Web Applications for Automatic Audio-to-Score Synchronization with Iterative Refinement

Adriano Baratè, Goffredo Haus
Luca A. Ludovico, Giorgio Presti
Laboratory of Music Informatics
Department of Computer Science
University of Milan
{first.last}@unimi.it

Stefano Di Bisceglie
Alessandro Minoli
Department of Computer Science
University of Milan
{first.last}@studenti.unimi.it

Davide A. Mauro
Department of Computer
and Information Technology
Marshall University
maurod@marshall.edu

ABSTRACT

The task of aligning a score to corresponding audio is a well-studied problem of particular relevance for a number of applications. Having this information allows users to explore the materials in unique ways and build rich interactive experiences. This contribution presents web applications that deal with the problem by implementing a two-step synchronization process. The first step implements a score-informed alignment while the second one can be seen as a further refinement, particularly useful for a previous manual or semi-automatic synchronization. These web implementations are specifically conceived to work with the IEEE 1599 standard, which allows for multiple instances of scores and audio renderings to be mutually synchronized together. By adopting web technologies, users are not tied to any specific platform. Evaluations of the performances and current limitations of these processes will be presented.

1. INTRODUCTION

This project originated from a practical need that emerged in the production and editing of multimedia materials. IEEE 1599 is an XML-based standard format aimed at the multilayer representation of music [1] . To this end, an IEEE 1599 document typically contains anywhere between one and many links to external audio files, scores, and timed information on the occurrence of musical events in order to support audio-to-score alignment. For a high-level description of the standard and a recent discussion on the benefits of possible applications to digital archives, see [1]). In this context, the process of synchronization, which involves identifying the note onsets, is often done manually by experts, thus obtaining relatively precise timings of musical events, but it is very time-consuming. Even if this may not be sufficient for some types of applications (for example, timed score following), it may constitute a valid starting point for automatic post-processing in the absence of a fully-automated procedure. A manual annotation process can also result in poor annotations due to human inaccuracy and delays introduced by the acquisition system. In our current implementation the first step, aimed at synchronizing audio to a reference score, draws inspiration from the literature and uses Dynamic Time Warping (DTW) to align two chromagram representations.

The second, or refinement, step is also based on a modified approach already presented in literature and an iterative time-correction algorithm specially designed for this purpose. It is worth noting that this second step can be employed regardless of the synchronization mechanism adopted for the previous step.

The implementations are currently available in the form of web applications that operate on IEEE 1599 documents. Unlike other methods, such as [2], a web implementation would ease the adoption for researchers who are not experts in locally deploying complex machine learning architectures.

The paper is structured as follows: in Sections 2 and 3 we provide some general definitions and the approaches that guided our solutions. In Section 4 we provide details about the implemented solutions, while in Section 5 we propose metrics to evaluate the performances. Finally, in Section 6 we will present the results achieved in the test phase, and, in Section 7 we present the current limitations and further possible directions for this research.

2. SCORE-INFORMED TRACKING

For the first part of our process, we based our implementation on the work presented in [3], namely a score-informed tracking algorithm.

The process can be described as follows: first, chromagrams are extracted from both the score and the audio; once these two vectors are available, Dynamic Time Warping (DTW) is employed to find the best alignment of two temporal sequences.

$X := (x_1, x_2, \dots, x_N)$ represents the chroma vectors extracted from the score while $Y := (y_1, y_2, \dots, y_M)$ are the one extracted from the audio with $x_n, y_m \in \mathcal{F}, n \in [1 : N], m \in [1 : M]$, where $\mathcal{F}$ is a feature space. Our implementation differs from the original one because both chromagrams are computed to have the same number of vectors $N = M$.

[1] For more information about the initiative, documentation, and examples, please visit https://ieee1599.lim.di.unimi.it/

Let $c : \mathcal{F} \times \mathcal{F} \rightarrow \mathbb{R}$ be the difference between the two vectors belonging to $\mathcal{F}$. We obtain the matrix $C(n,m) := c(x_n, y_m)$.

A warping path is defined as $p = (p_1, \dots, p_L)$ with $p_l = (n_l, m_l) \in [1:N] \times [1:M]$ for $l \in [1:L]$ being monotonic: $1 = n_1 \leq n_2 \leq \dots \leq n_L = N$ and $1 = m_1 \leq m_2 \leq \dots \leq m_L = M$. The following condition on the step size is also satisfied: $p_{l+1} - p_l \in (1,0),(0,1),(1,1)$. The total cost of p is defined as $\sum_{l=1}^{L} c(x_{n_l}, y_{m_l})$. Once an optimal path p^* has been obtained we recursively compute a new matrix D, that has the same size of C, where every element $D(n,m)$ contains the total cost of an optimal warping path from t $(x_1, \dots, x_n)$ and $(y_1, \dots, y_m)$. A new weight vector is also introduced $(w_d, w_h, w_v) \in \mathbb{R}^3$. The recursive definition of D is the following:

$$D(n,m) := min \begin{cases} D(n-1,m-1) + w_d \cdot c(x_n, y_m), \\ D(n-1,m) + w_h \cdot c(x_n, y_m), \\ D(n,m-1) + w_v \cdot c(x_n, y_m), \end{cases} \tag{1}$$

For $n, m > 1$. We also have that, $D(n,1) := \sum_{k=1}^{n} w_h \cdot c(x_k, y_1)$ for $n > 1$, $D(1,m) = \sum_{k=1}^{m} w_v \cdot c(x_1, y_k)$ for $m > 1$, and $D(1,1) := c(x_1, y_1)$.

