Improving Polyphonic and Poly-Instrumental Music to Score Alignment

Ferréol Soulez
IRCAM – Centre Pompidou
1, place Igor–Stravinsky,
75004 Paris, France
soulez@ircam.fr

Xavier Rodet
IRCAM – Centre Pompidou
1, place Igor–Stravinsky,
75004 Paris, France
rod@ircam.fr

Diemo Schwarz
IRCAM – Centre Pompidou
1, place Igor–Stravinsky,
75004 Paris, France
schwarz@ircam.fr

Abstract

Music alignment links events in a score and points on the audio performance time axis. All the parts of a recording can be thus indexed according to score information. The automatic alignment presented in this paper is based on a dynamic time warping method. Local distances are computed using the signal's spectral features through an attack plus sustain note modeling. The method is applied to mixtures of harmonic sustained instruments, excluding percussion for the moment. Good alignment has been obtained for polyphony of up to five instruments. The method is robust for difficulties such as trills, vibratos and fast sequences. It provides an accurate indicator giving position of score interpretation errors and extra or forgotten notes. Implementation optimizations allow aligning long sound files in a relatively short time. Evaluation results have been obtained on piano jazz recordings.

1 Introduction

Score alignment means linking score information to an audio performance of this score. The studied signal is a digital recording of musicians interpreting the score. Alignment associates score information to points on the audio performance time axis. It is equivalent to a performance segmentation according to the score.

To do this, we propose a dynamic time warping (DTW) based methodology. Local distances are computed using spectral features of the signal, and an attack plus release note modeling (Orio & Schwarz, 2001). Very efficient on monophonic signals, this method can now cope with any poly-instrumental performance made up of less than five instruments without percussion.

After a brief overview of possible applications in section 1.1, the note model and DTW implementation are discussed in section 2. Finally, results obtained with this method are presented in section 3.

1.1 Applications, Goal and Requirements

Automatic score alignment has several applications. Each goal requires specific information from this automatic process. The most important applications are:

1. In applications that deal with symbolic notation, alignment can link this notation and a performance, allowing musicologists to work on a symbolic notation while listening to a real performance (Vinet, Herrera, & Pachet, 2002).
2. Indexing of continuous media through segmentation for content-based retrieval. The total alignment cost between pairs of documents can be considered as a distance measure (as in early works on speech recognition). This allows finding of the best matching documents from a database. These first two applications only need a good global precision and robustness.
3. Musicological comparison of different performances, studying expressive parameters and interpretation characteristics of a specific musician.
4. Construction of a new score describing exactly a selected performance by adding information such as dynamics, mix information, or lyrics. This information can be added to pitch and length labeling when building a database. Nevertheless re-transcription of tempo necessitates high time precision.
5. Performance segmentation into note samples automatically labeled and indexed in order to build a unit database, for example for data-driven concatenative synthesis based on unit selection (Schwarz, 2000, 2003a, 2003b) or model training (Orio & Déchelle, 2001). This segmentation requires a precise detection of the start and end of a note. However, notes that are known to be misaligned can be disregarded (see section 3.3).

Alignment is close to real time synchronization between a performer and a computer, known as score following (Orio & Déchelle, 2001; Orio, Lemouton, Schwarz, & Schnell, 2003). However, in alignment, the whole signal can be used and more accurate resolution can be obtained if required by the application. Nevertheless, alignment can be a good bootstrap procedure for training score followers which use statistical models.

For now, the goal of the present work is to obtain a correct global alignment, i.e. a precise pairing between notes present in the score and those present in the recording. On this basis, very precise estimation of the beginning and end of notes will be added in the future, as detailed in section 4.

1.2 Previous Work

Automatic alignment of sequences is a very popular research topic, especially in genetics, molecular biology and speech recognition. A good overview of this topic is (Rabiner & Juang, 1993). There are two main strategies: the oldest uses dynamic programming (DTW) and the other uses hidden Markov models (HMMs). For pairwise alignment of sequences, HMMs and DTW are quite interchangeable techniques (Durbin et al., 1998).

Concerning automatic alignment specifically, the main works are score following techniques tuned for off line use (Raphael, 1999), the previous work of (Orio & Schwarz, 2001), or (Meron, 1999). A different approach of music alignment is very briefly described in (Turetsky, 2003). All of these techniques consider mainly monophonic recordings.

For note recognition, there are many pitch detection techniques using signal spectrum or auto-correlation, for instance. These techniques are often efficient in monophonic cases but none of these use score information and are therefore sub-optimal in our situation.

1.3 Principle

Score alignment is performed in four stages:

- First, construction of the score representation by parsing of the MIDI file into score events.
- Second, extraction of audio features from signal.
- Third, calculation of local distances between score and performance.
- Fourth, computation of the optimal alignment path which minimizes the global distance.

This last stage is carried out using DTW. Our choice for this algorithm is due to the possibility of optimizing memory requirements. Also, unlike HMMs, DTW does not have to be trained, so that a hand made training database is not necessary.

