

# Automatic Music Synchronization Using Partial Score Representation Based on IEEE 1599

Antonello D'Aguanno  
 Laboratorio di Informatica Musicale (LIM)  
 Dipartimento di Informatica e Comunicazione (DICO)  
 Università degli Studi di Milano  
 daguanno@dico.unimi.it

Giancarlo Vercellesi  
 Laboratorio di Informatica Musicale (LIM)  
 Dipartimento di Informatica e Comunicazione (DICO)  
 Università degli Studi di Milano  
 vercellesi@dico.unimi.it

**Abstract**—This article proposes the new algorithm *ParSi*, for Partial Synchronization, which aligns different PCM executions with a partial music score representation. While in general most synchronization algorithms require a complete score representation, this algorithm is able to synchronize different PCM audio executions with a partial music score codified in MIDI. Therefore, only one MIDI instrument - chosen by user - is needed for audio-score alignment. The MIDI file is selected by taking into account the related IEEE 1599 file.

Audio analysis is performed using a notch filter. A set of convenient parameters is used during the decisional phase. Analysis of synchronization results is provided, showing that it possible to have a good automatic synchronization, even if just a partial score is available.

**Index Terms**—IEEE 1599, Music Content Analysis, Synchronization, MIDI

## I. INTRODUCTION

CONTEMPORARY digital music archives consist of huge collections of heterogeneous documents. The heterogeneity of music information makes retrieval hard to accomplish [1] [2] [3] and as a consequence many problems remain unsolved [4]. One important problem, still to be solved, is *automatic music synchronization*, which requires implementation of algorithms that automatically link different audio streams of the same piece to symbolic data formats representing different scores.

Temporal linking of score and audio data can be useful for automatic tracking of score positions during a performance. Such algorithms may be also useful for score-following applications [5]. These possibilities represent a useful tool for music students, who can listen to music audio and, at the same time, see the corresponding notes.

The state-of-the-art proposes algorithms which generally require a complete score representation. Unfortunately, there are many cases where a only partial score is available. For this reason, an algorithm is described in this article which proposes a possible solution. The *ParSi Algorithm* can synchronize different PCM audio executions thanks to only a partial music score at the symbolic level. The score has to be codified in MIDI and linked in an IEEE 1599 file. Only one MIDI instrument - selected by user - is needed for audio-score alignment.

This writing is organized as follows. Section II presents an overview of score following and synchronization algorithms proposed in the literature. Section III describes the *ParSi Algorithm*. Section IV illustrates the capabilities of IEEE 1599 to represent synchronization anchors between score and audio executions. The analysis of results is discussed in Section V, and conclusions are summarized in Section VI.

## II. RELATED WORKS

Many algorithms have been proposed in the literature, which deal with synchronization. The majority of them can be subdivided into two groups: in the first one, audio and score have to be analyzed, while in the second the realization of correct links between these two layers is required ([6][7][8][9][10][11][12][13]).

The algorithms proposed in the literature use several different systems for audio analysis, with well-known tools from audio signal processing. For example, [11] uses a Short Time Fourier Transform, [7] proposes an onset detection followed by pitch detection. In [10], the feature extraction procedure performs these operations: decomposition of the audio signal into spectral bands corresponding to the fundamental pitches and harmonics, followed by computation of the positions of significant energy increases for each band -

such positions are candidates for note onsets. Other popular solutions proposed in the literature to select the correct links between audio and score are based on the *template matching technique* ([11] [6] [12]). Such algorithms build a MIDI score to obtain a template of the real execution, which is compared to real audio using a Dynamic Time Warping, or DTW, programming techniques [14]. The correct synchronization is then obtained from the difference between the agogics of the real execution and that of MIDI.

The algorithm described here uses an approach based on recursive research with notch filter for audio analysis and a partial score event extraction from MIDI coding.

### III. THE PARSİ ALGORITHM

The algorithm proposed here is able to synchronize a MIDI score with one or more PCM audio executions using only one instrument codified in the MIDI file, which is selected by the user. Therefore, there is no attempt to synchronize the whole score with the audio, but score information is extracted from one instrument to automatically align it with audio execution (Fig. 1).