A diagram of the complete process is depicted in Fig. 1.

3. ONSET DETECTION

Onset-detection techniques aim at locating the occurrences of the beginning of sound events in an audio signal. Many of the concepts presented below have been formally defined in [4].

The following definitions can help to understand some key concepts:

- The note's *attack* is the time interval in which the note sound's amplitude envelope grows;
- The *transient* is the time interval in which the signal evolves in a relatively unpredictable way;
- The *onset* is a specific timing chosen to mark the note's transient, often corresponding to the beginning of the transient or the first instant in which the transient can be reliably detected, marking the beginning of the sound event, which is what we are interested in.

Finding the onset time is usually achieved by reducing the signal to an Onset Detection Function (ODF), which ideally peaks in correspondence with an onset [4].

Fig. 2 provides a graphical representation of these concepts for the ideal case of a single note and a common scheme for an onset detection algorithm.

3.1 The Complex Domain ODF

Concerning the reduction of a signal to an ODF, there are mainly two categories: energy-based approaches and phase-based ones, each one characterized by pros and cons. Please note that ODFs detect the physical onsets of a signal. The more the attacks of the notes are slow and/or soft, the more these offsets tend to deviate temporally from the ones perceived when listening.

There is a third signal-reduction approach that presents the advantages of both the energy-based and the phase-based approaches since it simultaneously considers the information related to the signal energy and phase. This approach, called *Complex Domain* (CD) [5], is implemented in our proposed application.

In the energy-based and phase-based reduction approaches, we assume that, locally, in regions where the signal is stationary, its amplitude and instantaneous frequency values remain constant. To extend the assumption to both variables, a prediction of the signal values in the complex space is introduced and then compared with the actual measurement.

Let the measured value for the k-th bin of the n-th frame of the STFT be:

$$X_k(n) = R_k(n) e^{j\phi_k(n)} \tag{2}$$

where R_k and ϕ_k are the module and the phase of the current frame, respectively.

Then, the predicted value for the same bin and frame can be computed as:

$$\hat{X}_k(n) = \hat{R}_k(n) e^{j\hat{\phi}_k(n)} \tag{3}$$

where the target module $\hat{R}_k(n)$ corresponds to the module of the previous frame $X_k(n-1)$ and the target phase $\hat{\phi}_k(n)$ corresponds to the sum of the phase of the previous frame and the phase difference between the two previous frames:

$$\hat{\phi}_k(n) = princarg\Big(2\phi_k(n-1) - \phi_k(n-2)\Big) \tag{4}$$

The stationarity of the k-th bin can be calculated by measuring the Euclidean distance in the complex space between predicted and measured vectors.

$$\Gamma_k(n) = \left\{ \Big[\Re(\hat{X}_k(n)) - \Re(X_k(n))\Big]^2 + \Big[\Im(\hat{X}_k(n)) - \Im(X_k(n))\Big]^2 \right\}^{1/2} \tag{5}$$

By adding the stationarity measurements along all frequency bins, we can construct the Complex Domain ODF as:

$$CD(n) = \sum_{k=1}^{K} \Gamma_k(n) \tag{6}$$

CD presents sharp peaks at the low-stationarity points of the signal, and it is a robust spectrum-based ODF known in literature [5] and still currently used as benchmark (see [6]). As a further improvement, the proposed application does use a differential version of CD. This choice is motivated by the fact that we want to give more importance to sudden variations in the stationarity of the signal rather than to its "absolute" stationarity.

Figures 3 and 4 report a comparison between pure CD and the differential version on two audio fragments of different types. As a result, the peaks of the latter version tend

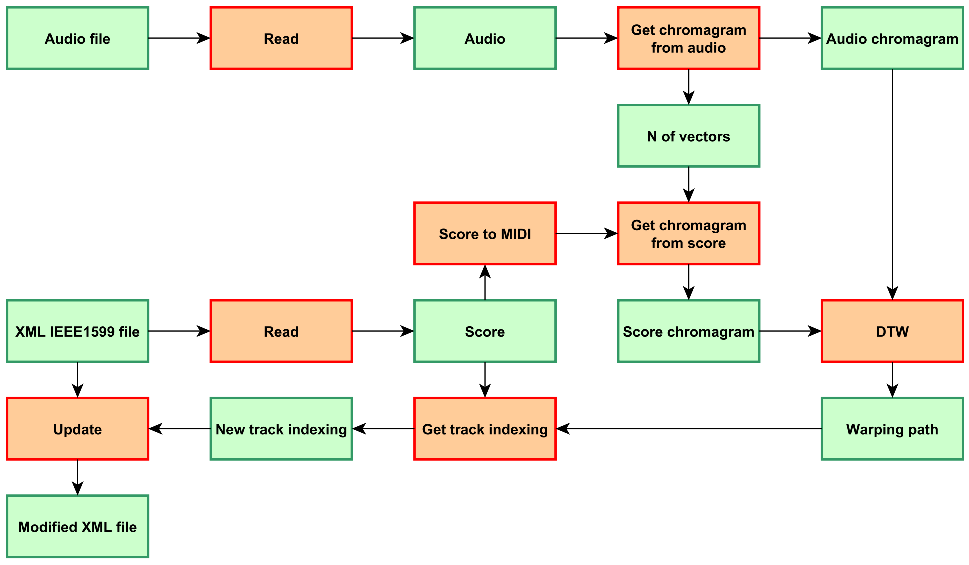

Figure 1. Functional diagram for the implemented solution for Step 1. Green marks data while red marks functions that process data.

