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

Real-time Soundprism

A.J. Muñoz-Montoro · J. Ranilla · P. Vera-Candeas · E.F. Combarro · P. Alonso-Jordá

Received: date / Accepted: date

Abstract This paper presents a parallel real-time sound source separation system for decomposing an audio signal captured with a single microphone in so many audio signals as the number of instruments that are really playing. This approach is usually known as *Soundprism*. The application scenario of the system is for a concert hall in which users, instead of listening to the mixed audio, want to receive the audio of just an instrument, focusing on a particular performance. The challenge is even greater since we are interested in a real-time system on handheld devices, i.e. devices characterized by both low power consumption and mobility. The results presented show that it is possible to obtain real-time results in the tested scenarios using an ARM processor aided by a GPU, when this one is present.

Keywords Sound source separation · Real-time · Score alignment · Audio processing · Parallel computing · GPGPU

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

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

P. Vera-Candeas
Department of Telecommunication Engineering, Universidad de Jaén, Spain
E-mail: pvera@ujaen.es

E.F. Combarro
Department of Computer Science, Universidad de Oviedo, Spain
E-mail: efernandezca@uniovi.es

P. Alonso-Jordá
Department of Information Systems and Computation, Universitat Politècnica de València, Spain
E-mail: palonso@upv.es

1 Introduction

Source separation (SS) of musical signals deals with the task of segregating the original sound signals from a polyphonic mixture. Its application can be used to ameliorate the way of consuming the musical entertainment. This field of research has been a trending topic over the last two decades and now offers a wide variety of possible applications for end-users and professionals, such as instrument-wise equalization [16], personal music remixing [15], music information retrieval [6], intelligent audio editing [9], etc. Depending on the final application, there are two approaches for this problem, which are called *offline* or *online*. In offline algorithms, the audio performance to be processed is available as a whole. Thus, these algorithms require to see the whole audio performance from start to end, and hence cannot be implemented in real time. On the other hand, for streaming scenarios, an online system has to process its input piece-by-piece in a serial fashion, in the order that the input (the audio stream) is fed to the algorithm, without having the entire input available from the start. Moreover, a low latency approach is required for a real-time behavior, understanding latency as the delay between receiving the signal and starting to perform the separation.

SS problems can be classified depending on the number of sources and sensors. *Over-determined* and *determined* cases are those in which the number of sensors is larger than or equal to the number of sources, respectively. In theses cases, several methods [19][25][27] have achieved good results. However, the single-channel SS problem is the extreme example of the *under-determined* cases, in which sensors are fewer than sources. Therefore, a high SS quality is not reachable by blind SS methods [26]. In that sense, it is necessary to exploit the knowledge of additional information to improve the separation [10][27]. This information can be of different natures: the instrumentation [17], score information [18], spatial and spectral information about the sources [20], information about the recording/mixing conditions [8], etc. Considering the case where a musical score (in the form of MIDI) is available [15][6][18], information about the instruments and notes of the musical piece can be used to guide the separation process even if the sources are hard to distinguish based on their spectrotemporal behavior.

In this context, extant score-informed SS systems need a well alignment between the score and the audio [22][15], or use dynamic time warping (DTW) to find the alignment [22][12] before separating sources. Duan et al. [7] proposed an online score-informed SS method, called *Soundprism*. Similar to an optical prism which breaks light up into its constituent spectral colors in real-time, *Soundprism* is a computer algorithm able to decompose a mixture of music on its constituent sound sources in real-time. Duan et al. address this problem dividing it in two stages: 1) audio-score alignment and 2) pitch-informed source separation. The alignment is computed using a hidden Markov process model, where each audio frame is associated with a 2-D state of score position and tempo. A multi-pitch observation model is employed for indicating how likely each audio frame contains the pitches of a hypothesized score position.

Afterwards, particle filtering is used to infer the score position and tempo of each frame. Regarding the separation process, score-informed pitches at the aligned score position are refined with a multi-pitch estimation algorithm to build a harmonic mask. Nevertheless, due to the high computational cost of this system, its real-time implementation is not possible with the current technology.

In this paper, we propose an online SS framework for single-channel audio signals of high polyphonic complexity that yields a server-client model application. This framework is the solution for a classical music concert where every audience member could select their favorite personal mix, switching between enjoying the full performance or concentrating on a specifical instrument. In this context, the server carries out the factorization of the audio input signal and offers to the clients all the isolated signals of each instrument. For this purpose, we decompose this problem in three stages: 1) the real-time audio-to-score alignment, 2) the factorization process and 3) the source reconstruction stage. The first stage is based on our system ReMAS [1]. ReMAS is a parallel and efficient Real-time Musical Alignment System that has been implemented and optimized for low-power processors, such as ARM processors. For the second stage, the library NNMFPack [4][5] for solving the NMF problem is used. NNMFPack is an efficient numerical library conceived for heterogeneous parallel shared memory systems. In the final stage, we propose to apply a Wiener filtering method to compute the energy contribution of each instrument of the audio mixture. With the aim of closing the whole process in real-time, a parallel implementation of this final stage is proposed, according to the methodology followed in ReMAS and NNMFPack. In the end, the whole system is a parallel hybrid solution using the CPU and the GPU simultaneously.

