed FFT is firstly applied to each channel and then, their time-frequency representations are converted to a resolution of 1/4 of a semitone. This resolution has been proven to achieve better results for separation tasks [7]. Once the FFT is computed for all the channels, the factorization parameters of Eq. 6 can be estimated under the nonnegativity restriction by minimizing the β-divergence between the observed $\mathbf{X}$ and the modeled $\hat{\mathbf{X}}$ spectrograms. Therefore, we have efficiently implemented the following multiplicative update rules required for the factorization,

$$\mathbf{A}_k(t) \leftarrow \mathbf{A}_k(t) \odot \left(\frac{\sum_{f,p,j,n} \mathbf{M}_{n,j} \mathbf{B}_{p,j}(f) \mathbf{E}_{k,p,j} [\hat{\mathbf{X}}_n(f,t)^{\beta-2} \odot \mathbf{X}_n(f,t)]}{\sum_{f,p,j,n} \mathbf{M}_{n,j} \mathbf{B}_{p,j}(f) \mathbf{E}_{k,p,j} \hat{\mathbf{X}}_n(f,t)^{\beta-1}} \right) \quad (7)$$

$$\mathbf{M}_{n,j} \leftarrow \mathbf{M}_{n,j} \odot \left(\frac{\sum_{f,t,p,k} \mathbf{B}_{p,j}(f) \mathbf{E}_{k,p,j} \mathbf{A}_k(t) [\hat{\mathbf{X}}_n(f,t)^{\beta-2} \odot \mathbf{X}_n(f,t)]}{\sum_{f,t,p,k} \mathbf{B}_{p,j}(f) \mathbf{E}_{k,p,j} \mathbf{A}_k(t) \hat{\mathbf{X}}_n(f,t)^{\beta-1}} \right) \quad (8)$$

where $\beta = 1.5$.

Finally, once both parameters are estimated, the spectrogram of each source in each channel $\hat{\mathbf{S}} \in \mathbb{R}_{+}^{F \times T \times J \times N}$ can be computed in the reconstruction stage. Observe that, following the block-by-block approach, this stage outputs the parameters estimation for each block independently.

Regarding the computational complexity, the theoretical cost of the feature extraction module of the separation stage (see Fig. 1) for the parallel version can be approximated by,

$$O\left(TN\left(\frac{L_T}{c}+L_F\log_2\left(\frac{L_F}{c}\right)\right)\right) \tag{9}$$

where N is the number of channels of the music performance. As in the alignment stage, the same parallelization strategies have been used, i.e. exploiting the parallel FFTW, BLAS/LAPACK and OpenMP routines.

The overall cost for the parameter estimation module (see Fig. 1) has been computed considering that the multiplicative update rules of Eq. 7 and Eq. 8 have been implemented using BLAS/LAPACK. Thus, for each source (i.e instrument) presented in the mixture, the implementation of Eq. 7 leads two matrix-matrix products (dgemm subroutine) and two vector-vector products (daxpy subroutine), together with other auxiliary operations of lower computational intensity. Likewise, the implementation of Eq. 8 requires the same type and number of operations, but in this case for each MIDI note p. Thereby, the theoretical computational cost can be approximated by,

$$O\Big(I\Big[\overbrace{F\big(TJP+KNJ+2TKN\big)}^{\text{Eq. 7}}+ \\ +\underbrace{TJ\big(FP+2NF+KP\big)}_{\text{Eq. 8}}\Big]\Big) \tag{10}$$

where J is the number of instruments of the composition, I is the number of iterations of the NMF approach and F is the frequency in MIDI resolution sampled (logarithmic resolution instead of linear resolution). For music SS purposes, F is fixed to 401, one sample per quarter of semitone. Note that, in this approximation, F and P are fixed parameters and are not dependent on the composition. In this sense, some assumptions can be considered. In general, J and N are significantly smaller than T and P, respectively. Furthermore, considering that in most recordings of orchestral instrumental ensembles there is usually one microphone channel for each type of instrument, we can assume that $J \approx N$. Therefore, the cost can be expressed as,

$$O\Big(L_F ITJ\big(2P+3K\big)\Big). \tag{11}$$

2.4 Reconstruction stage

Finally, the reconstruction of each source in each channel is computed using a soft-filter strategy. Firstly, the estimated parameters of the signal model presented in Eq. 6 are used to predict the magnitude spectrograms of each source j in each channel n by

$$\hat{\mathbf{S}}_{n,j}(f,t)=\sum_{k,p}\mathbf{M}_{n,j}\mathbf{B}_{p,j}(f)\mathbf{E}_{k,p,j}\mathbf{A}_k(t). \tag{12}$$