The first operation performed by the ParSi algorithm is the extraction of the file name related to audio execution and the one related to MIDI. After opening the files needed, the synchronization algorithm can start.

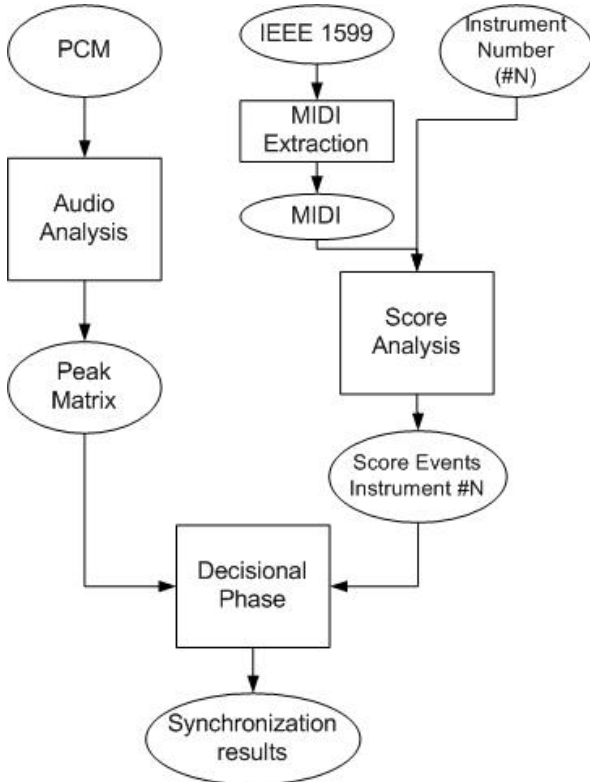


Figure 1: The ParSi architecture

It consists of three different phases.

- I. in the first phase, the selected instrument codified in the MIDI score is read in order to extract all relevant score events
- II. in the second phase, the PCM audio signal is analyzed according to the extracted score events, in order to identify all possible time-attacks notes in terms of energy peaks
- III. in the third and last phase, a decisional matching is performed to relate the event at the score level of the selected instrument with the same event at the audio level. The decisional method is based on the attack time of notes extracted at audio level.

Each phase is described in the following sections.

#### A. MIDI Score Analysis

In this phase, the selected instrument codified in the MIDI file is read to extract the score events, as shown in Fig. 2.

For each measure, only notes with a strong accent are selected, because it is easier to recognize them at the audio level. Thus the score is represented in a suitable manner for synchronization. For each strong accent, each note is codified with its fundamental frequency, the duration in MIDI ticks, the beat and measure number. Notes grouped between two asterisks (\*\*) correspond to a chord. A vertical rest is represented with -1. An example is shown in the table below (table 1); the corresponding score is shown in Fig. 3.

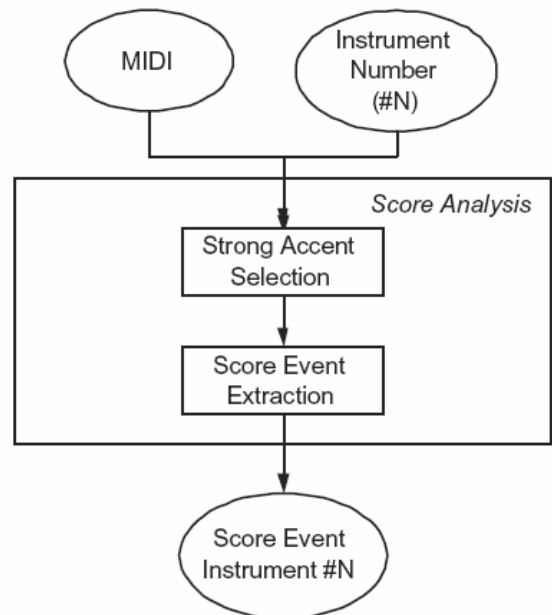


Figure 2: Score analysis flowchart



Figure 3: The score represented in table 1

TABLE I.  
EXAMPLE OF A SINGLE INSTRUMENT CODING

Fundamental Frequencies	Durations	Beat	Measure
*			
-1	480	1	1
*			
659	480	1	2
*			
523	480	1	3
*			
349	480	1	4
*			
392	480	2	1
*			
880	480	2	2
*			
415	480	2	3
*			
370	480	2	4
*			
698	960	3	1
*			
-1	960	3	3
*			

B. PCM Audio Analysis

In the audio analysis, a recursive, second-order notch filter bank [15] is used, as shown in Fig. 4.

