HReMAS: Hybrid Real-time Musical Alignment System

P. Cabañas-Molero · Raquel Cortina · E. F. Combarro · Pedro Alonso · F.J. Bris-Peñalver

Abstract This paper presents a real-time audio-to-score alignment system for musical applications. The aim of these systems is to synchronize a live musical performance with its symbolic representation in a music sheet. We have used as a base our previous real-time alignment system by enhancing it with a *traceback stage*, a stage used in offline alignment to improve the accuracy of the aligned note. This stage introduces some delay, what forces to assume a trade-off between output delay and alignment accuracy that must be considered in the design of this type of hybrid techniques. We have also improved our former system to execute faster in order to minimize this delay. Other interesting improvements, like identification of silence frames, have also been incorporated to our proposed system.

Keywords hybrid audio-to-score alignment · Audio-to-score alignment · Score following · Dynamic Time Warping

1 Introduction

Online audio-to-score alignment (also known as *score following*) is the task of synchronizing a live performance of music with its corresponding symbolic score in real-time [7]. Essentially, a score follower is a software that "listens" to the audio signal and determines almost instantly the musicians' position in the sheet (represented internally in MIDI, music XML or any other structured format).

P. Cabañas-Molero · F.J. Bris-Peñalver
Department of Telecommunication Engineering,
University of Jaén, Linares, Jaén, Spain
E-mail: {pcabanas,fbris}@ujaen.es

Raquel Cortina · E.F. Combarro
Depto. de Informática
Universidad de Oviedo, Spain
E-mail: {raquel,efernandezca}@uniovi.es

Pedro Alonso
Depto. de Sistemas Informáticos y Computación
Universitat Politècnica de València, Spain
E-mail: palonso@upv.es

Unlike offline alignment, where the whole performance is available before starting the synchronization, online algorithms take a decision as the audio is acquired, using only past and present information. This fact makes online alignment less accurate than offline alignment. However, online alignment is required for some practical real-time applications where a slight delay on the output can be admitted as, for example, in automatic page turning [4], automatic music accompaniment for live soloists [20], real-time sound source separation [11], interactive educational music software [8], display of synchronized context information [3] or automatic control of equipment on stage (e.g. lighting, screens or cameras). In this context, many recent efforts in the field have been focused on improving the robustness and speed of music real-time tracking systems, making them appropriate for mobile devices and for a wide range of real-life contexts [1,2,3].

In score following, in general, there exists a trade-off between output delay and alignment accuracy. By aligning the current frame, greater accuracy can be achieved if neighboring frames (including future ones) are available for processing, at the expense of both introducing some latency and using more resources. For a given score following algorithm it is worth studying this trade-off since a significant gain in accuracy can be obtained by sacrificing a little response speed. Provided latency (and complexity) is kept within acceptable limits, robustness is more important than response time for some of the applications. This is the case, for instance, of an application that receives the audio signal and displays the music sheet with synchronized highlighting of the current position (with automatic page turning or scrolling). It does not matter too much if the position is estimated with a short delay, but it is crucial not to deviate too much during the performance. This trade-off is useful in strictly online tracking of classical music, where pieces are long and full of different situations (slow parts, long pauses, fast sections, high level of polyphony). Similarly, this is also the case of applications based on score-guided source separation, where the key aspect is the quality in separation, even if the output audio is played with a certain delay.

In this paper we analyze the relationship between alignment accuracy and delay. We base this analysis on the framework of our system ReMAS [2]. 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, which are the heart of smartphones, laptops, tablets, and other embedded systems. ReMAS uses a fast spectral decomposition algorithm to measure the matching between each input frame and all events in the score [5]. It also uses a variation of online Dynamic Time Warping (DTW) to perform the alignment, where only the forward path is computed and the result is returned without delay. The system presented in this paper, HReMAS (Hybrid Real-time Musical Alignment System), improves ReMAS by incorporating a *traceback stage*, borrowed from the offline alignment, to the online DTW with a configurable delay parameter that produces a "hybrid" alignment. Consequently, the alignment resulted for each frame is returned after a certain delay by following the backward path started at the point estimated for the current input frame.

HReMAS also includes other important improvements such as, e.g. an algorithm to detect silence audio frames. We propose a mathematical model to characterized and accurately detect these frames, whose identification is a key factor to synchronize the whole performance.