Afterward, we apply a Wiener filter to reconstitute the different sources of the mixture based on the power spectrum ratio between the reference signals. Once the spectrograms are estimated, soft masking $\mathbf{V} \in \mathbf{R}_{+}^{F \times T \times J \times N}$ is computed for each source and channel,

$$\mathbf{V}_{n,j}(f,t) = \frac{|\hat{\mathbf{S}}_{n,j}(f,t)|^2}{\sum_j |\hat{\mathbf{S}}_{n,j}(f,t)|^2} \tag{13}$$

where $\mathbf{V}$ represents the relative energy contribution of each source j in each channel n for each time-frequency bin. Then, to obtain the estimated source magnitude spectrogram $\mathbf{Y} \in \mathbb{R}_{+}^{F \times T \times J \times N}$, Eq. 13 is used as follows,

$$\mathbf{Y}_{n,j}(f,t) = \sqrt{\mathbf{V}_{n,j}(f,t)} \cdot \mathbf{X}_n(f,t). \tag{14}$$

Finally, the IFFT block is performed (see Fig. 1). The time representation is converted from MIDI resolution to linear frequency allocating the value of a MIDI bin to its corresponding frequency bins. Afterwards, the phase spectrogram of the input mixture is applied for each source. Then, a windowed inverse fast Fourier transform (IFFT) is computed with the same features as in the FFT block. The procedure of the whole system is summarized in Algorithm 1.

Algorithm 1 Proposed system algorithm

```
Load B and E from the preprocessing stage.
Divide the audio signal into M blocks.
for m=1 to M do
    Read audio frame block x_n(t).
    Compute ReMAS with the traceback stage to obtain the optimal path.
    Compute the FFT to each channel and change to a 1/4 of a semitone MIDI resolution.
    for it=1 to I do
        Update the gains matrix using the Eq. 7.
        Update the panning matrix using the Eq. 8.
    end for
    for n=1 to N do
        Compute Wiener mask using the Eq. 13.
        for j=1 to J do
            Estimate the spectrogram of each source Y using Eq. 14.
            Change to linear resolution.
            Compute the IFFT.
    end for
end for
```

From the computational point of view, Eq. 13 and Eq. 14 have been implemented exploiting the spatial locality. In this manner, the data has been stored using matrices whose dimensions have been arranged according to the subsequent use. Accordingly, the Wiener filter is implemented using one vector-matrix product (BLAS/LAPACK dgemv subroutine) for each source and block. The rest of the operations have been implemented using the FFTW

package and OpenMP, as in the previous stages. Regarding the use of OpenMP, there are loops in this stage where the cost of their iterations depends on different search spaces (non-uniform cost), becoming the computational intensity of some loop iterations irrelevant with respect to the rest. To mitigate this situation, in addition to using the appropriate OpenMP clauses (dynamic/guided scheduling), nested parallelism is enabled in certain parts of this stage. Thus, some processes, such as sequential operations, initializations, etc., can concur with other operation more expensive (e.g. BLAS/LAPACK or IFFTs), improving the use of resources. Finally, the temporal complexity of this stage can be expressed by,

$$O\left(TNJ\left(FP+\frac{L_F\log_2(L_F)}{c}\right)\right). \tag{15}$$

3 Evaluation and experimental results

This section presents the experimentation carried out in the evaluation of the proposed SS framework. In this evaluation, we have exploited two different databases. Firstly, we have used a subset of the University of Rochester Multimodal Music Performance (URMP) dataset presented in [23]. This subset is compounded by 4 classical chamber music pieces ranging from duets to quintets and played by 9 different common instruments in orchestra. The musical score, the audio recordings of the individual tracks, the audio recordings of the assembled mixture and ground-truth annotation files are available for each piece. Secondly, we have used the orchestra database developed by Pätynen et al. [29]. It consists of four excerpts of approximately 3 minutes each one composed by symphonic music from Classical and Romantic style. The four pieces vary in terms of number of instruments sections, style, dynamics, and size of the orchestra. More details of both databases are provided in Table 1.