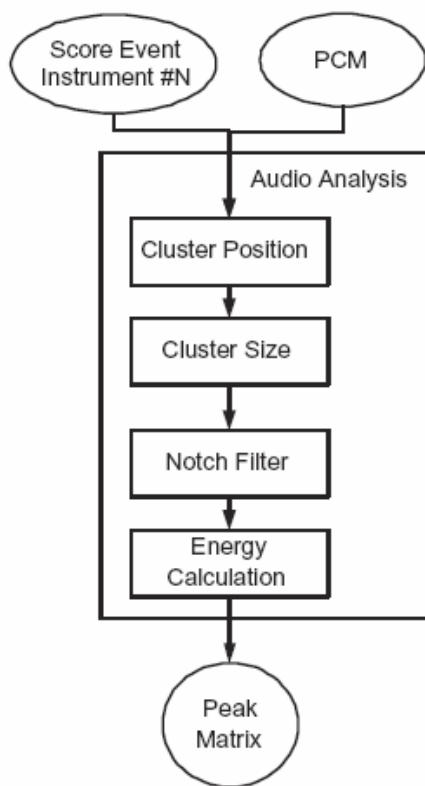


Figure 4: Audio analysis flowchart

Aim of this phase is the detection of each attack-time for each note codified in the selected instrument at the score level. For each verticalised score event (see Table 1), the audio signal is filtered with a notch filter centered both on the fundamental frequency and on the first four harmonics of the examined note. The portion of signal analyzed is calculated by defining a conveniently cluster position and size.

The cluster position is obtained as follow. Let  $T_{track}$  and  $T_{score}$  be the audio track duration in seconds and the MIDI track duration in MIDI ticks, respectively. Tick duration is computed as  $T_{tick} = T_{track} / T_{score}$ , while cluster position is determined as  $T_{cluster} = N_{tick} / T_{tick}$ , where  $N_{tick}$  is the sum of ticks associated to the previous notes.

The cluster size is calculated as  $Cluster\_size = (T_{track} / N_{events}) * variation$ , where  $N_{events}$  is the total event number extracted at score level; variation is an input parameter - related to the interpretation - which represents the amount of tempo variation of the audio execution with respect to the MIDI performance. Variation lies in a range from 1 to 4, where 2 means a not meaningful variation. In other words, this parameter allows increase or decrease of the cluster\_size in respect to the amount of expected tempo change.

After the notch filter, the energy envelope is calculated in order to generate the so-called *peak matrix*. After energy computation, its peaks are calculated in respect to score notes. Then, all peaks retrieved are stored into the peak matrix, as shown in Table 2. The first column contains the analyzed fundamental frequency; the matrix row number is equal to the 'number of events extracted at the score level'  $N_{events}$ . The second column contains a list of all possible peak times (in seconds). If no peak is retrieved, the matrix is filled with 0.

To be able to retrieve every note and avoid superimposition between adjacent semitones, the bandwidth of the notch filter depends on the analyzed frequency according to this formula:  $bandwidth = 3 * 2^{\log_{10}(f)}$ , where  $f$  is the frequency in Hertz.

C. Decisional phase

This is the last step of ParSi Algorithm, in which time-alignment between audio and score events is created (Fig. 5). As mentioned above, only score events belonging to a single instrument (as chosen by user) is considered for automatic synchronization.

TABLE II.  
AN EXAMPLE OF PEAK MATRIX

Fundamental Frequencies	Retrieved Peak Time
659	$T_1^1, T_2^1, \dots, T_n^1$
523	$T_1^2, T_2^2, \dots, T_n^2$
349	$T_1^3, T_2^3, \dots, T_n^3$
...	...