To the best of our knowledge, there has not yet been presented a holistic, free and cross-platform system that addresses these problems in real-time running on systems-on-chips (SoC) with ARM architecture, the most common in devices such as tablets or mobile phones, among others. As a proof of concept, some experiments are carried out on a dataset of orchestral music, showing reliable results in terms of sound quality. Accordingly, the structure of the paper is organized as follows. In Section 2, we describe the architecture of the proposed framework whilst the evaluation datasets and the experimental results are shown in Section 3. Finally, conclusions are outlined in Section 4.

2 Framework description and background

2.1 System overview

The aim of this work is to address the SS problem for high polyphonic complexity signals through the development of an automated computer *Soundprism* system. For this proposal, we have decomposed the problem into three stages: the alignment between the input MIDI score and the input audio performance, the factorization of this real audio mixture and the reconstruction of each in-

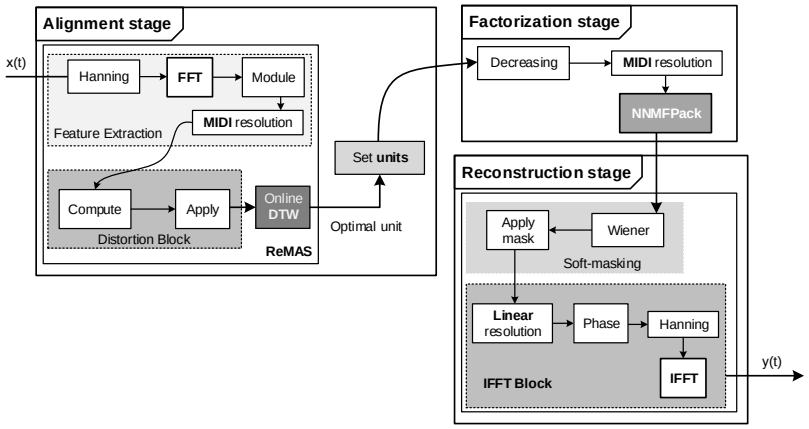

Fig. 1 Block diagram of the proposed framework.

strument signal. Fig. 1 sketches the full system with the main blocks which compound each stage.

The software solution proposed has been developed satisfying the following two basic requirements: real-time and mobility. Therefore, this design should take the low computational power of the handheld devices into consideration, specially the cheapest ones, and should deeply use the possibilities offered by some heterogeneous parallel architectures.

2.2 Alignment stage

The goal of this stage is to associate each temporal frame of the input audio signal to an event in the MIDI score. This matching will be used as input by the following stage. In that sense, we employ our algorithm ReMAS [1]. As can be observed in Fig. 1, when a new frame arrives, $x(t)$, the process starts at the feature extraction block. This block extracts from the audio signal the features that characterize some specific information about the musical content. With this intention, a low-level spectral representation of the audio data (time-frequency representation) is obtained by a Hanning-windowed fast Fourier transform (FFT). Performed the FFT, the time-frequency representation is converted from linear frequency to MIDI resolution. To obtain this computation, first the magnitude spectrogram (inset Module in Fig. 1) is computed from the complex output of the FFT. Afterwards, the frequency bins belonging to the same MIDI interval are summed up (inside MIDI Resolution in Fig. 1).

Later, the distortion block is performed. It measures the similarity between the time-frequency representation of each frame of the input signal and the

spectral patterns of each score unit, defining a score unit as the occurrence of concurrent or isolated notes in the score (described in [24]).

Finally, DTW is used to associate the score position with each input signal frame. In this step, no backtracking is allowed in order to reduce latency. In that way, the decision is made directly from the information contained in frame t, taking the minimum value of the accumulated cost. A deep description of the theoretical aspects of the problem and the design of the multi-tier architecture for the DTW functional block can be found in [3][24]. Our parallel implementation of this algorithm, presented in [1], has been used here including some optimizations carried out to avoid the downside of a reduction operation in parallel.

The output of ReMAS system at each audio frame is an event in the MIDI score. This event is used to inform of the notes and instruments that are sounding in that frame to the factorization stage. However, the alignment algorithm, in which ReMAS is based on, commits misalignment errors [24], so much so that the minimum onset deviation threshold required to obtain a full precision is about $2s$. In order to avoid misalignment errors, the final block of the alignment process (inside Set Units in Fig. 1) selects the score events which are adjacent in the score to the selected one by the ReMAS output. In that way, this block looks up in MIDI score all the units which appear 1 s (referred to MIDI time) before and after the MIDI event selected by the DTW. Thus, the factorization stage receives, as an input, a collection of units ϕ_t, which correspond to the possible concurrent notes and instruments for each real audio frame.