Given all this and according to the best of our knowledge, there has not yet been presented a holistic, flexible, free and cross-platform software that addresses this kind of problems. As a proof of concept, some experiments are carried out on a dataset of orchestral music, showing that the traceback stage can significantly improve the alignment accuracy while introducing only a small delay, without degrading performance and maintaining a reasonable memory consumption.

The paper is organized as follows. In Section 2 we review related works on score following and briefly describe our base system. Section 3 describes the traceback stage with configurable delay incorporated into our score follower. In Section 4, we detail the improvements over ReMAS and the silence-detection process. Experimental results are shown in Section 5 and conclusions are outlined in Section 6.

2 Backgroud of the alignment system

Audio-to-score alignment has been an important research topic since the 1990s, where computers became powerful enough to address the problem. Early systems were limited to monophonic music and based on pitch detectors. Later, systems were proposed for polyphonic music, and started using more complex signal features and alignment techniques.

An audio-to-score alignment system typically contains two steps: *feature extraction* and *alignment*. In the first stage, comparable features are computed directly from the score and the audio signal, such that both data sources are represented by the two feature sequences that must be aligned: $U = (\mathbf{u}_1, \ldots, \mathbf{u}_n, \ldots, \mathbf{u}_N)$ and $V = (\mathbf{v}_1, \ldots, \mathbf{v}_t, \ldots, \mathbf{v}_T)$. Here, N represents the number of time instants in the score and T is the number of time instants in the audio, which, for online processing, typically corresponds to the number of frames acquired up to the current time. Features proposed in the literature are designed to capture robustly the musical content, and to be as much discriminative as possible between different musical situations. The most common ones are chroma vectors [13], semigrams [10], onset-based measures [12, 15], spectral peak structure [9] or spectral bases extracted from analysis with Non-negative Matrix Factorization (NMF) [6]. In the alignment stage, the system finds corresponding time points in the two feature sequences (i.e. the corresponding position in the score for each point in time). For this task, the majority of the methods compute a comparison measure for each pair from U and V, and find the best general alignment by employing stochastic approaches based on Hidden Markov Models (HMMs) [20] or, more commonly, DTW [18].

For online alignment, one of the most famous approaches is the online DTW algorithm proposed by Dixon [10]. Like standard DTW, the method uses a local cost function to compare pairs of audio and score segments

$$d(n, t) = f(\mathbf{u}_n, \mathbf{v}_t),$$

where $f(\cdot)$ is any kind of function to compare features, and $d(n, t)$ can be viewed as the cost of aligning $\mathbf{u}_n$ with $\mathbf{v}_t$. From $d(n, t)$, an accumulated cost matrix is constructed, containing the value of the minimum cost path up to (n, t). The difference with classic (offline) DTW is that the backward path is not implemented, because, at time t, the current point (n, t) with minimum cost is returned as a result. In the method by Dixon, only the neighboring cells around the current position are considered, thus saving processing time and memory consumption. The

main drawback of the framework described in [10] is that the algorithm tends to get lost in the score, and is not able to recover due to the heavy constraints imposed on the path. The algorithm is significantly improved in [3], where a "backward-forward" approach is proposed to reconsider past decisions. Specifically, after every two audio frames, the backward path is computed starting in the current point and ending after a few steps backward. From this point "in the past", a new forward path is started, which enables to analyze if the tracking is going too fast, too slow or good. The output of the system is still online, but the tracker can detect and correct errors faster. Another improvement proposed in [3] is the incorporation of a tempo model. Again, as the backward path has better information about the performance than the forward path, its slope is used to estimate the recent tempo difference between the score and the audio. This tempo estimate is used to stretch or compress the feature sequence of the score, in order to match the performance tempo. Other authors have proposed modifications of the classic DTW algorithm to increase speed or reduce memory cost. In [14], a short-time version of DTW is proposed, which performs the matching by dividing the sequences into shorter portions, providing the same result than standard DTW under some hypothesis. Unfortunately, the algorithm is not intended to work online, but only as a tool to reduce memory consumption.

Our online audio-to-score alignment algorithm [5] is also composed of two main modules: the *feature extraction module* and the *alignment module*.

The feature extraction module pre-processes the symbolic score to represent it in a suitable format for alignment purposes. The goal is to convert the score into a time sequence of features in the form $U = (\mathbf{u}_1, \ldots, \mathbf{u}_n, \ldots, \mathbf{u}_N)$. First, the score is analyzed to detect how many unique combinations of notes occur during the piece. Each unique combination of notes is called a *score unit*. These combinations take into account the instruments involved, such that two combinations having identical note numbers but different instruments are considered as two different score units. Since the music is repetitive and many combinations of notes usually span across multiple frames, the number of units will be often much smaller than the number of frames.