Composer	**Piece name**	**Dur.**	**Channels**	**Instr.**	**Score units**
Vivaldi	Spring	0m 35s	3	2	27
Chopin	Nocturne	1m 36s	4	3	103
Bach	The Art of the Fugue	2m 52s	5	4	530
Hubert	Jerusalem	1m 59s	6	5	224
Mozart	Don Giovanni	3m 47s	10	8	1046
Beethoven	Symphony no. 7	3m 11s	10	10	604
Bruckner	Symphony no. 8	1m 27s	10	10	567
Mahler	Symphony no. 1	2m 12s	10	10	1176

Table 1: Characteristics of the chamber and orchestral dataset used for the evaluation of our SS system.

For both datasets, the multichannel mixtures have been generated by simulating the spatial position of all sources in a room. We have used the software

Roomsim [5] wiht the same setup employed in [26] to generate the room impulse responses.

Regarding the testbed, we have focused our interest on two different systems. Firstly, we have used a server with an Intel® Xeon® Silver 4110 processor with 8 cores. It operates at 2.1 GHz and runs CentOS Linux 7. Note that HyperThreading and Turbo Boost are both deactivated. Secondly, the experiments were conducted on a NVIDIA Jetson AGX Xavier development kit with an eight-core ARM v8.2 CPU always operating at 2.26 GHz (adjusted by the *NVPModel* command tool) and running Ubuntu Linux 16.04. Both systems use the same version of the OpenBlas and FFTW packages, as well as the GNU C Compiler 7, supporting the specification 4.5 of OpenMP. Finally, it should be remarked that the used data type is "*double*" (i.e. IEEE 754 double-precision binary floating-point format).

3.1 Experimental results

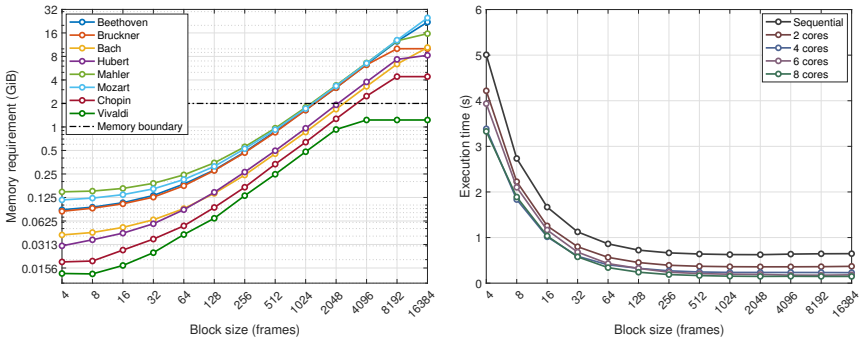
Firstly, we have measured the execution time and memory consumption depending on the number of frames per audio block (i.e. the block size). Fig. 3 illustrates the obtained results for all pieces described in Table 1.

As shown, the memory consumption considerably increases as the size of the audio block grows (see Fig. 3a). Performances with a high number of instruments and/or recording channels (i.e. Mozart, Beethoven, Bruckner and Mahler) require a larger amount of memory, regardless of the block size. This results from the fact that all parameters of the signal model, except the gains matrix $\mathbf{A}$ (see Sect. 2.3), directly depend on these two variables.

On the other hand, as the block size becomes longer, the execution time decreases (see Fig. 3b). This improvement is only noticeable for blocks of a short size where the decay is significant, while the execution time is always very similar for blocks of a size longer than 1024 frames. The reason is that NMF, which has been mainly implemented by matrix–matrix operations, has a strong impact on the overall complexity of the system. Note that BLAS/LAPACK stabilizes its speedup when the size of the matrices is reasonably large. In short, this is an important finding for our system, because it reveals that the execution time does not decrease for blocks longer than 1024 frames.

This analysis demonstrates that choosing a block size longer than 1024 frames implies a high memory cost (more than 2 GB) without any improvement in terms of execution time. Moreover, as explained in Sect. 2.2, at least two seconds of audio (i.e. 156 frames with the hop size used between frames) are required to achieve reliable alignment results. Therefore, for the sake of brevity, we propose to consider a block size of 1024 frames for carrying out a deeper study of the performance of our proposal.

Secondly, we have evaluated our proposal as a function of the number of computing cores used simultaneously for a fixed block size of 1024 frames. Since the total execution time varies among the total duration of each musical piece,



(a) Memory requirement measured in GiB. (b) Execution times measured in seconds.