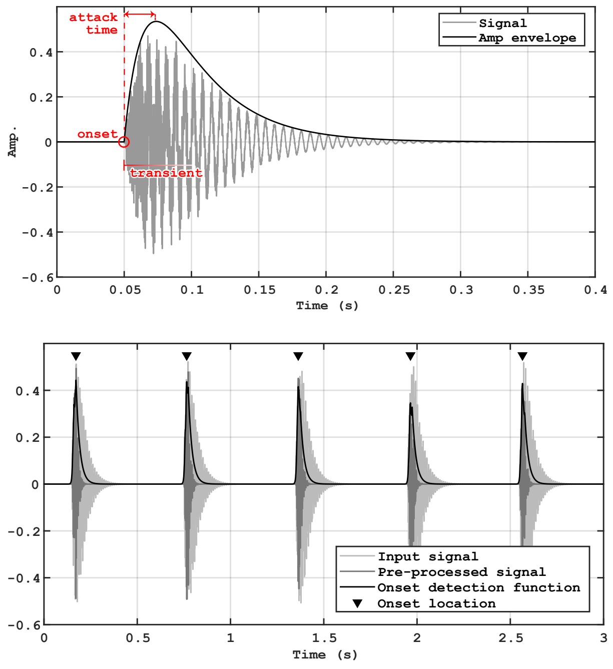

Figure 2. Top: An example of a single note's attack, transient, and onset. Bottom: The phases of a typical onset detection algorithm.

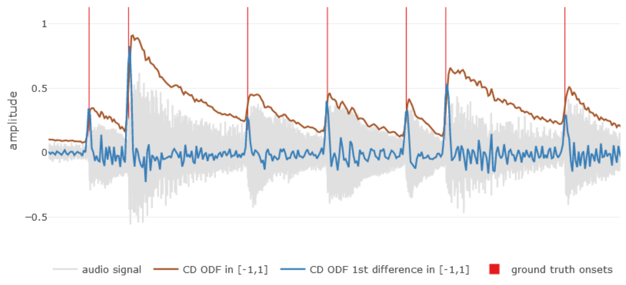

Figure 3. Pure and differential CD versions for a piano signal.

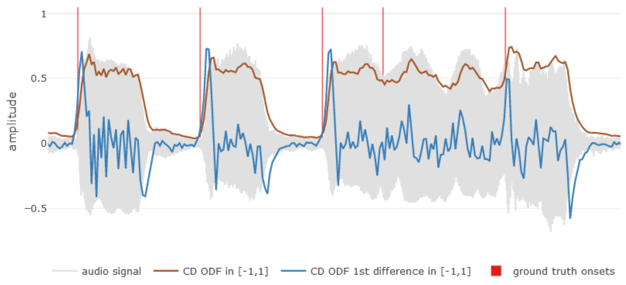

Figure 4. Pure and differential CD versions for a saxophone signal.

to adhere more to the ground-truth onsets of the signal than those of the former one.

3.2 Peak Selection

Once the ODF has been calculated, to facilitate the process of peak selection, it is possible to normalize and reduce the effect of the noise through smoothing. After the post-processing, the ODF still contains peaks that do not correspond to any signal onset. It is necessary to have a threshold that discriminates the onset peaks from the non-significant ones. A possible strategy is to set a positive value δ and consider the peaks as onsets only when $ODF(n) \geq \delta$. However, this choice would be suitable only for signals with few variations in dynamics; in other cases, there would be a tendency to detect fewer onsets in the signal sections with lower dynamics, and too many onsets where the dynamic level is high.

In literature, the most common definition of threshold for this context makes it adapt to variations in the amplitude of the ODF, in particular by exploiting a moving median, as shown here:

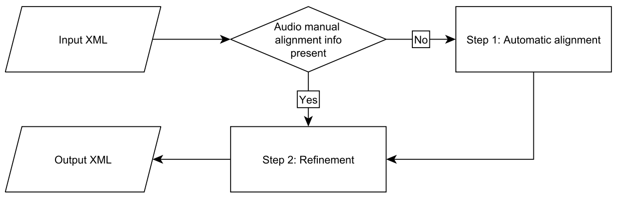
Input XML
Audio manual alignment info present
No
Step 1: Automatic alignment
Yes
Output XML
Step 2: Refinement

Figure 5. Overall view of the implemented solution.

$$\tilde{\delta}(n) = \delta + \lambda \, median\Big\{ODF(n-M), ..., ODF(n+M)\Big\} \tag{7}$$

where δ and λ are two constants that must be manually calibrated by experimenting with signals of various types and M is the window width of the moving median. Often λ is set to 1 because it does not affect the results significantly. Conversely, the choice of δ is critical, since it behaves like an absolute threshold offset used to reduce the sensitivity to very low ODF values. M should represent the longest time interval within which the dynamic of the piece is not expected to change ($\approx 100\,\text{ms}$) [5].

After subtracting the threshold from the ODF, its local maxima greater than 0 are chosen as the onsets.