Once the score is structured into score units, the feature extraction module learns a single spectral pattern for each of them. To do this, a MIDI version of the score is converted into an audio file by using a software synthesizer. The result is a synthetic low-quality signal for which the activation time of each unit is known. This signal is converted to the time-frequency domain, and decomposed using supervised NMF. As a result, a single spectral pattern is obtained for each unit.

Each feature $\mathbf{u}_n$ in the sequence U is the spectral pattern corresponding to the active combination in instant n. Regarding the audio signal, when a new frame is acquired at time t, the feature extraction module extracts a vector of features $\mathbf{v}_t$ from the frame. In our approach, this feature vector $\mathbf{v}_t$ is just the input frame converted to the frequency domain.

The alignment module follows the live musical performance online, and reports the current position of the musicians in the score. For each input frame, the alignment module computes the matching cost between $\mathbf{v}_t$ and every element in the score sequence U. The corresponding position in the score is determined from the accumulated costs given by an implementation of online DTW. In out approach the cost measure between t and every instant in the score is given by the following

distortion:

$$d(n,t) = D_\beta\left(g_{n,t}\mathbf{u}_n|\mathbf{v}_t\right), \tag{1}$$

where $D_\beta(\cdot)$ is the β-divergence function, $\beta \in [0,2]$, and $g_{n,t}$ is obtained by the following fast spectral decomposition equation [5]:

$$g_{n,t} = \frac{|\mathbf{v}_t\mathbf{u}_n^{(\beta-1)}|_1}{|\mathbf{u}_n^{\beta}|_1}. \tag{2}$$

3 Proposed DTW traceback with configurable delay

When a new audio frame arrives at time t, the matching cost $d(n,t)$ of aligning instant t with every instant n in the score ($1 \leq n \leq N$) is computed using Eq. (1). From these local distances, the accumulated cost matrix D is computed using the following recursion:

$$D(n,t) = \min_{c_n,c_t}\left\{\begin{array}{c} D(n-1,t-c_t) + d(n,t)\sigma_{1,c_t} \\ D(n-c_n,t-1) + d(n,t)\sigma_{c_n,1} \end{array}\right\},$$

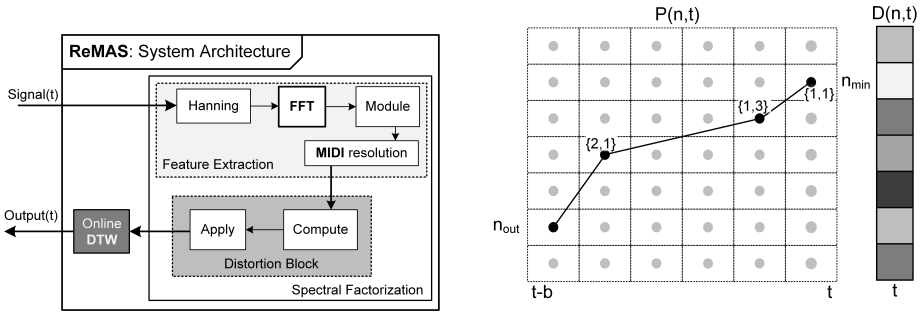
where c_n and c_t are the step sizes at each dimension, whose values are the integers in the range $c_n \in [1, C_n]$ and $c_t \in [1, C_t]$. Scalars C_n and C_t are then the maximum allowed step sizes in the score and in the performance, respectively. The weights σ control the bias toward diagonal steps, being set to $\sigma_{x,y} = \sqrt{x^2+y^2}$. Observe that $D(n,t)$ is the accumulated cost matrix of the minimum-cost path from $(1,1)$ to (n,t), and that $D(1,1)$ is initialized to $d(1,1)$, because the alignment result is constrained to include the point $(1,1)$.

The accumulated cost matrix D is filled as new audio frames arrive to the system. At each time t, the corresponding position in the score is estimated directly from the information accumulated up to t, which can be considered as a sub-optimal solution. In its simplest version, the algorithm simply returns the score position associated to the best forward path, that is, $n_{\mathtt{out}} = \arg\min_n D(n,t)$. Other versions incorporate certain improvements to estimate a better forward path, such as the use of anchor points (see [21] for details). Observe that only the last C_t columns of D are needed to compute a new one, which allows to use a circular buffer of dimensions $N \times C_t$ to implement matrix D in the computer.