Fig. 3: Experimental results as a function of the block size in frames.

we have considered normalizing it to that duration for a better comparison. The results obtained by the Intel® Xeon® Silver 4110 are depicted in Fig. 4.

As previously explained in Sec. 2, the computational complexity of the whole system depends mainly on three variables when the block size is fixed (see Eq. 5, Eq. 11 and Eq. 15): the duration of the score (i.e. the number of score units), the number of recording channels and the number of instruments of the composition. In this sense, regarding the number of instruments and channels, only the chamber music performances, where a low number of instruments and channels are presented (i.e. Bach, Hubert, Chopin and Vivaldi), can be run in real-time using parallel computing. Note that real-time is guaranteed when the normalized execution time is lower than 1. In the case of orchestral compositions (i.e. Beethoven, Bruckner, Mahler and Mozart), it is not possible to reach real-time due to the combination of a relatively high score duration and a large number of instruments and channels. However, this limitation can be sorted out just by reducing the number of output audio files. Note that this implementation outputs an audio file per each channel and instrument. Nevertheless, from a music point of view, only the audio file corresponding to the best separation per each instrument is required. Therefore, the number of output files can be reduced by up to one per instrument. Moreover, the number of iterations of the NMF approach can be also downsized without affecting the convergence of the algorithm at the expense of losing some quality in the separation.

Attending to the duration of the score, Mozart and Mahler are the most demanding in terms of score units. For all cases, both obtain higher execution times than Beethoven and Bruckner despite having the same number of channels and instruments.

Regarding the efficiency, the behavior obtained by the system is as expected. Note that some memory bound operations of the system are performed as matrix–vector, such as the Wiener filter, Eq. 7 and Eq. 8. Therefore, the se-

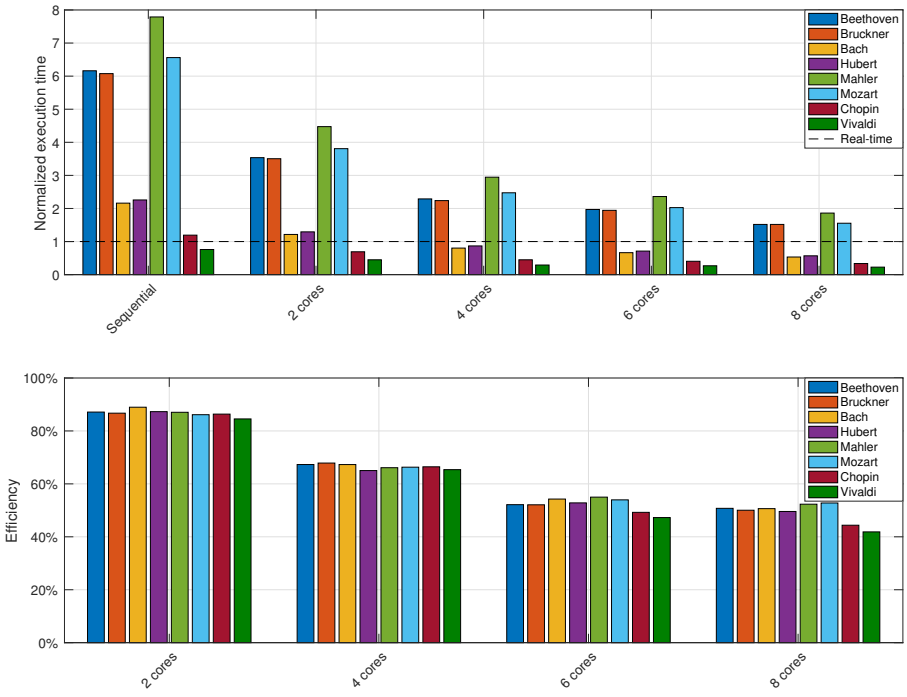


Fig. 4: Evolution of the normalized execution time and efficiency on the Intel® Xeon® Silver 4110 for each composition as a function of the number of cores used simultaneously.

quential approach maximizes the performance, taking advantage of the whole memory bandwidth, while a parallel approach is limited by this fact.

Additionally, in order to assess the upper bound of the fastest separation that can be reached with our proposal, we have reduced the number of output audio files to one per instrument and the number of iterations of the NMF algorithm guaranteeing its convergence. Table 2 summarized the execution time in seconds, the normalized execution time and the efficiency on the Intel® Xeon® Silver 4110 using 8 cores for each stage and for each orchestral composition.