2.3 Factorization stage

The factorization stage is focused on the estimation of the spectrograms of each source. The main idea is that an audio spectrogram can be decomposed as a linear combination of spectral basis functions $b_n(f)$. In such a model, the magnitude spectrogram of the signal $x(f,t)$ in time-frame t and frequency f is modeled as a weighted sum of basis functions as:

$$x(f,t) \approx \hat{x}(f,t) = \sum_{n=1}^{N} b_n(f) g_n(t) \ , \tag{1}$$

where $g_n(t)$ is the gain of the basis function n at frame t, $b_n(f)$ are the bases and N is the total number of units in the composition.

Applied to our problem, each basis function represents the spectral pattern of each score unit and the corresponding gains contain information about the onset and offset times of these units. The spectral patterns $b_n(f)$ are obtained in the preprocessing stage for each score unit.

The computational intensity of the estimation of the parameter $g_n(t)$ is extremely large due to its dependency on the number of score units. Consequently, the higher the number of units of the score, the larger the computational intensity. In orchestra signals, several instruments playing different

melodies are present, and that means that the score is compound of a large number of units. However, as explained in Section 2.2, only a few units are selected by the alignment stage at each frame. Denoting by ϕ_t the subset of units that can be potentially active at frame t, the signal model can be expressed as follows:

$$x(f,t) \approx \hat{x}(f,t) = \sum_{n \in \phi_t} b_n(f) g_n(t) \ , \tag{2}$$

where the number M of units of ϕ_t is much lower than N.

The Decreasing block (Fig. 1) reads the bases indicated by the subset ϕ_t from the dictionary of bases, and arranges them in a block of memory which is employed as input for the factorization process. Without this block, it would not be possible to compute the factorization process in real-time, due to the high computational cost of using the whole dictionary.

Once the Decreasing block is executed, the time-frequency representation of the input signal is converted to a resolution of 1/4 of a semitone, which has been proven to achieve better results for separation tasks [2]. Afterwards, the factorization of the signal is carried out using our numerical library, NNMF-Pack [4][5], that provides efficient algorithms to compute the NMF.

Under the nonnegativity restriction, the factorization parameters of Eq. 2 can be estimated by minimizing the β-divergence between the observed $x(f,t)$ and the modeled $\hat{x}(f,t)$ spectrograms. We have used some subroutines of the library that implement the following multiplicative update rules required for the factorization:

$$g_n(t) \leftarrow g_n(t) \odot \left(\frac{\sum_{f,n} b_n(f)[\hat{x}(f,t)^{\beta-2} \odot x(f,t)]}{\sum_{f,n} b_n(f) \hat{x}(f,t)^{\beta-1}} \right). \tag{3}$$

where $\beta = 1.5$.

Finally, given the time-varying amplitudes of each unit $g_n(t)$ at each frame, the spectrogram of each source $\hat{x}_s(f,t)$ can be computed knowing the concurrent and isolated instruments for each unit.

2.4 Source reconstruction stage

In this final stage, the reconstruction of each source is computed using a soft-filter strategy. Fig. 1 outlines the main blocks which compound this stage. Firstly, the spectrogram of each source is estimated by:

$$x_s(f,t) = \sum_{n \in \phi_t} b_{n,s}(f) g_n(t) \ , \tag{4}$$

where $g_n(t)$ is the time-varying amplitude acquired in the factorization stage and $b_{n,s}(f)$ is the spectral patterns obtained in the preprocessing stage where the amplitudes of each note of a musical instrument are learned in advance by

using the Real World Computing (RWC) music database [13][14] as a training database of solo instruments playing isolated notes.

Subsequently, Wiener filter is applied 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 $M_s(f,t)$ is computed for each source:

$$M_s(f,t) = \frac{|x_s(f,t)|^2}{\sum_s |x_s(f,t)|^2} \ , \tag{5}$$

where $M_s(f,t)$ represents the relative energy contribution of each source s for each time-frequency bin and $x_s(f,t)$ is the magnitude spectrogram per instrument. Then, to obtain the estimated source magnitude spectrogram $\hat{x}_s(f,t)$, Eq. 5 is used (inside Apply Mask in Fig. 1).