For those situations in which a certain delay is allowed, the use of a limited traceback can improve the alignment results, at the expense of using more resources. In this work, we modify our online DTW to incorporate a traceback stage, in which a backward path limited to b frames in audio time is computed. At each instant t, the algorithm delivers the alignment result corresponding to the audio frame $t-b$, obtained by computing the forward path up to t and following the recursion b steps backward in the x-axis (performance). In our specific implementation, the backtracking length b (or delay) is a configurable parameter of the algorithm, so we can compare the results of the score follower for different selected delays.

In order to compute a backward path with b frames, the algorithm must store the best step sizes for each (n,t) across the last b frames. A matrix P of step sizes is constructed as follows

$$P(n,t) = \arg\min_{c_n,c_t}\left\{\begin{array}{c} D(n-1,t-c_t) + d(n,t)\sigma_{1,c_t} \\ D(n-c_n,t-1) + d(n,t)\sigma_{c_n,1} \end{array}\right\}, \tag{3}$$



(a) (b)

Fig. 1: (a) ReMAS's block diagram. (b) DTW traceback example for a delay of b frames, where the backtrack path starts at $(n_{\texttt{min}}, t)$ and ends in $(n_{\texttt{out}}, t-b)$.

where each element $P(n,t)$ stores the pair $\{c_n, c_t\}$ corresponding to the last step of the minimum cost path up to (n,t). Since b is the maximum allowed delay, P can be implemented in the computer as a matrix with $N \times b$ elements (for example, in the form of indexes to a table of allowed step sizes). Consequently, the traceback stage increases the memory consumption in comparison to online DTW, depending on the chosen delay.

At each time t, the score position corresponding to the best forward path is selected as $n_{\texttt{min}} = \arg\min_n D(n,t)$, and the backward path is constructed step-by-step from $(n_{\texttt{min}}, t)$ by reading the step sizes stored in P. After following the backward path during b audio frames at least, the new score position $n_{\texttt{out}}$ is returned as a result for the audio frame $t-b$. The traceback process is illustrated graphically with an example in Fig. 1.

4 HReMAS: Hybrid ReMAS

ReMAS, whose architecture is shown in Fig. 1 (a), is a parallel system based on a fast spectral decomposition algorithm and a variation of online DTW. HReMAS has been implemented in C with OpenMP for multicores and in CUDA for NVIDIA GPUs. In ReMAS only the forward path is computed so that the result is returned without delay. To incorporate the traceback stage assuming a configurable delay parameter it is necessary to incorporate some improvements to ReMAS. We have accomplished a major review with the aim of improving its stability, usability, and performance. The main improvements are:

1. The input to ReMAS is an unstructured file obtained from an external system that is in charge of capturing audio, something that restricts its practical use. In HReMAS, the input data comes from either *wav* audio files or from audio captured from the microphone. The input data structures have been modified accordingly in order to allow both options.
2. As a consequence of the previous modification, the *feature extraction module* has been redesigned. We have regrouped several CUDA kernels to form a small group improving, as a result, spatial and temporal locality.

Table 1: Average time per frame (in milliseconds) of ReMAS [2] and the improved ReMAS developed for HReMAS for scores of different length (in seconds).

Score length	ReMAS [2] CPU	ReMAS [2] GPU	Improved ReMAS CPU	Improved ReMAS GPU
150	1.8694	2.8822	0.8810	0.7909
300	2.6922	3.1928	1.2328	0.9114
600	4.5609	3.3213	2.1805	1.2828
900	6.8854	3.7335	3.2177	1.7268
1800	12.459	4.7536	5.8552	2.8699

3. The "Distortion block" has also been optimized, improving the relationship between the number of computing operations and memory accesses.
4. Before incorporating the traceback process to the DTW module, certain DTW operations, intended to avoid deviations, have been eliminated reducing, thus, the total computational weight of the DTW module.

Table 1 compares the average time per frame of the version of ReMAS improved in this work with the ReMAS developed in [2] under the same conditions. The target platform used to obtain the figures in Table 1 is a NVIDIA Jetson Development Kit which features a Quad-core ARM® A57 processor and a NVIDIA Maxwell™ GPU with 256 NVIDIA® CUDA® cores. As it can be seen, the changes incorporated to ReMAS have led to a substantial improvement.