In view of the results, it can be concluded that real-time is achieved for all composition except Mahler. In this exceptional case, reducing the FFT length would give the system the opportunity to be executed in real-time, affecting only the quality of low-pitch sounds in separation. Another possible solution could be to use "*float*" as the data type. This would reduce the memory consumption and decrease the execution time without loosing the accuracy of perceptual quality results. As can be observed, the separation stage, which consumes between 74% and 81% of the total execution time, means a high load in the system, while the alignment and reconstruction stages barely account for 2% and 15%, respectively.

Composition	***Beethoven***	***Bruckner***	***Mahler***	***Mozart***
Stage			*Execution Time (s)*	
Alignment	3.99	1.88	3.25	6.04
Separation	122.85	54.52	111.98	168.37
Reconstruction	39.95	18.00	27.38	33.03
			Normalized Execution Time	
Alignment	0.02	0.02	0.02	0.03
Separation	0.64	0.63	0.85	0.74
Reconstruction	0.21	0.21	0.21	0.15
				Efficiency
Alignment	28%	25%	42%	41%
Separation	50%	50%	52%	52%
Reconstruction	55%	56%	54%	57%

Table 2: Execution times measured in seconds, normalized execution times and efficiency on the Intel® Xeon® Silver 4110 for each stage and audio composition.

As for the NVIDIA Jetson AGX Xavier, the experimental results obtained in the evaluation are provided in Table 3. As the results obtained by the orchestral performances do not reach real-time, for the sake of brevity, only the chamber music results are shown. As can be observed, real-time can be reached for all the chamber compositions when more than 6 cores are used. The results obtained for the efficiency is always above 48%, even in the worst-case scenario.

Composition	***Bach***	***Hubert***	***Nocturne***	***Spring***
Cores			*Execution Time (s)*	
1	632.94	397.83	161.42	36.76
2	355.83	234.59	107.78	23.98
4	200.83	144.58	58.64	13.68
6	142.46	92.36	42.01	11.01
8	108.35	72.01	33.74	9.66
			Normalized Execution Time	
1	3.62	3.32	1.68	1.02
2	2.03	1.95	1.12	0.67
4	1.15	1.2	0.61	0.38
6	0.81	0.77	0.44	0.31
8	0.62	0.6	0.35	0.27
				Efficiency
2	89%	85%	75%	77%
4	79%	69%	69%	67%
6	74%	72%	64%	56%
8	73%	69%	60%	48%

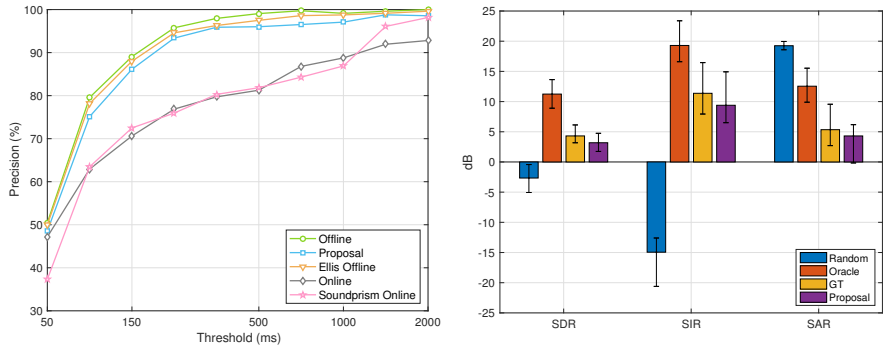
Table 3: Execution times measured in seconds, normalized execution times and efficiency on the the NVIDIA Jetson AGX Xavier for each audio composition as a function of the number of cores used.

Finally, we have analyzed the performance of our proposal evaluating the alignment accuracy and the separation results. For this evaluation, we have also considered a fixed block size of 1024 frames. First, we have compared the alignment performance with other offline and online reference methods: (a) Carabias's offline [8], (b) Carabias's online [8], (c) Ellis' offline [32], and (d) Soundprism [14]. Figure 5a illustrates the alignment results in terms of precision values of the analyzed methods as a function of the onset deviation threshold. The threshold value varies from 50 to 2000 ms. As can be observed, our block-by-block proposal reaches results very similar to the offline approaches and clearly outperforms the online cases. This corroborates the assumption made in Section 2.2 that a block size greater than 2 seconds guarantees an alignment accuracy almost equal to the offline case.