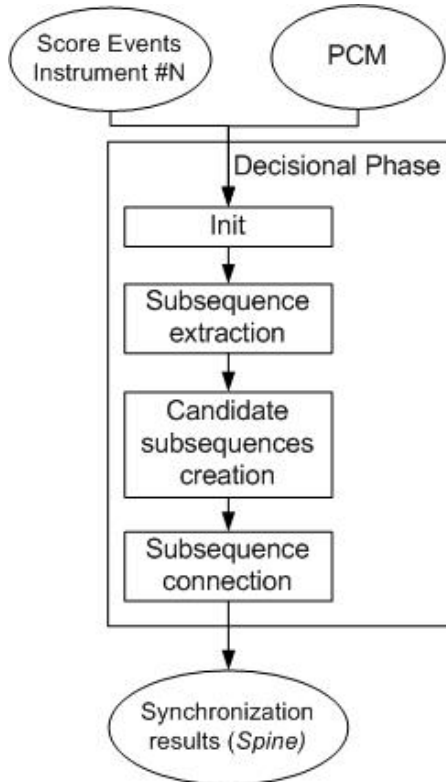


Figure 5: Decisional Phase flowchart

The **init** function is invoked first that initializes some variables used by the algorithm, such as  $SS_{size}$ ,  $N_{virtual\_peaks}$  and *Tolerance*:

- the maximum size of subsequence  $SS_{size}$  is provided as input by user. This value is used to define the subsequence limits where the algorithm will extract the candidate synchronization sequences
- $N_{virtual\_peaks}$  is the number of virtual peaks. They are inserted into the matrix peak when a 0 is found. To avoid bad alignment, this number must not exceed 35% of subsequence length  $SS_{size}$
- the *Tolerance* range is used to evaluate the reliability of retrieved event time at the audio level in respect to score event time in MIDI ticks. The default range lies between 0.4 and 0.8.

Subsequence region by means of the  $SS_{size}$  value is defined next, and the **subsequence extraction** function is invoked. The aim of this function is to find the most convenient values for *Tolerance* and  $N_{virtual\_peaks}$  to retrieve all those possibly valid synchronization events that will be inserted into the candidate subsequence. The following situations may arise:

- an acceptable subsequence is not found. In this case, the tolerance range is increased by one third:  $\Delta_{tolerance} = \Delta_{tolerance} + 1/3$ . If this is not sufficient, virtual peak number is increased as well.
- too many acceptable subsequences are found. In this case, the tolerance range is decreased by one third:  $\Delta_{tolerance} = \Delta_{tolerance} - 1/3$ .

If a sequence of chords is found, it is decomposed in all its

possible monody sequences.

The output of this function provides a matrix with two or three dimension that contains, for each note/chord belonging to the considered subsequence, the possible synchronization events which respect the tolerance. If the limit of *Tolerance* and  $N_{virtual\_peaks}$  is enforced and no acceptable subsequences are found, the matrix is filled with 0.

At the end of this phase, the **candidate subsequences creation** function is invoked. Its objective is to select the most reliable synchronization events given as output by **subsequence extraction** function. The candidate subsequence is found by selecting, for each note, the peak which has the lowest distance between the peak time-position in seconds retrieved at the audio level and the time-position in ticks calculated at the score level.

The final step is the **subsequence connection** where the final sequence is created (Table 3). Peak for peak, the final sequence is built linking together the candidate subsequences as follows:

- if there is a single candidate event greater than 0, and its value is greater than the previous, this value is inserted into the final sequence
- if there are two or more candidate events greater than 0, the arithmetic mean of these values is calculated
- if all candidate events are equal to 0, their value is fitted by linear interpolation.

The decisional phase generating the synchronization results codified into the IEEE 1599 spine terminates as illustrated in the next section.

#### IV. THE IEEE 1599 SPINE

In this section, the capabilities of IEEE 1599 to represent the synchronization anchors between audio and score are described. In this new IEEE standard, synchronization anchors are saved in three different layers: *Audio*, *Notational*, *Logic* (Fig. 6).

TABLE III.  
AN EXAMPLE OF SUBSEQUENCE CONNECTION

CS #1	CS #2	CS #3	CS #4	FS
0.00				0.00
0.50				0.50
1.15	0			1.15
1.91	0			1.91
	0			2.09
	0			2.66
	0	3.25		3.25
		3.54		3.54
		4.25	3.90	4.075
		4.60	4.32	4.46
			4.88	4.88



Figure 6: An example of Spine

The *track\_event* tag in the *Audio* layer has two attributes: *event\_ref*, which contains the identifier for the current event, and *start\_time*, which contains the related audio time.