4. WEB IMPLEMENTATIONS

The general goal of the proposed solution is to provide web tools to perform audio-to-score synchronization. This process can be seen as organized into two steps, as visible in Fig. 5: the first step is the raw synchronization of audio events' onsets with symbolic score events, which can be performed in a manual, semi-automated, or automated way. The latter automated approach minimizes the effort of producing a complete multi-layer musical document. The inputs required are symbolic and audio information, respectively; the expected output is a music document encoding an acceptable audio-to-score synchronization.

Step 2 is a refinement process aiming to improve the onset synchronization of the events previously associated. For the refinement step, in input, the application requires an audio track on which to perform onset detection and a list of onset timings expressed in seconds to be checked. In output, it returns a list of readjusted timings that should improve the list supplied in the input.

The proposed solution is based on the IEEE 1599 format, an internationally recognized XML-based standard aiming at a comprehensive representation of music. Structured as a multi-layer document, an IEEE 1599 file potentially includes the *General*, the *Logic*, the *Structural*, the *Notational*, the *Performance*, and the *Audio* layers. Providing an in-depth description of the format and its characteristics would go beyond the scope of this paper; the interested reader can refer to the scientific literature on the subject [7–9]. For our purposes, it is sufficient to know that an IEEE 1599 document can host multiple audio tracks referring to a single music piece. Musical events are listed in a data structure known as the *spine* and, in the *Audio* layer, these labels are associated with timings. Different audio tracks potentially present different associations between musical events and timings, since they may come from different performances.

In the IEEE 1599 format, rests are considered musical events, too. [2] When rests are logically aligned with sound events, the algorithm relies on the onsets of the latter to synchronize the occurrence of rests. If, on the other hand, rests take place simultaneously for all voices, the algorithm does not correct their time positions.

The application adopts the following languages and libraries:

- the *Web Audio API*, [3] a powerful and versatile audio control system for the web that lets developers perform audio operations such as adding effects, analyzing audio features, etc. In this work, the *Web Audio API* is used to obtain the arrays of samples for the audio files of the tracks to be synchronized;
- *Plotly*, [4] a high-level and declarative open-source library for creating graphics;
- *Nayuki's Fast Fourier Transform (FFT) algorithm*, [5] a free and lightweight implementation for the optimized calculation of the Discrete Fourier Transform (DFT);
- the *IEEE 1599 framework*, a platform capable of managing and manipulating IEEE 1599 documents, used to extract the event timings to be readjusted. Please note that the approach described in this paper can be easily extended to other formats to represent event timing (e.g., CSV files or plain-text formats with a list of markers, such as the ones exported by Digital Audio Workstations); in these cases, the IEEE 1599 technology would not be required.

Our solution implements the process described in Section 3 and shown in Fig. 2. The spectrum of the input audio signal is calculated through a STFT with a window size of 2048 samples and a hop size of 441 samples, as suggested in [10].

For the validation of the refinement step, several ODFs have been implemented and tested before choosing the final one. Further information and an in-depth analysis of the results are available in the Appendix (see Section 7).

The chosen ODF is standardized, i.e. scaled so that its mean is set to 0 and its standard deviation to 1. Then, a threshold is calculated by means of a moving median, centered in the current frame, with a window size equal to 21 frames, and subtracted.

The definition of the threshold is the one specified in Equation 7 with $\delta = 0$ and $\lambda = 1$. As mentioned before, the parameter δ is critical for the discrimination of each peak as an onset. Normally, this value is chosen to

[2] Conversely, in a computer-driven performance format (e.g., MIDI or SASL/SAOL) rests would result from the absence of messages/sound.

[3] `https://developer.mozilla.org/en-US/docs/Web/API/Web_Audio_API`

[4] `https://plotly.com/javascript`

[5] `https://www.nayuki.io/page/free-small-fft-in-multiple-languages`

try to maximize onset detection on entire datasets. Choosing $\delta = 0$ frees the user from the necessity to conduct an in-depth analysis based on data that is not available; In our approach, the handling of low ODF values is provided by a latter stage of processing. Fig. 4 shows the graph of an ODF with the mentioned threshold. All its positive local maxima in a sliding window of size 7 are considered as onsets. Fig. 6 shows the graph of an ODF with the respective peaks. We can observe that – due to $\delta = 0$ – all local maxima are marked as onsets, instead of the more pronounced ones only, as in traditional approaches. For each peak selected as an onset, the time and thresholded ODF amplitude values are stored.

The strategy for adjusting timings is based on *time neighborhoods*. Specifically, for each onset in the source document, we consider all the peaks selected at the end of the previous phase that fall within a time neighborhood of user-defined width; in this sense, the implementation lets the user improve the recognition of onsets based on some features easily retrievable by a perceptual analysis. In particular, the user can discriminate between tracks with sharp and soft onsets. This choice influences the width of the neighborhoods: 0.1 s in the former case, 0.05 s in the latter. While this is currently a manual setting in future works an automatic detection and dynamic adaptation to the track type could be implemented thanks to the "transient presence detection" algorithm described in [11]. As mentioned before, the candidate peaks are more numerous than the real onsets of the signal, and more than one peak may fall within each time neighborhood, since with $\delta = 0$ we are keeping all detected peaks.

To adaptively discard low ODF values, the choice falls on the peak which has the maximum ODF value in the neighborhood and, therefore, corresponds to the point of minimum stationarity of the signal in the neighborhood. This choice seems meaningful from a physical point of view and allows to easily overcome the problem of the overabundance of peaks. Fig. 7 show two examples of time-adjustment graphs. In the image, tapping onsets refer to the result of a manual process where users can use a web application to manually produce the synchronization.