In order to assess the separation performance, we have used BSS_EVAL [34] toolbox, which provides three metrics: *Source to Distortion Ratio* (SDR), *Source to Interference Ratio* (SIR) and *Source to Artifacts Ratio* (SAR). These metrics are commonly accepted in the field of source separation and thus, allow a fair comparison with other state-of-the-art methods. In this sense, different configurations of the proposed separation framework has been compared in order to show its separation capabilities. First, *Oracle* computes the optimal value of the Wiener mask at each frequency and time component assuming that the signals to be separated are known in advance. This approach represents the upper bound for the best separation that can be reached with the used time-frequency representation. *Rand* uses random values at each frequency and time component as input for the evaluation. This provides a starting point for the separation algorithms. Finally, we have included an oracle variant denoted as ground-truth GT which uses the score manually annotated by musicologist as a hard prior initialization for the times varying gains. Figure 5b depicts the median values of SDR, SIR and SAR obtained in the evaluation of the proposed database for each approach. As can be seen, the proposed framework (SDR = 3,18 dB, SIR = 9,39 dB, SAR = 4,30 dB) provides competitive separation results, and comparable with the oracle GT solution, which informs about the best separation results that can be obtained using our method when perfect annotation is available.

4 Conclusion

In this paper, we present a parallel multichannel SS system. Our approach has focused on achieving real-time execution assuming a latency of a few seconds and reaching reliable results in terms of sound quality. Under these conditions, we have decomposed the system in three main stages. First, an alignment stage based on DTW is performed to synchronize the audio signal of the musical piece with its corresponding score. Then, the magnitude spectrograms of each audio channel are estimated using a NMF approach during the separation stage. In this sense, a novel multichannel signal model has been proposed to



(a) Alignment evaluation in terms of precision values in function of the tolerance window.

(b) Source separation evaluation of the proposed framework using the BBS_EVAL metrics.

Fig. 5: Evaluation of the alignment accuracy and the separation performance.

encode the score information in form of score units. Finally, the reconstruction of each source in each channel is computed using a soft-filter strategy.

The proposed framework has been implemented for multi-core architectures and, thus, the application can be executed in a wide range of devices. Moreover, the system has been evaluated considering chamber and orchestral compositions and reaching real-time in most of the cases. To our best knowledge, our proposal is the first multichannel implementation in real-time that obtains reliable results in terms of sound quality for highly polyphonic ensembles.

Acknowledgements This work has been supported by the Regional Ministry of the Principado de Asturias under grants *FC-GRUPIN-IDI/2018/000226*, the Government of the Junta de Andalucía under grants *"PAIDI 2020. Convocatoria 2017 de Ayudas a Actividades de Transferencia de Conocimiento"* with reference **AT-6044**, and the University of Jaén under the program *"Acción 1. Apoyo a las estructuras de investigación de la Universidad de Jaén para incrementar su competitividad atendiendo a sus singularidades"*.

References

1. Alonso, P., Cortina, R., Rodríguez-Serrano, F.J., Vera-Candeas, P., Alonso-González, M., Ranilla, J.: Parallel online time warping for real-time audio-to-score alignment in multi-core systems. The Journal of Supercomputing **73**(1), 126–138 (2017). DOI 10.1007/s11227-016-1647-5
2. Alonso, P., Vera-Candeas, P., Cortina, R., Ranilla, J.: An efficient musical accompaniment parallel system for mobile devices. Journal of Supercomputing **73**(1), 343–353 (2017). DOI 10.1007/s11227-016-1865-x
3. Anderson, E., Bai, Z., Bischof, C., Blackford, S., Demmel, J., Dongarra, J., Du Croz, J., Greenbaum, A., Hammarling, S., McKenney, A., Sorensen, D.: LAPACK Users' Guide, third edn. Society for Industrial and Applied Mathematics, Philadelphia, PA (1999)
4. Cabañas-Molero, P., Cortina-Parajón, R., Combarro, E.F., Alonso, P., Bris-Peñalver, F.J.: HReMAS: hybrid real-time musical alignment system. The Journal of Supercomputing **75**(3), 1001–1013 (2019). DOI 10.1007/s11227-018-2265-1