Similarly, in the *Notational Layer*, the *graphic\_event* tag is present with five different attributes: *event\_ref* is the name of the event considered, while the other four attributes represent the position of the event in a graphic image. Element *spine* in the *Logic Layer* links graphic and track event, which have the same *event\_ref* attribute. Attributes *timings* and *hpos* have the following meaning:

*timing* represents the temporal co-ordinate of the event, while *hpos* refers to a virtual horizontal dimension. The score is viewed from an abstract vantage point (real scores are treated in the *Notational Layer*), and as a consequence staff systems never generate new lines. There is only one staff system, including all symbols on a single line, so that only the horizontal axis is used in establishing vertical alignment.

V. EXPERIMENTS AND RESULTS

A prototype of the synchronization algorithm has been implemented in C++, and this section describes experiments and results. The tests were run on an Intel Pentium IV, 2.6 GHz with 512 MByte RAM under Windows XP SP2.

For these tests, 31 pieces of music have been considered with have a length from 30 to 416 seconds. They can be grouped in:

- polyphonic and polytimbric MIDI pieces

- polyphonic piano pieces
- opera pieces, with human voice
- symphonic pieces, without human voice
- pop pieces.

A complete list can be found at the website <http://www.lim.dico.unimi.it/parsilist.html>.

The results given by ParSi Algorithm have been compared to manual synchronization of the same pieces. Objective evaluation is provided in term of percentage of **correct measures** (Fig. 7).

A *correct measure* is obtained if all synchronization events of the measure are correct. A synchronization event retrieved by ParSi Algorithm ( $se_i(\text{automatic})$ ) is correct if it does neither come before nor after the previous and the next synchronization events obtained by manual synchronization (respectively  $se_{i-1}(\text{manual})$  and  $se_{i+1}(\text{manual})$ ).

$$se_{i-1}(\text{manual}) < se_i(\text{automatic}) < se_{i+1}(\text{manual}) \quad (1)$$

It is possible to consider this criteria sufficient for applications where an accurate and precise synchronization is not required.

In order to obtain the best synchronization results, the instrument is selected to get as many score events as possible. Furthermore, when possible, the piano is chosen because, as state-of-the-art shows, it is generally the easiest to synchronize (see the section on Related Works).

Experiments show that ParSi Algorithm synchronizes correctly, namely at 96% - 100%, polyphonic and polytimbric MIDI pieces (piece numbered from 1 to 3).

ParSi synchronization of polyphonic piano pieces (numbered from 4 to 17) gives interesting results with a percentage of success between 60% and 95%.

Opera pieces, numbered from 18 to 22, have less synchronization precision, namely 35% - 60%, compared to piano pieces because of the higher complexity of the audio signal. A similar situation occurs with symphonic pieces, numbered from 23 to 27: the worst case is the 'Arabic Dance' from The Nutcracker Suite by Tchaikovski, number 23,

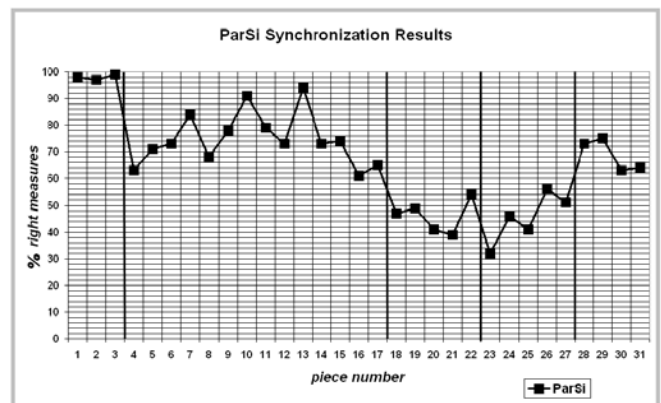


Figure 7: ParSi synchronization results

because of several repeated notes that generate similar time peaks.

Finally, pop music pieces, pieces numbered from 28 to 31, provide a promising success percentage, between 55% and 70%.

If we compare these results with other works that require a complete score ([7] [8] [9] [10] [11] [13]), it is possible to state that ParSi provides good automatic synchronization results with only a partial score representation.