$$\hat{x}_s(f,t) = \sqrt{M_s(f,t)} \cdot x(f,t) \ . \tag{6}$$

Afterwards, the IFFT block is performed. The time representation is converted from MIDI resolution to linear frequency allocating the value of a MIDI bin to its corresponding frequency bins (inside Linear Resolution in Fig. 1). The phase spectrogram of the input mixture is applied for each source (inside Phase in Fig. 1). Finally, 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_n(f) from the preprocessing stage.
while audio stream do
    Read audio frame x(f,t).
    Compute ReMAS to obtain the optimal unit.
    Determine the subset φ_t.
    Arrange the block of memory for the factorization process.
    Change to a quarter of a semitone MIDI resolution.
    Update the gains using the Eq. 3.
    for s=1 to S do
        Compute Wiener mask using the Eq. 5.
        Estimate the spectrogram of each source x_s(f,t) using Eq. 6.
        Change to linear resolution.
        Compute the IFFT.
        Play x_s(t).
    end for
end while
```

From the computational point of view, several decisions have been made to successfully address this stage regarding the way of storing the data. Consequently, it is stored using vectors whose leading dimension have been arranged according to the subsequent use. In this manner, we can exploit spatial locality and use the high performance BLAS packages. Thus, Wiener filter is implemented using vector-vector and matrix-vector routines. Taking into account

these considerations and observing Eq. 5 and Eq. 6, the temporal complexity of this stage at each frame can be approximated by:

$$S \cdot (P_m F_{mf} + \frac{F \cdot log_2(F)}{P}) \tag{7}$$

where P_m is the number of pitches (MIDI notes) per instrument, S is the number of sources (instruments), F is the fast Fourier transform length, P is the number of cores and F_{mf} is the frequency in MIDI resolution sampled for the factorization stage. As in [23], we have selected $P_m = 456$ that corresponds to a range of notes of a 9.5-octaves in one sample per quarter of a semitone of MIDI resolution. F_{mf} is fixed as 401, one sample per quarter of a semitone. On the other hand, S depends on the composition and is obtained from the reading of the MIDI score. Finally, we have used $F = 16,384$ bins as in [23], since this value is chosen to have enough frequency resolution for low frequency sounds.

2.5 Hybrid GPU-CPU implementation

From the computational complexity point of view, source reconstruction and ReMAS stages have a high impact on the system. On the one hand, compositions with a high number of instruments increase the parameter S (see Eq. 7) involving a high computational cost in the reconstruction stage, regardless of the duration of the composition. On the other hand, compositions with a high duration lead to an increase in the execution time of alignment stage because the number of score units and the MIDI time frames grow (see [1]). There will be situations where the combination of the values of these parameters will make the execution time of both stages the same. In others, a pure CPU system (homogeneous) could not be executed in real-time due to the high duration and large number of instruments.

Looking back to Fig. 1, once the audio frame has been captured, the factorization stage begins when the alignment stage (based on ReMAS) has associated the audio frame to a MIDI score point. This MIDI score point is represented by a scalar. In addition, during the execution of ReMAS, the FFT of the audio is computed and stored; this information, obtained at the beginning of ReMAS, is necessary to carry out the reconstruction of each instrument. Finally, the reconstruction stage generates an output audio frame for each source, which must be sent to the output sound card.

In [11], Fastl et al. studied the time difference perceived by the human hearing system between the directly transmitted audio signals and the processed audio signals and a critical value of 50 ms for speech and music applications was fixed. This fact allows us to assume a maximum response time of 50 ms for our system.

Considering all these aspects, we can divide the logic of *Soundprism* into two main blocks running concurrently: ReMAS (alignment stage) and NMF+Re (factorization and reconstruction stages). Thus, while one functional unit (CPU

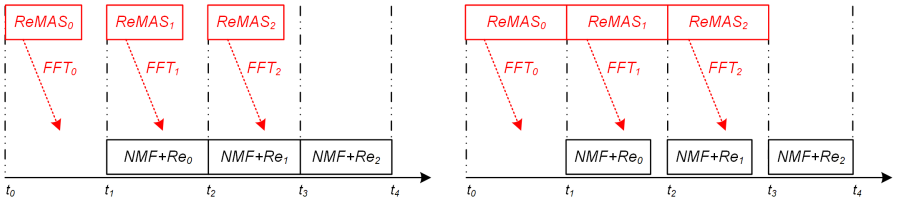

Fig. 2 Hybrid GPU-CPU parallelization scheme. Left when the NMF+Re stage is the longest. Right when ReMAS is the longest.