5. Campbell, D.R., Palomaki, K.J., Brown, G.J.: A MATLAB simulation of "shoebox" room acoustics for use in teaching and research. Computing and Information Systems J (2005)
6. Canadas-Quesada, F., Vera-Candeas, P., Martinez-Munoz, D., Ruiz-Reyes, N., Carabias-Orti, J., Cabanas-Molero, P.: Constrained non-negative matrix factorization for score-informed piano music restoration. Digital Signal Processing **50**, 240–257 (2016). DOI 10.1016/j.dsp.2016.01.004
7. Carabias-Orti, J.J., Cobos, M., Vera-Candeas, P., Rodríguez-Serrano, F.J.: Nonnegative signal factorization with learnt instrument models for sound source separation in close-microphone recordings. EURASIP Journal on Advances in Signal Processing **2013**(1), 184 (2013). DOI 10.1186/1687-6180-2013-184
8. Carabias-Orti, J.J., Rodriguez-Serrano, F., Vera-Candeas, P., Ruiz-Reyes, N., Canadas-Quesada, F.J.: An audio to score alignment framework using spectral factorization and dynamic time warping. ISMIR: proceedings of the International Conference of Music Information Retrieval pp. 742–748 (2015)
9. Carabias-Orti, J.J., Virtanen, T., Vera-Candeas, P., Ruiz-Reyes, N., Canadas-Quesada, F.J.: Musical instrument sound multi-excitation model for non-negative spectrogram factorization. IEEE Journal on Selected Topics in Signal Processing **5**(6), 1144–1158 (2011). DOI 10.1109/JSTSP.2011.2159700
10. Chordia, P., Rae, A.: Using source separation to improve tempo detection. In: Proceedings of the 10th International Society for Music Information Retrieval Conference, ISMIR 2009, pp. 183–188 (2009)
11. Dagum, L., Menon, R.: Openmp: an industry standard api for shared-memory programming. Computational Science & Engineering, IEEE **5**(1), 46–55 (1998)
12. Dessein, A., Cont, A., Lemaitre, G.: Real-time polyphonic music transcription with non-negative matrix factorization and beta-divergence. In: Proceedings of the 11th International Society for Music Information Retrieval Conference, ISMIR 2010, pp. 489–494 (2010)
13. Dessein, A., Cont, A., Lemaitre, G.: Real-Time Detection of Overlapping Sound Events with Non-Negative Matrix Factorization. In: Matrix Information Geometry, pp. 341–371. Springer Berlin Heidelberg, Berlin, Heidelberg (2013). DOI 10.1007/978-3-642-30232-9{_}14
14. Duan, Z., Pardo, B.: Soundprism: An Online System for Score-Informed Source Separation of Music Audio. IEEE Journal of Selected Topics in Signal Processing **5**(6), 1205–1215 (2011). DOI 10.1109/JSTSP.2011.2159701
15. Duong, N.Q.K., Vincent, E., Gribonval, R.: Under-Determined Reverberant Audio Source Separation Using a Full-Rank Spatial Covariance Model. IEEE Transactions on Audio, Speech, and Language Processing **18**(7), 1830–1840 (2010). DOI 10.1109/TASL.2010.2050716
16. Ewert, S., Muller, M.: Estimating note intensities in music recordings. In: 2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 385–388. IEEE (2011). DOI 10.1109/ICASSP.2011.5946421
17. Frigo, M.: A fast fourier transform compiler. ACM SIGPLAN Notices **34** (1999). DOI 10.1145/989393.989457
18. Goto, M.: Development of the RWC music database. Proceedings of the 18th International Congress on Acoustics (ICA 2004) (April), 553–556 (2004)
19. Goto, M., Hashiguchi, H., Nishimura, T., Oka, R.: RWC Music Database: Popular, Classical and Jazz Music Databases. Ismir **2**(October), 287–288 (2002)
20. Hennequin, R., David, B., Badeau, R.: Score informed audio source separation using a parametric model of non-negative spectrogram. In: 2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 1, pp. 45–48. IEEE (2011). DOI 10.1109/ICASSP.2011.5946324
21. Huang, P.S., Chen, S.D., Smaragdis, P., Hasegawa-Johnson, M.: Singing-Voice Separation From Monaural Recordings Using Robust Principal Component Analysis. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) pp. 57–60 (2012)
22. Itoyama, K., Goto, M., Komatani, K., Ogata, T., Okuno, H.G.: Instrument Equalizer for Query-by-Example Retrieval: Improving Sound Source Separation Based on Integrated Harmonic and Inharmonic Models. Ismir (2008). DOI 10.1136/bmj.324.7341.827