As shown in Fig. 1 (b), DTW traceback consists in following the (unique) path between the final position and the initial position using the information computed with DTW in the forward stage. In addition to storing the cost (in the aforementioned matrix D) we need to store the position among those possible that reach the minimum cost path (3). HReMAS does not store the whole D, but only a submatrix with as many columns as the maximum possible step sizes (the C_t and C_n values) and operates on it in a circular way. Thus, we minimize memory consumption and the number of data block movements. We also use a circular buffer, whose size is the maximum admissible delay, to store the steps.

The process is similar to that described in [2] when the maximum delay is not reached, but without computing the minimum cost path. Once the maximum delay is reached, we enter the traceback phase. The computational impact due to traceback is analyzed in Section 5.

Another thing worth mentioning is that HReMAS does not return the minimal-cost position of each frame but the *tempo* (speed). The new tempo is computed from the minimum that was obtained and the current tempo. This computation is sensitive to small differences. This is why we have implemented a unidimensional Gaussian filter of width equal to the number of steps (again the C_t and C_n constants). This filtering results in a more conservative system with less fluctuations and more accurate results.

In addition, an algorithm to detect silence audio frames was incorporated into the system. This extension allows the system to know when the musicians are not playing (either because they have not started yet or because of pauses in the middle of the performance) to stop temporally the alignment during those moments. A spectral pattern representing silence is defined as $\mathbf{u}_0$. This pattern is a vector set to ones that models the flat spectral shape of silence and gaussian noise. Then,

the distortion between the input frame and $\mathbf{u}_0$ is computed similarly to (1), as follows:

$$d_{\texttt{silence}}(t) = D_\beta \left(g_{0,t}\mathbf{u}_0 | \mathbf{v}_t \right), \tag{4}$$

where $g_{0,t}$ is computed as in (2). From all distortions $d(n,t)$ and $d_{\texttt{silence}}(t)$, the silence likelihood is finally determined as

$$p_{\texttt{silence}}(t) = F_{\texttt{sig}} \left(\min_n d(n,t) - d_{\texttt{silence}}(t) \right), \tag{5}$$

where $F_{\texttt{sig}}(\cdot)$ is the sigmoid function $F_{\texttt{sig}}(x) = 1/(1+e^{-x})$. To incorporate certain temporal information from past audio frames, the decision is made by a causal HMM with 2 states: "silence" and "non-silence", with likelihoods equal to $p_{\texttt{silence}}(t)$ and $1 - p_{\texttt{silence}}(t)$, respectively. The transition probabilities of this HMM are chosen to maintain certain steadiness in each state.

The silence detector is executed for each frame before continuing with the DTW computations. In other words, if the current frame is detected to be silence, then the frame is discarded, and no new columns of D nor P are computed. Otherwise, the DTW module proceeds as usual.

5 Experiments

The audio database used for evaluating our alignment accuracy was proposed in [19], and consists of four passages of orchestral music from the Classical and Romantic periods. The dataset includes the scores of the compositions, as well as the ground-truth alignment between audio and score (generated in [16]). The four pieces differ in terms of number of instruments sections, style, dynamics, and size of the orchestra. The first piece is Symphony no. 7 by Beethoven, featuring big chords, string crescendos and relatively long pauses. The second passage is Bruckner's Symphony no. 8, corresponding to the late Romantic period, and featuring large dynamics and a full orchestra. Mahler's Symphony no. 1 also features a large orchestra and a typical Romantic style, but is considerably more complex than Bruckner's piece. The last excerpt is a segment from the opera Don Giovanni by Mozart, with Classical style and played by a small group of musicians, including a soloist segment. A detailed description of the database is provided in Table 2, including measurements of the complexity of the compositions calculated from the score, such as the polyphony density (average number of simultaneous notes per frame) and the inter-onset duration (average time gap between onsets).

As explained in [19], the recordings were made in an anechoic chamber using professional recording equipment. To simulate a more challenging recording scenario, the audio signals were played with a Genelec 8020B loudspeaker in a room with dimensions 7.25 m × 8.20 m (a regular laboratory), and captured with an inexpensive Genius MIC-01A microphone situated 2 m away from the loudspeaker. The resulting files were used for our alignment experiments. All audio files are sampled at 44100 Hz.

Results are expressed in terms of alignment accuracy, which is defined as the percentage of correctly aligned notes in the score [7]. A note is considered to be correctly aligned if its reported onset does not deviate more than a threshold (or tolerance window) from the reference alignment. Typical threshold values range from 50 to 500 ms. Missed notes (i.e. notes present in the reference score but not in

Table 2: Characteristics of the orchestral dataset used for the evaluation of our score following system. For the polyphony density and the inter-onset duration, both the mean and the standard deviation (in parenthesis) are included.