or GPU) executes ReMAS over frame i, the other applies NMF+Re to the frame $i-1$ (see Fig. 2). Remember that the FFT information needed by NMF+Re is ready to send at the beginning of aligned stage, and thereby, *Soundprism* can overlap computing and communications hiding the impact of this kind of communications over time. However, note that ReMAS over frame i has to finish to start NMF+Re over the same frame. As mentioned, NMF+Re generates the audio to send to the sound card and, to avoid additional communications, the best choice is to assign ReMAS to the GPU and NMF+Re to the CPU. In this manner, we can achieve a real-time behaviour with high duration compositions and with a large number of instruments. Thus, for example, when the execution time of ReMAS and NMF+Re is approximately the same, the theoretical overall execution time will be reduced by half, compared to the parallel implementation using only the CPU.

3 Evaluation and experimental results

For the experimentation of our proposed framework, we have employed a database developed by Pätynen et al. [21]. The database consists of four compositions of symphonic music from Classical and Romantic styles. The audio files are approximately 3 minutes long and are sampled at 44.1 KHz from real performances. The passages are composed of ten different kind of sources (instruments). Furthermore, we have analyzed the performance of the proposal with two more musical pieces of different duration and number of sources: *"American Quartet"*, a string quartet by Antonín Dvořák of about 26 minutes of duration and *"Finland"*, a symphonic poem written by Jean Sibelius of about 8 minutes. Table 1 provides more details of the database, including duration, the number of instruments that make up the performance, and the number of units extracted from the MIDI file. As shown, Sibelius has the highest number of sources (16 instruments), and Dvořák the lowest.

Regarding the used testbed, this is an Nvidia Jetson TX2 development kit, which is an embedded system-on-module (SoM) with a NVIDIA Pascal GPU 256 cores and one quad-core ARM Cortex-A57 CPU. It operates at 2 GHz and runs a version of Linux operating system specially tailored to this device. This kind of architecture is the heart of smartphones, laptops, tablets, and other embedded systems.

Composer	Piece name	Dur.	Instruments	Score units
Beethoven	Symphony no. 7	3m 11s	10	908
Bruckner	Symphony no. 8	1m 27s	10	521
Mahler	Symphony no. 1	2m 12s	10	717
Mozart	Don Giovanni	3m 47s	8	857
Dvořák	American Quartet	26m 43s	4	3392
Sibelius	Finland	8m 33s	16	1549

Table 1 Characteristics of the orchestral dataset used for the evaluation of our source separation system.

3.1 Experimental results

In this section, we are going to analyze the experimental results obtained in the evaluation of the proposed database. Table 2 shows the execution time of ReMAS and NMF+Re blocks for each implementation and audio excerpt. In addition, speedup and efficiency results are included.

3.1.1 CPU results

The computational complexity of the system depends mainly on the duration of the score and the number of instruments of the composition. In this sense, attending firstly to the duration of the score, the alignment stage (ReMAS in Table 2) is about the 54% of the total execution time of the system for the Dvořák composition independently of the number of cores. However, for the remaining scores, the alignment stage time represents just between 10% and 19%. As expected, the complexity burden per frame of the alignment block increases due to score duration. CPU implementation guarantees real-time with 4 cores for: (1) long compositions (tested until 1603 seconds) with a reduced number of instruments (4 instruments in Dvořák score), and (2) short compositions (tested until 512 seconds) in symphonies. Focusing on 3 cores, the only audio excerpt that is not possible to execute in real-time is Sibelius's, which combines a relatively high duration (512 seconds) and a large number of instruments (16 instruments). However, this limitation can be sorted out just by reducing the FFT length to $F = 8,192$. Changing this parameter only affects the quality of low pitch sounds in separation, giving the system the opportunity to be executed in real-time. Using less cores, we conclude that the execution time can not be considered real-time.