23. Li, B., Liu, X., Dinesh, K., Duan, Z., Sharma, G.: Creating a Multitrack Classical Music Performance Dataset for Multimodal Music Analysis: Challenges, Insights, and Applications. IEEE Transactions on Multimedia **21**(2), 522–535 (2019). DOI 10.1109/TMM.2018.2856090
24. Marxer, R.: Audio Source Separation for Music in Low-latency and High-latency Scenarios (2013)
25. Miron, M., Carabias, J.J., Janer, J.: Improving Score-Informed Source Separation for Classical Music Through Note Refinement. Proceedings of the International Society for Music Information Retrieval {(ISMIR)} Conference pp. 448–454 (2015)
26. Miron, M., Carabias-Orti, J.J., Bosch, J.J., Gómez, E., Janer, J.: Score-Informed Source Separation for Multichannel Orchestral Recordings. Journal of Electrical and Computer Engineering **2016**, 1–19 (2016). DOI 10.1155/2016/8363507
27. Munoz-Montoro, A.J., Carabias-Orti, J.J., Vera-Candeas, P., Canadas-Quesada, F.J., Ruiz-Reyes, N.: Online/offline score informed music signal decomposition: application to minus one. EURASIP Journal on Audio, Speech, and Music Processing **2019**(1), 23 (2019). DOI 10.1186/s13636-019-0168-6
28. Muñoz-Montoro, A.J., Ranilla, J., Vera-Candeas, P., Combarro, E.F., Alonso-Jordá, P.: Real-time Soundprism. The Journal of Supercomputing pp. 1–16 (2018). DOI 10.1007/s11227-018-2703-0. URL `http://link.springer.com/10.1007/s11227-018-2703-0`
29. Pätynen, J., Pulkki, V., Lokki, T.: Anechoic Recording System for Symphony Orchestra. Acta Acustica united with Acustica **94**(6), 856–865 (2008). DOI 10.3813/AAA.918104
30. Rodríguez-Serrano, F.J., Carabias-Orti, J.J., Vera-Candeas, P., Canadas-Quesada, F.J., Ruiz-Reyes, N.: Monophonic constrained non-negative sparse coding using instrument models for audio separation and transcription of monophonic source-based polyphonic mixtures. Multimedia Tools and Applications **72**(1), 925–949 (2014). DOI 10.1007/s11042-013-1398-8
31. Rodriguez-Serrano, F.J., Duan, Z., Vera-Candeas, P., Pardo, B., Carabias-Orti, J.J.: Online Score-Informed Source Separation with Adaptive Instrument Models. Journal of New Music Research **44**(2), 83–96 (2015). DOI 10.1080/09298215.2014.989174
32. Turetsky, R., Ellis, D.: Ground-Truth Transcriptions of Real Music from Force-Aligned MIDI Syntheses. Proceedings of the 4th International Symposium on Music Information Retrieval pp. 135–141 (2003). DOI 10.7916/D8S472CZ. URL `https://academiccommons.columbia.edu/doi/10.7916/D8S472CZ`
33. Vincent, E., Araki, S., Theis, F., Nolte, G., Bofill, P., Sawada, H., Ozerov, A., Gowreesunker, V., Lutter, D., Duong, N.Q.: The signal separation evaluation campaign (2007–2010): Achievements and remaining challenges. Signal Processing **92**(8), 1928–1936 (2012). DOI 10.1016/j.sigpro.2011.10.007
34. Vincent, E., Gribonval, R., Fevotte, C.: Performance measurement in blind audio source separation. IEEE Transactions on Audio, Speech and Language Processing **14**(4), 1462–1469 (2006). DOI 10.1109/TSA.2005.858005. URL `http://ieeexplore.ieee.org/document/1643671/`
35. Vincent, E., Plumbley, M.: A prototype system for object coding of musical audio. In: IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, 2005., pp. 239–242. IEEE (2005). DOI 10.1109/ASPAA.2005.1540214
36. Viste, H., Evangelista, G.: Sound source separation: Preprocessing for hearing aids and structured audio coding. In: COST G-6 Conference on Digital Audio Effects (DAFX-01), pp. 67–70 (2001)
37. Woodruff, J., Pardo, B., Dannenberg, R.: Remixing Stereo Music with Score-Informed Source Separation. Proceedings of the 7th International Society for Music Information Retrieval Conference (ISMIR) (2006)