Composer	Piece name	Dur.	Tracks	Notes	Poly. dens.	Inter-onset dur.
Beethoven	Symphony no. 7	3m 11s	20	3075	8.7 (4.9)	0.21s (0.19s)
Bruckner	Symphony no. 8	1m 27s	39	2789	10.6 (4.5)	0.21s (0.08s)
Mahler	Symphony no. 1	2m 12s	30	2822	5.9 (3.5)	0.24s (0.23s)
Mozart	Don Giovanni	3m 47s	10	2724	5.0 (2.4)	0.26s (0.17s)

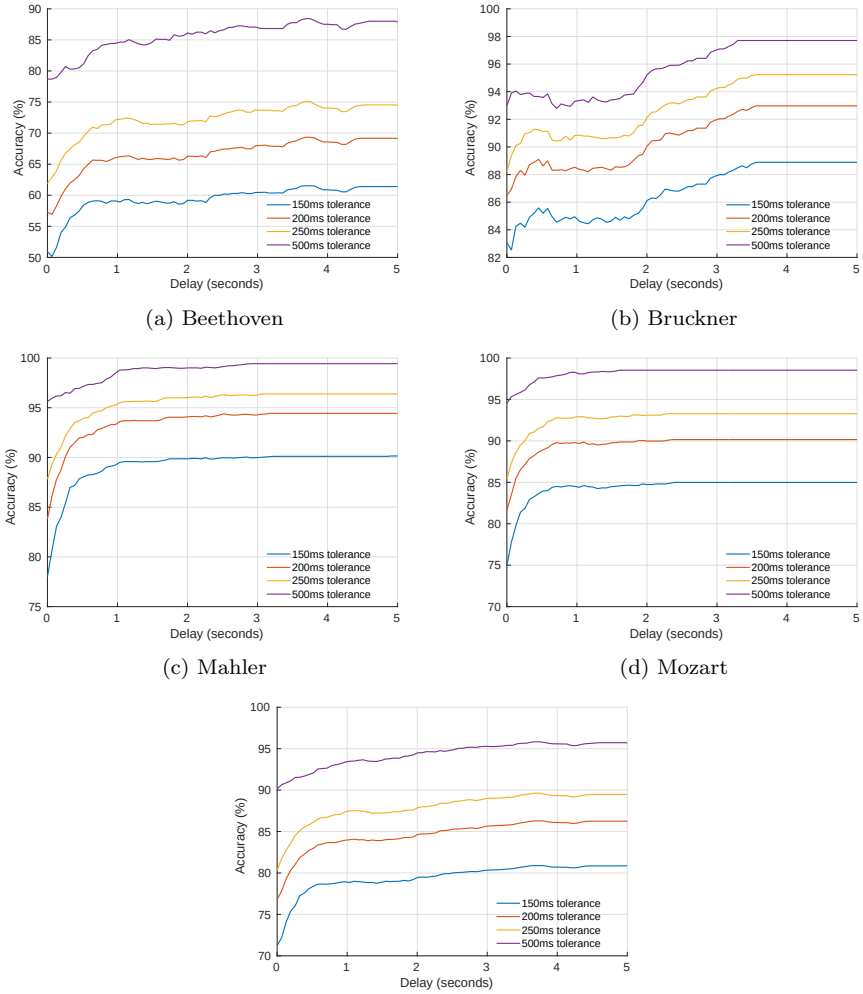
Table 3: Alignment results expressed as the percentage of correctly aligned notes for different tolerance windows. The results correspond to the mean accuracy over the four pieces for different system delays. The last column shows the best theoretical result that can be reached.