Some additional scenarios must be taken into account. First, depending on the frequency of musical events and the window width, multiple neighborhoods of original timings can overlap. Overlaps are managed by considering a peak only as a part of the neighborhood of the closest timing. Moreover, it is possible that a neighborhood does not contain any peak of the ODF; in this case, the original time is validated with no adjustment.

After the first readjustment phase based on the ODF, we consider the average deviation between the new and the originally notated time instants. If such a deviation is positive, the original positions on average occurred too early, and, if negative, too late with respect to the ideal solution. The correction step is made iterative by translating the notated timings by the average deviation and then performing a new adjustment step. The whole process is repeated until either the average deviation between input and output timings reaches a low threshold value or 10 iterations have been performed. The results obtained during the last iteration represent the final output.

The final output is an IEEE 1599 document where the *Audio* layer contains a revised version of track timings and a diagram similar to Fig. 7 to provide a graphical representation of the adjustments.

5. EVALUATION METRICS

This section aims to evaluate the performance of the algorithm on some relevant music examples. Since ground-truth timings are needed, we obtained or produced them by following two processes:

1. By using *Onset_Leveau*, a publicly-available dataset made of 17 tracks with ground-truth onset timings [12]. These time values, manually annotated by a group of expert listeners, are considered a reliable ground truth on which to measure the performance of onset detection algorithms. The tracks are heterogeneous; in particular, they can be divided into solo performances of monodic instruments (M), solo performances of polyphonic instruments (P), and complex mixtures of several instruments (CM); [6]

2. By transcribing ad-hoc scores into *MuseScore*, an open-source digital score editor, and exporting the excerpts as IEEE 1599 documents thanks to an ad-hoc plugin. Audio tracks can be obtained as renderings produced by *MuseScore* itself in MP3 format. This operation also permitted obtaining an authoritative list of accurate timings in the MP3s. Such a list was extracted by invoking a Python script on the corresponding MIDI files created by *MuseScore* thanks to the *Mido* library. [7]

As the algorithm employed in the second step was originally developed to adjust potentially inaccurate timings, new lists of timings for music events had to be manually produced and incorporated into the IEEE 1599 documents via the *Audio Mapper* plugin of the *IEEE 1599 Framework*. In this way, for both approaches, we had a starting scenario in which the temporal localization of events was not necessarily accurate and a ground truth that represented the ideal results expected from the application of the algorithm.

In order to assess the proposed solution, we introduced the following metrics. Let us call z_{fi} the i-th ground-truth onset time of a signal, and z_{oi} the i-th estimated onset time of a signal. The residuals $z_{fi} - z_{oi}$ measure how far the estimated times differ from the ground truth.

A common metric for measuring the accuracy of estimated times is the *Root Mean Square Error* ($RMSE$), which indicates how far each estimated value is, on average, from the ground truth. $RMSE$ is defined as the standard deviation of the residuals. Another metric that shares the same goal is the *Mean Absolute Error* (MAE),

[6] The database can be downloaded from http://perso.telecom-paristech.fr/grichard/Onset_Labellizer/Leveau.zip

[7] https://mido.readthedocs.io

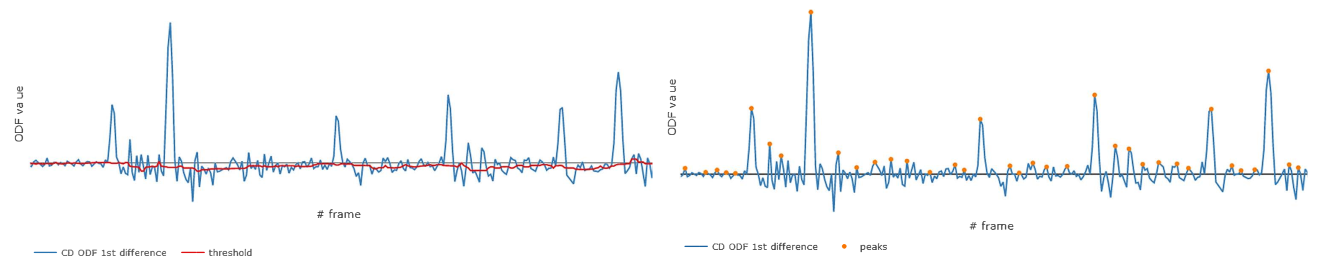

Figure 6. Left: Differential CD ODF and threshold. Right: Differential CD ODF and the peaks selected as onsets.