## VI. CONCLUSIONS

In this work, an algorithm has been proposed to automatically synchronize different PCM audio performance with a partial music score codified in MIDI extracted from an IEEE 1599 file. It has been found that instrument selection is the most critical decision and deeply affects the quality of the results. An instrument with the largest number of events is recommended.

From an analysis of the experimental results, it appears that the ParSi approach is capable of providing interesting results even if only a partial score is available. The experiment with pop music is the most interesting because, more often than not, only incomplete scores are available. Experiments with opera and symphonic music show that synchronization results ought to be improved, and future work is planned to improve the algorithm and obtain a better time-alignment for such music genres.

As mentioned in [13], standard IEEE 1599 has been used to archive the synchronization results, and thanks to this standard they will be used again in future applications.

## REFERENCES

- [1] J.S. Downie, "Music information retrieval", chapter 7 *Blaise Cronin*, 2003, 37, 295-340.
- [2] R Typke, F Wiering, RC Veltkamp. A Survey of Music Information Retrieval Systems 6th International Conference on Music Information Retrieval, ISMIR 2005, 2005, 153-160.
- [3] Foote, J. An overview of audio information retrieval *Multimedia Systems, Springer-Verlag New York, Inc.*, 1999, 7, 2-10.
- [4] Clausen, M.; Kurth, F.; Müller, M. & Ribbrock, A. Content-based Retrieval in Digital Music Libraries *Proc. ECDDL, Bath, UK, Springer*, 2004.
- [5] Orio, N.L. Score Following: State of the Art and New Developments *Proceedings of the Conference on New Interfaces for Musical Expression*, 2003, 36-41.
- [6] Dannenberg, R.B. & Hu, N. Polyphonic Audio Matching for Score Following and Intelligent Audio Editors *Proceedings of the 2003 International Computer Music Conference*, 2003.
- [7] Arifi, V.; Clausen, M.; Kurth, F. & Muller, M. Automatic Synchronization of Music Data in Score-, MIDI- and PCM- Format *4th International Conference on Music Information Retrieval, ISMIR 2003*, 2003.
- [8] Hewlett, W. & E. Selfridge-Fields, e. (ed.) Automatic Synchronization of Musical Data: A Mathematical Approach *MIT Press*, 2004.
- [9] Dixon, S. & Widmer, G. MATCH: A Music Alignment Tool Chest *6th International Conference on Music Information Retrieval, ISMIR 2005*, 2005.
- [10] Muller, M.; Kurth, F. & Roder, T. Towards an Efficient Algorithm for Automatic Score-to-Audio Synchronization *5th International Conference on Music Information Retrieval, ISMIR 2004*, 2004.
- [11] Soulez, F.; Rodet, X. & Schwarz, D. Improving Polyphonic and Poly-Instrumental Music to Score Alignment *4th International Conference on Music Information Retrieval, ISMIR 2003*, 2003, 143-148.
- [12] Turetsky, R. & Ellis, D. Ground-truth transcriptions of real music from force-aligned midi syntheses *4th International Conference on Music Information Retrieval, ISMIR 2003*, 2003.
- [13] D'Aguanno, A. & Vercellesi, G. Automatic synchronisation between audio and score musical description layers *Semantic Multimedia, Springer-Verlag Berlin Heidelberg*, 2007, 4816, 200-210.
- [14] Rabiner, L.R. & Juang, B.H. Fundamentals of Speech Recognition *Prentice*, 1993.
- [15] Oppenheim, A. & Schaffer, R. Discrete-time signal processing *Prentice-Hall, Englewood Cliffs, USA*, 1989.

**Antonello D'Aguanno** is a member of the *Laboratorio di Informatica Musicale*, LIM, of the *Dipartimento di Informatica e Comunicazione*, DICO, Università degli Studi di Milano.

**Dr. Giancarlo Vercellesi** is a member of *Laboratorio di Informatica Musicale*, LIM, *Dipartimento di Informatica e Comunicazione*, DICO, Università degli Studi di Milano. His research focuses on analysis (MIR) and manipulation of signal audio in compressed domains, and he is currently studying methods to evaluate perceived audio quality from both a subjective and an objective approach.