	DELAY (sec)							
TOL. (sec)	0	0.5	1	1.5	3	4.5	offline	oracle
≤ 0.05	40.7 %	40.7 %	40.7 %	40.8 %	41.0 %	41.2 %	41.3 %	93.9 %
≤ 0.10	61.6 %	67.2 %	68.0 %	67.8 %	69.0 %	69.4 %	69.5 %	98.7 %
≤ 0.15	71.2 %	78.4 %	78.9 %	78.9 %	80.3 %	80.8 %	80.8 %	99.6 %
≤ 0.20	76.8 %	83.0 %	83.9 %	83.9 %	85.6 %	86.2 %	86.2 %	99.8 %
≤ 0.25	80.3 %	86.1 %	87.4 %	87.2 %	89.0 %	89.5 %	89.5 %	99.9 %
≤ 0.30	83.4 %	88.0 %	89.3 %	89.2 %	90.9 %	91.3 %	91.3 %	100 %
≤ 0.35	85.7 %	89.3 %	90.6 %	90.7 %	92.2 %	92.7 %	92.7 %	100 %
≤ 0.40	87.6 %	90.4 %	91.8 %	92.0 %	93.5 %	93.9 %	93.9 %	100 %
≤ 0.45	89.0 %	91.3 %	92.8 %	92.9 %	94.5 %	94.9 %	94.9 %	100 %
≤ 0.50	90.1 %	92.1 %	93.4 %	93.6 %	95.3 %	95.7 %	95.7 %	100 %

the result) and notes whose onsets are out of the tolerance window are considered as misaligned.

The alignment accuracy was evaluated for different values of delay, from 0 to 5 sec., and different tolerance windows (from 50 ms to 500 ms). The results of each individual piece and the global results (over the whole database) are illustrated in Fig. 2. Table 3 shows accuracy in numbers including also the offline DTW for comparison. The column "oracle" shows the best theoretical alignment result that can be reached for each tolerance window. This limit exists because, in practice, musicians do not play all onsets perfectly synchronized between themselves, but with a deviation that, in some cases, can be longer than the tolerance window.

Subfigure (e) of Figure 2 summarizes the behaviour observed in the other figures: as the delay becomes longer the alignment accuracy improves. However, this improvement is noticeable only for short delays and narrow tolerance windows. In particular, for tolerance windows between 150 and 250 ms, the overall accuracy difference between a system with 1 sec. delay and a fully online system is around 7-8%. Beyond that point, the accuracy improves more slowly, reaching 9-10% of improvement for a delay equal to 4.5 sec. It does not make much sense to increase the delay further, because the score follower reaches its limit around that point. According to our experiment, for delays longer than 4.5 sec. the accuracy is always very similar, almost matching those obtained by the offline system. This is



(a) Beethoven (b) Bruckner

(c) Mahler (d) Mozart

(e) All

Fig. 2: Alignment accuracy results as a function of delay and tolerance window.

an important finding for certain applications such as audio-to-midi matching systems [17], because it reveals that, for a particular midi-audio pair, the matching measure does not improve with longer queries.

As observed in Fig. 2, the impact of the delay is more significant for audios difficult to align. For *beethoven* and a tolerance window of 150 ms, a delay of 1 sec. achieves around 9% more of accuracy. For *mahler*, this improvement is around 11% also for 1 sec. delay. For *bruckner*, the delay is not very beneficial, because the online system obtains a very good alignment result already. In all files, the results do not improve significantly beyond a certain delay value, although it seems that this value depends on the file, ranging from 1 sec. for *mozart* to 4-5 sec. for *beethoven* and *bruckner*.

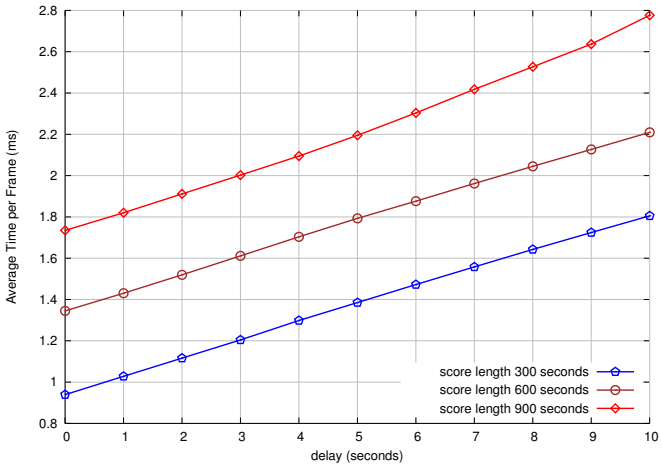


Fig. 3: Average time per frame with delay oscillating between 0 and 10 seconds.

Finally, Fig. 3 shows the average time per frame when the delay varies between 0 and 10 seconds, when the GPU is used, for medium/long compositions. As it can be seen, time grows linearly with the delay, but remains within the limits established in [2] as real-time.