Secondly, regarding the number of instruments, we have tested the system with a variety of symphonies of different composers (Beethoven, Mozart, Mahler, Bruckner and Sibelius) all with a number of groups of instruments higher than 8. These examples are the more demanding ones in terms of instruments in the case of classical music scores. For all orchestra scores, the NMF+Re block increases its execution times with respect to ReMAS, consuming between 80% and 90% of the total execution time of the system in comparison with a range from 40% to 50% in Dvořák.

At source reconstruction stage, some of the blocks require to repeat some operations depending on the number of score instruments, such as wiener Mask estimation and the IFFT algorithm. The approach used in this work is based on separating all the group of instruments at the score, just to be fair with the analogy of the name *Soundprism*. Consequently, the system is designed to perform the separation of audio input in all its components (instruments in the audio case). However, the system proposed in [7] only implements the separation of a single instrument, and even so, it cannot be executed in real-time.

Finally, the behavior obtained by the system is as expected regarding the efficiency. Better results are obtained with long compositions, due to the alignment stage efficiency increases with higher score duration. For the NMF+Re block, we obtain a global efficiency higher than 60% for the worst-case scenario. Note that, for the Wiener filter, a set of matrix-vector routines (memory bound) is computed. Therefore, the sequential approach maximizes the performance, taking advantage of the whole memory bandwidth, while a/the parallel approach is limited by this fact. At the same time, the NNMFPack library also has a memory bound component and, furthermore, the optimal performance is obtained with higher matrix dimensions (see [5]). Fig. 3 displays the efficiency results for each audio excerpt and number of cores.

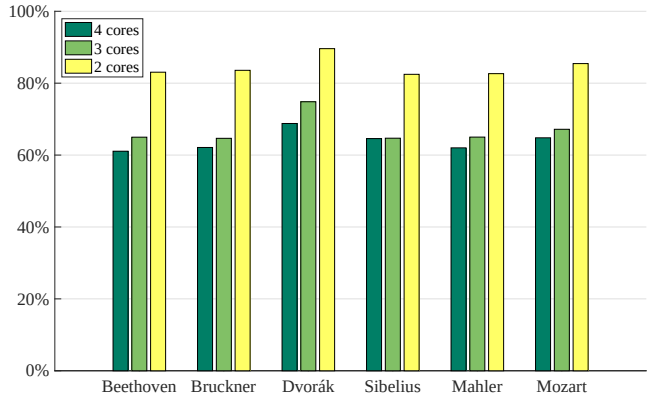

Fig. 3 Efficiency of the CPU implementation depending on the number of cores for each composition.

3.1.2 Hybrid parallel scheme results

The goal of our proposal is to provide a parallel real-time sound source separation system for decomposing an audio signal when devices characterized by both low power consumption and mobility are used. As discussed in the previous subsection, the pure CPU system (homogeneous) achieves a real-time

behaviour for all signals of the dataset when all available cores are used. However, in exceptional cases in which the number of instruments is very high and the duration of the composition is very long, and/or there is even an additional load in the system, it is possible that the response time is not within the real-time window.

The hybrid GPU-CPU proposal resolves precisely these more demanding situations. Looking at the numerical results in Table 2, the behavior of this implementation is as expected. On the one hand, the execution time of the NMF+Re block is equal to the pure CPU implementation when all the cores are used. This fact allows to infer that if the number of cores used was lower, the execution time obtained would be approximately equal to that obtained with the CPU version, allowing to free resources for other tasks in the system and providing greater stability in the response time of *Soundprism*. On the other hand, the ReMAS time is substantially reduced. This is especially relevant in long compositions, where the speedup over the CPU version using all the cores is about 50% (see Fig. 4). Regarding less demanding compositions, the speedup is above 10%. Then, as can be observed in Table 2, the overall execution time of this version presents a fixed overload of low impact lower than 2.5% with respect to the theoretical value, which would be the maximum time between ReMAS and NMF+Re. This value indicates a high concurrence level and a perfect masking of the communications between both subsystems (from GPU to CPU), an objective pursued.

Finally, from the point of view of the usefulness and/or scalability of the system, it should be remarked that the execution time in long compositions (see Dvořák in Table 2) is approximately 30% of the duration, reaching response times less than 5 milliseconds.

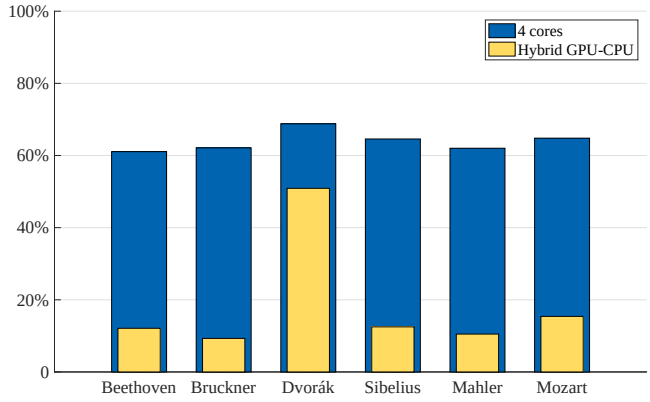

Fig. 4 Overlapping of the speedup of the hybrid proposal and the efficiency of the pure CPU implementation.

Execution Time																
Implementation		Hybrid GPU-CPU			4 cores			3 cores			2 cores			1 core		