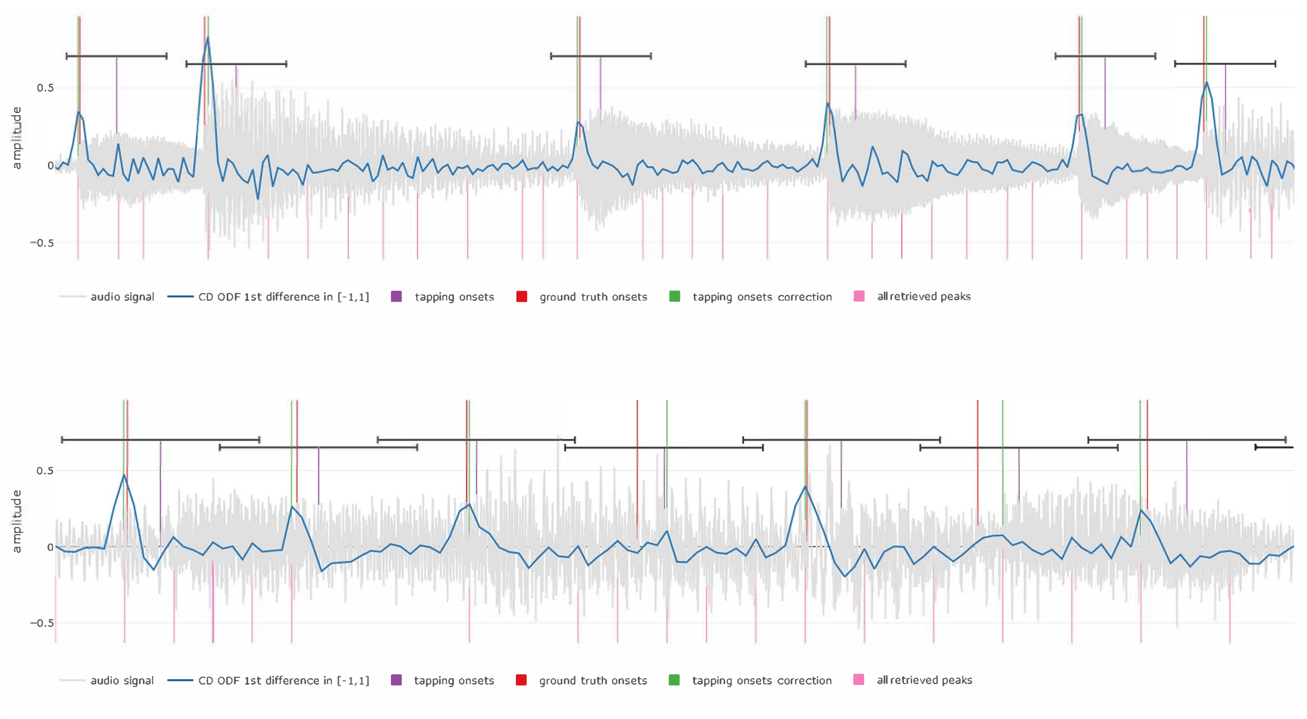

Figure 7. Examples of timing adjustments. Different "neighborhood" functions were adopted.

which measures the absolute value of the average residual. On one side, MAE penalizes all residuals equally, whereas $RMSE$, by squaring residuals, is more sensitive to estimates that deviate to a greater extent from the ground truth.

If the process improves the timings, the following conditions are expected to be satisfied:

- $RMSE_{(ground,\, adjusted)} < RMSE_{(ground,\, original)}$;
- $MAE_{(ground,\, adjusted)} < MAE_{(ground,\, original)}$.

We also compute the percentage of adjusted timings closer to the ground truth than the original ones, called $\%_M$.

6. TESTS

The score-informed tracking algorithm has been tested with a heterogeneous set of 21 tracks, 4 of which also appear in the second step (the onset detection refinement). For each track, 3 metrics have been computed against the ground truth. Results are shown in Table 1.

In order to verify the effectiveness of the proposed solution, the application was first launched on 8 test documents specially prepared in IEEE 1599 format. Test signals belong to one of the following families:

- Pitched percussive (PP) signals, for example with keyboard instruments;
- Pitched non-percussive (PNP) signals, for example strings or wind instruments;
- Complex mixture (CM) signals, i.e. complex mixtures of several instruments.

The characteristics of the analyzed tracks and the results achieved on them using two neighborhood dimensions (0.1 s and 0.05 s) are shown in Table 2. The last column provides a measure of the execution time, which is formed by the mean and the standard deviation of the times measured in 5 iterations. Please note that we are not looking for the presence or absence of an onset; conversely, we know that an onset is there from Step 1, and we want to move the onset mark to the most likely timing inside a window. For this reason, we think that scores based on mean error are most suitable than confusion matrices of found onsets.

Then, the solution was tested on excerpts from the *Onset_Leveau* dataset. Compared to the scenario of IEEE

Title	**Performer/Author**	**Ensemble**	**Avg(diff)**	**Avg(ABS(diff))**	**STD(diff)**
Ave Maria	Andrea Bocelli	Opera	-0,088s	0,238s	0,462s
Intermezzo sinfonico	Paul Bateman	Orchestra	0,368s	0,370s	0,619s
Minuetto in Sol minore	Romain Nosbaum	Piano	0,051s	0,110s	0,161s
Eleanor Rigby	The Beatles	Strings + voice	0,012s	0,046s	0,066s
I pianti che grondano	Elvira Vakhitova	Piano + voice	0,051s	0,116s	0,224s
Pineapple Rag	Tom Pascale	Piano	-0,023s	0,050s	0,070s
Gottes Macht und Vorsehung	Franz Crass	Piano + voice	0,010s	0,139s	0,170s
Musique d'enfants: Valse	Unknown	Piano	0,018s	0,076s	0,107s
City Of Stars	Ryan Gosling, Emma Stone	Piano + voice	-0,047s	0,090s	0,140s
Gymnopedie n.1	Chase Coleman	Piano	-0,051s	0,114s	0,197s
Toccata	Valeria Carissimi	Harp	0,050s	0,060s	0,060s
Ave Verum Corpus	Vienna Boys	Choir	0,039s	0,134s	0,181s
Der Hölle Rache	Cristina Deutekom	Opera	-0,184s	0,234s	0,221s
Andante religioso	Barrios Mangoré	Guitar	0,082s	0,191s	0,450s
Salve Decus Genitoris	Unknown	Choir	0,105s	0,111s	0,070s
Musette in D (T1)	Johann Sebastian Bach	Piano	-0.067s	0.069s	0.043s
Für Elise(T2)	Ludwig Van Beethoven	Piano	-0.044s	0.056s	0.054s
Por una cabeza (T3)	Carlos Gardel	String-4et	-0.050s	0.065s	0.058s
September	Earth, Wind and Fire	Disco music	-0.087s	0.087s	0.019s
The promised neverland (T4)	Isabella's Lullaby	Violin	-0.005s	0.037s	0.050s
The legend of Zelda	Koji Kondo	Orchestra	-0.102s	0.102s	0.047s

Table 1. Results achieved for selected IEEE 1599 documents.