6 Conclusions

In this paper, we analyzed the effect of the latency in the alignment accuracy achieved by our score following algorithm. In general, many score followers have a trade-off between accuracy and delay, because future information allows to improve the alignment. An adequately selected delay can increase performace, and still supports real-time applications. We modified our score follower, based on online DTW, by incorporating a traceback stage with configurable delay. Experiments on a database of orchestral music showed that a small increment of the latency produces a certain improvement of the accuracy. Furthermore, beyond a certain delay, the results are almost equal to offline alignment. We conclude that this modification can be useful for applications not too sensitive to delay, such as online score-informed source separation, sheet page turning or audio-based searching of compositions.

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, 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)

2. Alonso, P., Vera-Candeas, P., Cortina, R., Ranilla, J.: An efficient musical accompaniment parallel system for mobile devices. The Journal of Supercomputing **73**(1), 343–353 (2017)
3. Arzt, A.: Flexible and robust music tracking. Ph.D. thesis, Johannes Kepler University Linz, Linz, Österreich (2016)
4. Arzt, A., Widmer, G., Dixon, S.: Automatic page turning for musicians via real-time machine listening. In: Proceedings of the 18th European Conference on Artificial Intelligence (ECAI), pp. 241–245. Amsterdam, The Netherlands (2008)
5. Carabias-Orti, J., Rodríguez-Serrano, F., Vera-Candeas, P., Ruiz-Reyes, N., Cañadas-Quesada, F.: An audio to score alignment framework using spectral factorization and dynamic time warping. In: Proc. ISMIR, pp. 742–748 (2015)
6. Cont, A.: Realtime audio to score alignment for polyphonic music instruments, using sparse non-negative constraints and hierarchical hmms. In: 2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings, vol. 5, pp. V–V (2006)
7. Cont, A., Schwarz, D., Schnell, N., Raphael, C.: Evaluation of Real-Time Audio-to-Score Alignment. In: International Symposium on Music Information Retrieval (ISMIR). Vienna, Austria (2007)
8. Dannenberg, R.B., Raphael, C.: Music score alignment and computer accompaniment. Commun. ACM **49**(8), 38–43 (2006)
9. Devaney, J., Ellis, D.: Handling asynchrony in audio-score alignment. In: Proceedings of the International Computer Music Conference Computer Music Association, pp. 29–32 (2009)
10. Dixon, S.: An on-line time warping algorithm for tracking musical performances. In: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), pp. 1727–1728 (2005)
11. 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)
12. Ewert, S., Muller, M., Grosche, P.: High resolution audio synchronization using chroma onset features. In: Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on, pp. 1869–1872 (2009)
13. Hu, N., Dannenberg, R., Tzanetakis, G.: Polyphonic audio matching and alignment for music retrieval. In: Applications of Signal Processing to Audio and Acoustics, 2003 IEEE Workshop on., pp. 185–188 (2003)
14. Kaprykowsky, H., Rodet, X.: Globally optimal short-time dynamic time warping, application to score to audio alignment. In: Proc. International Conference on Acoustics Speech and Signal Processing Proceedings, vol. 5, pp. V–V (2006)
15. Li, B., Duan, Z.: An approach to score following for piano performances with the sustained effect. IEEE/ACM Transactions on Audio, Speech, and Language Processing **24**(12), 2425–2438 (2016)
16. 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**(8363507), 1–19 (2016)
17. Muñoz-Montoro, A., Cabañas-Molero, P., Bris-Peñalver, F., Combarro, E., Cortina, R., Alonso, P.: Discovering the composition of audio files by audio-to-midi alignment. In: Proc. 17th International Conference on Computational and Mathematical Methods in Science and Engineering, pp. 1522–1529 (2017)
18. Orio, N., Schwarz, D.: Alignment of monophonic and polyphonic music to a score. In: Proc. Int. Comput. Music Conf. (ICMC), pp. 155–158 (2001)
19. Pätynen, J., Pulkki, V., Lokki, T.: Anechoic recording system for symphony orchestra. Acta Acustica united with Acustica **94**(6), 856–865 (2008)
20. Raphael, C.: Music plus one and machine learning. In: Proceedings of the 27th International Conference on Machine Learning (ICML), pp. 21–28 (2010)
21. Rodriguez-Serrano, F.J., Carabias-Orti, J.J., Vera-Candeas, P., Martinez-Munoz, D.: Tempo driven audio-to-score alignment using spectral decomposition and online dynamic time warping. ACM Trans. Intell. Syst. Technol. **8**(2), 22:1–22:20 (2016)