Composition	Duration (s)	ReMAS (s)	NMF+Re (s)	Total (s)	ReMAS (s)	NMF+Re (s)	Total (s)	ReMAS (s)	NMF+Re (s)	Total (s)	ReMAS (s)	NMF+Re (s)	Total (s)	ReMAS (s)	NMF+Re (s)	Total (s)
Beethoven	191	10.42	118.04	120.95	19.01	118.04	137.05	23.40	148.46	171.86	29.35	172.40	201.75	53.53	281.99	335.52
Bruckner	87	3.70	53.61	54.33	6.01	53.61	59.62	7.74	68.69	76.43	9.49	79.27	88.76	16.72	131.80	148.52
Dvorak	1603	203.75	502.75	515.55	543.04	502.75	1045.79	653.65	629.31	1282.96	880.67	727.49	1608.16	1678.72	1206.45	2885.17
Sibelius	512	36.02	433.15	444.85	73.83	433.15	506.98	91.40	583.95	675.35	123.26	671.81	795.07	222.03	1090.05	1312.08
Mahler	132	6.28	80.20	82.00	11.05	80.20	91.25	13.80	102.34	116.14	17.22	119.84	137.06	31.04	195.82	226.86
Mozart	227	12.45	113.47	115.85	22.98	113.47	136.45	29.11	146.57	175.68	35.92	171.32	207.24	65.88	288.74	354.62

Execution Time per Frame																
Implementation		Hybrid GPU-CPU			4 cores			3 cores			2 cores			1 core		
Composition	Duration (s)	ReMAS (ms)	NMF+Re (ms)	Total (ms)	ReMAS (ms)	NMF+Re (ms)	Total (ms)	ReMAS (ms)	NMF+Re (ms)	Total (ms)	ReMAS (ms)	NMF+Re (ms)	Total (ms)	ReMAS (ms)	NMF+Re (ms)	Total (ms)
Beethoven	191			8.19	1.29	7.99	9.28	1.58	10.05	11.63	1.99	11.67	13.66	3.62	19.09	22.71
Bruckner	87			8.08	0.89	7.97	8.87	1.15	10.22	11.37	1.41	11.79	13.20	2.49	19.60	22.09
Dvorak	1603			4.16	4.38	4.05	8.43	5.27	5.07	10.34	7.10	5.87	12.97	13.54	9.73	23.26
Sibelius	512			11.22	1.86	10.92	12.79	2.30	14.73	17.03	3.11	16.94	20.05	5.60	27.49	33.09
Mahler	132			8.03	1.08	7.86	8.94	1.35	10.02	11.38	1.69	11.74	13.43	3.04	19.18	22.22
Mozart	227			6.60	1.31	6.46	7.77	1.66	8.35	10.00	2.05	9.76	11.80	3.75	16.44	20.19

Speedup vs. 4 cores					Efficiency								
Implementation		Hybrid GPU-CPU			4 cores			3 cores			2 cores		
Composition	Duration (s)	ReMAS	NMF+Re	Total	ReMAS	NMF+Re	Total	ReMAS	NMF+Re	Total	ReMAS	NMF+Re	Total
Beethoven	191			12.1%	70.4%	59.7%	61.2%	76.3%	63.3%	65.1%	91.2%	81.8%	83.2%
Bruckner	87			9.3%	69.6%	61.5%	62.3%	72.0%	64.0%	64.8%	88.1%	83.1%	83.7%
Dvorak	1603			50.9%	77.3%	60.0%	69.0%	85.6%	63.9%	75.0%	95.3%	82.9%	89.7%
Sibelius	512			12.5%	75.2%	62.9%	64.7%	81.0%	62.2%	64.8%	90.1%	81.1%	82.5%
Mahler	132			10.5%	70.2%	61.0%	62.2%	75.0%	63.8%	65.1%	90.1%	81.7%	82.8%
Mozart	227			15.4%	71.7%	63.6%	65.0%	75.4%	65.7%	67.3%	91.7%	84.3%	85.6%

Table 2 Execution times measured in seconds and in milliseconds per frame, speedup and efficiency for each implementation, block and audio excerpt.

4 Conclusions and future work

First, we would like to stress that the proposed system is, to the best of our knowledge, the first implementation in real-time for the *Soundprism* paradigm. Our approach has focused on achieving real-time execution using handheld devices characterized by both low power consumption and mobility. Under these limitations, the proposed system is real-time for all the audio excerpts in the case of an execution in the CPU implementation with 4 cores and in the hybrid parallel proposal. Secondly, it has been shown that the length of the score affects moderately the total complexity of the system. However, the number of score instruments has a strong impact, because it requires the computation of the same number of Wiener masks and IFFT blocks as there are instruments.

Finally, for future work, we plan to extend the current approach to be able to manage with multi-channel audio input. To achieve this goal, the concert hall should include a group of microphones distributed along the room. Using this proposal, the quality of the separation will be greatly increased because the system will deliver to the user the signal received for the microphone closer to a particular instrument.

Acknowledgements This work has been supported by the "Ministerio de Economía y Competitividad" of Spain and FEDER under projects TEC2015-67387-C4-{1,2,3}-R.

References

1. Alonso P, Cortina R, Rodríguez-Serrano FJ, Vera-Candeas P, Alonso-González M, Ranilla J (2017) Parallel online time warping for real-time audio-to-score alignment in multi-core systems. J Supercomput 73: 126. https://doi.org/10.1007/s11227-016-1647-5
2. 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: 184. https://doi.org/10.1186/1687-6180-2013-184
3. Carabias-Orti JJ, Rodriguez-Serrano FJ, Vera-Candeas P, Canadas-Quesada FJ, Ruiz-Reyes N (2015) An audio to score alignment framework using spectral factorization and dynamic time warping. In: 16th International Society for music information retrieval conference 742–748