MP3 file	🕒	**# onsets**	**type**	**original**		**narrow neighborhood**			**large neighborhood**			**runtime ± STD**
				$RMSE$	MAE	$RMSE$	MAE	$\%_M$	$RMSE$	MAE	$\%_M$	
piano1 **(T1)**	33 s	51	PP	23 ms	17 ms	19 ms	9 ms	70.59	**6 ms**	**4 ms**	**72.55**	4.33 s ± 0.03 s
piano2 **(T2)**	33 s	60	PP	27 ms	19 ms	27 ms	14 ms	76.67	**6 ms**	**5 ms**	**86.67**	4.37 s ± 0.02 s
violin1 **(T4)**	29 s	35	PNP	49 ms	40 ms	**32 ms**	**22 ms**	**71.43**	56 ms	39 ms	54.29	4.76 s ± 0.10 s
violin2	16 s	41	PNP	45 ms	43 ms	**35 ms**	**29 ms**	**80.49**	44 ms	35 ms	70.73	2.02 s ± 0.11 s
sax	22 s	39	PNP	38 ms	33 ms	35 ms	30 ms	58.97	**34 ms**	**29 ms**	**56.41**	2.76 s ± 0.18 s
string_4et **(T3)**	18 s	54	CM, PNP	20 ms	15 ms	*26 ms*	*19 ms*	*48.15*	35 ms	24 ms	44.44	2.36 s ± 0.12 s
trumpet_4et	19 s	31	CM, PNP	48 ms	43 ms	*67 ms*	*59 ms*	*22.58*	85 ms	66 ms	32.26	2.53 s ± 0.14 s
jazz_4et	12 s	53	CM, PP	39 ms	37 ms	24 ms	10 ms	94.34	**8 ms**	**5 ms**	**98.11**	1.64 s ± 0.08 s

Table 2. Results for the IEEE 1599 documents specially prepared for testing. Improvements **in bold**, declines *in italics*.

category	**narrow neighborhood**				**large neighborhood**			
	μ_{RMSE}	σ_{RMSE}	μ_{MAE}	σ_{MAE}	μ_{RMSE}	σ_{RMSE}	μ_{MAE}	σ_{MAE}
M	19 ms	3.91 ms	16 ms	3.52 ms	43 ms	9.54 ms	37 ms	9.44 ms
P	17 ms	2.15 ms	12 ms	1.52 ms	29 ms	4.71 ms	21 ms	3.36 ms
CM	18 ms	1.27 ms	14 ms	0.99 ms	25 ms	2.22 ms	19 ms	1.46 ms
sharp onsets	13 ms	1.08 ms	11 ms	0.90 ms	17 ms	1.43 ms	12 ms	1.02 ms
smooth onsets	21 ms	1.30 ms	17 ms	0.99 ms	41 ms	2.84 ms	32 ms	2.32 ms

Table 3. Aggregated results achived on *Onset_Leveau*.

1599 documents, which also contain a *Logic* level with the symbolic description of the events, in this case the score is not available. For this reason, we prepared IEEE 1599 documents which in the *Audio* layer already contained ground-truth timings. In an ideal case, the algorithm should not change them, as they already correspond to the correct time position of the onsets. The best results will be those where $RMSE$ and MAE are closest to 0.

By weighing each track by its number of onsets, the means and standard deviations were calculated, grouping results by track family and by the characteristics of their onsets. These measures are respectively denoted by symbols μ_{RMSE}, μ_{MAE}, σ_{RMSE} and σ_{MAE}, and reported in Table 3.

Concerning the IEEE 1599 test documents, the analysis of user-defined original timings highlights that, on average, they are delayed with respect to the ground truth. This can be explained by considering the way for determining timings through the *IEEE 1599 Framework*: it is a human-driven process where the user is asked to keep the time like an "adaptive" metronome able to follow possible agogic variations. In this sense, it is very uncommon to perceive an onset in advance of its occurrence, whereas a delay whose amount depends on the onset nature is normal. In general, the sharp onsets typical of a drum track are expected to be perceived with more temporal accuracy than the smooth ones of a violin track.

The values reported in Table 2 indicate that the application has synchronized the onsets of the piano and the jazz quartet tracks improving them. Due to the percussive character of the instruments, these tracks have sharp onsets that abruptly alter the energy of the signal. This results in the construction of ODFs with high peaks, therefore detectable without difficulty. Being able to detect the peaks in a robust way allows us to use a neighborhood of 0.1 s in the correction of the timings and, therefore, adjust onsets that deviate also considerably from the ground truth.

The two violin tracks and the saxophone one, even if such instruments do not have a percussive nature, were also improved. The ODFs of these signals are generally not as easily interpretable as those discussed above. The best results on violin tracks were obtained with the time interval set to 0.05 s, which assumes that the original timings are already quite accurate. Conversely, saxophone events, with an ODF similar to the one in Fig. 4, were corrected in a slightly better way with the large neighborhood. However, being the signal particularly clean, the differences in performance using the narrow or the large neighborhood interval are almost negligible. The application did not improve the tracks containing the string quartet and the trumpet quartet. In these scenarios, better results were obtained with the narrow interval. Unfortunately, effective ODFs for mixtures of instruments presenting smooth or undefined attacks are an open problem and our current approach does not provide improvements.

Concerning the *Onset_Leveau* dataset, Table 3 shows that the onset-detection accuracy by adopting the narrow neighborhood always remained below a maximum value of 28 ms in terms of $RMSE$. Aggregated data reported in Table 3 suggest that such precision is not strictly related to whether the signal contains a monophonic solo instrument, a polyphonic solo instrument, or a mix of instruments. Rather, precision always proved to be better when dealing with sharp onsets instead of smooth ones, both when the neighborhood was narrow and, even more so, when it widened.

7. CONCLUSIONS

The development of the solutions presented in this contribution is based on the extensive literature available that deals with the problem of audio-to-score synchronization. The accuracy of the recalculated onset timings is inevitably dependent on both the characteristics of the audio signal under exam and the accuracy of the original timings. The results obtained for instruments with sharp note attacks are promising, and the performance of the algorithm is good also for non-complex instrumental tracks with smooth attacks. Unfortunately, results are poor for complex instrumental mixtures, especially when note attacks are smooth.

A challenge in assessing the performances of the applications stems from having access to ground-truth timings and converting them into a suitable representation format.

The applications can be easily generalized to support other timing-representation strategies, such as the lists of markers exported by many digital audio workstations and waveform editors. The applications and companion materials can be downloaded from a GitHub repository [8] and [13] and they will be integrated into the official website of the IEEE 1599 standard. [9]

8. REFERENCES

[1] A. Baratè, L. A. Ludovico, D. A. Mauro, and F. Simonetta, "On the adoption of standard encoding formats to ensure interoperability of music digital archives: The IEEE 1599 format," in *DLfM '19: 6th International Conference on Digital Libraries for Musicology*. ACM, 2019, pp. 20–24.

[2] F. Simonetta, S. Ntalampiras, and F. Avanzini, "Audio-to-score alignment using deep automatic music transcription," in *2021 IEEE 23rd International Workshop on Multimedia Signal Processing (MMSP)*, 2021, pp. 1–6.

[3] F. Kurth, M. Müller, C. Fremerey, Y.-h. Chang, and M. Clausen, "Automated synchronization of scanned sheet music with audio recordings." in *ISMIR*, 2007, pp. 261–266.

[4] J. P. Bello, L. Daudet, S. Abdallah, C. Duxbury, M. Davies, and M. B. Sandler, "A tutorial on onset detection in music signals," *IEEE Transactions on speech and audio processing*, vol. 13, no. 5, pp. 1035–1047, 2005.

[5] J. P. Bello, C. Duxbury, M. Davies, and M. Sandler, "On the use of phase and energy for musical onset detection in the complex domain," *IEEE Signal Processing Letters*, vol. 11, no. 6, pp. 553–556, 2004.

[6] M. Schwabe, S. Murgul, and M. Heizmann, "Dual task monophonic singing transcription," *Journal of the Audio Engineering Society*, vol. 70, no. 12, pp. 1038–1047, December 2022.

[7] A. Baratè, G. Haus, and L. A. Ludovico, "A critical review of the IEEE 1599 standard," *Computer Standards & Interfaces*, vol. 46, pp. 46–51, 2016, doi:10.1016/j.csi.2016.02.001.

[8] ——, "State of the Art and Perspectives in Multi-Layer Formats for Music Representation," in *Proceedings of the 2019 International Workshop on Multilayer Music Representation and Processing (MMRP 2019)*. IEEE CPS, 2019, pp. 27–34, doi:10.1109/MMRP.2019.8665381.

[9] A. Baratè and L. A. Ludovico, "Local and Global Semantic Networks for the Representation of Music Information," *Journal of e-Learning and Knowledge Society*, vol. 12, no. 4, pp. 109–123, 2016.

[10] S. Dixon, "Onset detection revisited," in *Proceedings of the 9th International Conference on Digital Audio Effects*, vol. 120. Citeseer, 2006, pp. 133–137.

[11] G. Haus, D. A. Mauro, and G. Presti, "TRAP: Transient presence detection exploiting continuous brightness estimation (cobe)," in *Proc. 12th Int. Conf. on Sound and Music Computing (SMC-15), Maynooth, Ireland*, T. Lysaght and J. Timoney, Eds. Maynooth: Maynooth University, 2015, pp. 379–385.

[12] P. Leveau and L. Daudet, "Methodology and tools for the evaluation of automatic onset detection algorithms in music," in *In Proc. Int. Symp. on Music Information Retrieval*. Citeseer, 2004.

[13] A. Minoli, "LIMUNIMI/IEEE-1599-Onset-Refinement: Version 1.0," 5 2022, doi:10.5281/zenodo.6543247.

[8] `https://github.com/LIMUNIMI/SMC2023WebAudio-ScoreSynchronization`

[9] The web player can already be used at `https://ieee1599.lim.di.unimi.it/music_collection.php`