4. Díaz-Gracia N, Cocaña-Fernández A, Alonso-González M, Martínez-Zaldívar FJ, Cortina R, García-Mollá VM, Alonso P, Ranilla J (2014) NNMFPACK: a versatile approach to an NNMF parallel library. Proc 2014 Int Conf Comput Math Methods Sci Eng 456–465
5. Díaz-Gracia N, Cocaña-Fernández A, Alonso-González M, Martínez-Zaldívar FJ, Cortina R, García-Mollá VM, Vidal AM (2015) Improving NNMFPACK with heterogeneous and efficient kernels for β-divergence metrics. J Supercomput 71: 1846–1856. https://doi.org/10.1007/s11227-014-1363-y
6. Driedger J, Grohganz H, Prätzlich T, Ewert S, Müller M (2013) Score-informed audio decomposition and applications. In: Proceedings of the 21st ACM international conference on Multimedia pp. 541-544
7. Duan Z, Pardo B (2011) Soundprism: an online system for score-informed source separation of music audio. IEEE J. Sel. Top. Signal Process. 5(6):1205–1215

8. Duong NQ, Vincent E, Gribonval R (2010) Under-determined reverberant audio source separation using a full-rank spatial covariance model. IEEE T Audio Speech 18(7): 1830-1840. https://doi.org/10.1109/TASL.2010.2050716
9. Ewert S, Müller M (2011) Estimating note intensities in music recordings. In: Int Conf Acoust Spee pp. 385-388
10. Ewert S, Pardo B, Mueller M, Plumbley MD (2014) Score-Informed Source Separation for Musical Audio Recordings: An overview. IEEE Signal Process Mag 31:116–124. https://doi.org/10.1109/MSP.2013.2296076
11. Fastl H, Zwicker E (2007) Psychoacoustics. Springer Berlin Heidelberg, Berlin, Heidelberg
12. Ganseman J, Scheunders P, Mysore GJ, Abel JS (2010) Source separation by score synthesis. Int Comput Music Conf 2010 1–4
13. Goto M, Hashiguchi H, Nishimura T, Oka R (2002) RWC Music Database: Popular, Classical and Jazz Music Databases. Ismir 2:287–288
14. Goto M (2004) Development of the RWC music database. Proc 18th Int Congr Acoust (ICA 2004) 553–556
15. Hennequin R, David B, Badeau R (2011) Score informed audio source separation using a parametric model of non-negative spectrogram. Acoust Speech Signal Process (ICASSP), 2011 IEEE Int Conf 45–48. https://doi.org/10.1109/ICASSP.2011.5946324
16. Itoyama K, Goto M, Komatani K, et al (2008) Instrument Equalizer for Query-by-Example Retrieval: Improving Sound Source Separation Based on Integrated Harmonic and Inharmonic Models. Ismir. https://doi.org/10.1136/bmj.324.7341.827
17. Marxer R, Janer J, Bonada J (2012) Low-latency instrument separation in polyphonic audio using timbre models. In: International Conference on Latent Variable Analysis and Signal Separation 314-321
18. Miron M, Carabias-Orti JJ, Janer J (2015) Improving score-informed source separation for classical music through note refinement. ISMIR 448-454
19. Ozerov A, Févotte C (2010) Multichannel nonnegative matrix factorization in convolutive mixtures for audio source separation. IEEE Trans Audio, Speech Lang Process. https://doi.org/10.1109/TASL.2009.2031510
20. Ozerov A, Vincent E, Bimbot F (2012) A general flexible framework for the handling of prior information in audio source separation. IEEE Trans Audio, Speech Lang Process. https://doi.org/10.1109/TASL.2011.2172425
21. Pätynen J, Pulkki V, Lokki T (2008) Anechoic Recording System for Symphony Orchestra. Acta Acust united with Acust 94:856–865. https://doi.org/10.3813/AAA.918104
22. Raphael C (2008) A Classifier-Based Approach to Score-Guided Source Separation of Musical Audio. Comput Music J 32:51–59. https://doi.org/10.1162/comj.2008.32.1.51
23. 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 44:83–96. https://doi.org/10.1080/09298215.2014.989174
24. Rodriguez-Serrano FJ, Carabias-Orti JJ, Vera-Candeas P, Martinez-Munoz D (2016) Tempo Driven Audio-to-Score Alignment Using Spectral Decomposition and Online Dynamic Time Warping. ACM Trans Intell Syst Technol 8:1–20. https://doi.org/10.1145/2926717
25. Sawada H, Araki S, Makino S (2011) Underdetermined convolutive blind source separation via frequency bin-wise clustering and permutation alignment. IEEE Trans Audio, Speech Lang Process 19(3): 516-527. https://doi.org/10.1109/TASL.2010.2051355
26. Vincent E, Araki S, Theis F, et al (2012) The signal separation evaluation campaign (2007–2010): Achievements and remaining challenges. Signal Processing 92:1928–1936. https://doi.org/10.1016/j.sigpro.2011.10.007
27. Vincent E, Bertin N, Gribonval R, Bimbot F (2014) From blind to guided audio source separation: How models and side information can improve the separation of sound. IEEE Signal Process Mag 31:107–115. https://doi.org/10.1109/MSP.2013.2297440