2 The Method

For each sequence, the score and the performance are divided into frames described by features. Score information is extracted from standard MIDI files, the format of most of the available score databases. However this format is very heterogeneous and does not contain all classical score symbols. The only available features from these MIDI files are the fundamental frequencies present at any time, and note attack and end positions. As implicitly introduced in (Orio & Schwarz, 2001), the result of the score parsing is a time-ordered sequence of *score events* at every change of polyphony, i.e. at each note start and end, as exemplified in figure 1.

The features of the performance are extracted through signal analysis techniques using short time Fourier transformation (usually with a 4096 points hamming window, 93 ms at 44.1 kHz). The temporal resolution needed for the alignment determines the hop size of frames in the performance. The score is then divided into approximately the same number of frames as the performance. In consequence, the global alignment path should follow approximately the diagonal of the local distance matrix (see section 2.2).

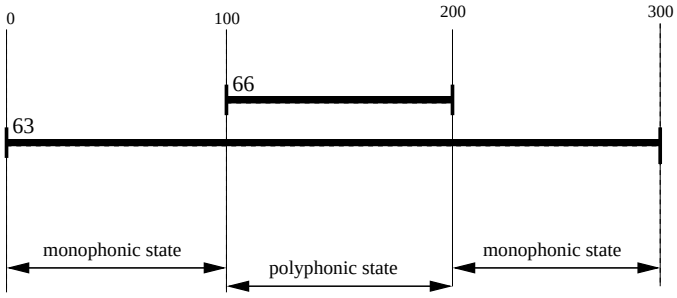


Figure 1: Parsing of a MIDI score into score events and the states between them.

Finally, DTW finds the best alignment based on local distances using a Viterbi path finding algorithm which minimizes the global distance between the sequences.

2.1 Model: Local Distance Computation

The local distance is calculated for each pair made up of a frame m in the performance and a frame n in the score. This distance, representing the similarity of the performance frame m to the score frame n, is calculated using spectral information. The local distances are stored in the local distance matrix $ldm(m, n)$.

The only significant features contained in the score are the pitch, the note limits and the instrument. Since having a good instrument model is difficult, only pitch and transients were chosen as features for the performance. This is why the note model is defined with attack frames using pitch and onset information, and sustain frames using only pitch.

2.1.1 Sustain Model

The sustain model uses only pitch. As pitch tracking algorithms are error prone, especially for polyphonic signals, a method called *Peak Structure Match* (Orio & Schwarz, 2001) is used. With this method, the local *Peak Structure Distance* (*PSD*) is the ratio of the signal energy filtered by harmonic band pass filters corresponding to each expected pitch present in the score frame, over total energy.

This technique is very efficient in monophonic cases. However in the poly-instrumental situation, the different instruments do not have the same loudness, and it is very difficult to localize low and short notes under continuous loud notes. Coding energies on a logarithmic scale reduces level ratio between the different instruments and thus improves results.

However, this model has two major drawbacks. First, in polyphonic cases, filter banks corresponding to a chord tend to cover the major part of the signal spectrum, increasing the likeness of this chord with any part of the performance. As result, filters need to be as precise as possible.

Secondly, such a model with narrow filters is adapted to fixed pitch instruments, such as the piano, in which small frequency variations, error, or vibrato, are impossible. For string instru-

ments and the voice, such variations can be as large as a semi tone around the nominal frequency of the note. A simple solution is to define vibrato as a chord of the upper and the lower frequency, but vibrato is not included in most MIDI based scores. Another solution is to give a degree of freedom to each filter around its nominal frequency. For each performance frame, the filter is tuned within a certain range to yield the highest energy. The energy is weighted by a Gaussian window centered on the nominal frequency of the filter, lowering the preference for a high energy peak far away and favoring a low but close one. Amazingly, we have observed that shifting filters independently gives better results than shifting the whole harmonic comb.

Moreover, this filter tolerance improves distance calculation for slightly inharmonic instruments. After a number of tests, working with the first F_n = 6 harmonics filters gives acceptable results. Equivalent results were obtained for F_n = 7 or 8. The best and most homogeneous results are obtained with a filter width of $\frac{1}{10}^{\text{th}}$ semitone (10 cents) and a tolerance of about $\frac{3}{4}^{\text{th}}$ semitones (75 cents around the nominal frequency.

2.1.2 Attack Model

Tests using only the sustain model show some imprecision of the alignment marks, which are often late. Worse, in very polyphonic cases (more than three simultaneous notes), some notes are not detected at all.

There are two reasons for the markers' imprecision. First, the partials' reverberation of the previous notes is still present during the beginning of the next one. Second, during attacks, energy is often spread all over the spectrum and the energy maximum in the filters is reached several frames after the true attack. With the sustain model alone, alignment marks are set at the instant when the energy of the current note rises above the energy of the last note, several hundredths of a second after the true onset.

Moreover, in the polyphonic case, during chords, several notes often have common partials. If only one note of this chord changes, too few partials may vary to cause enough difference in the spectral structure to be detectable by the *PSD*.

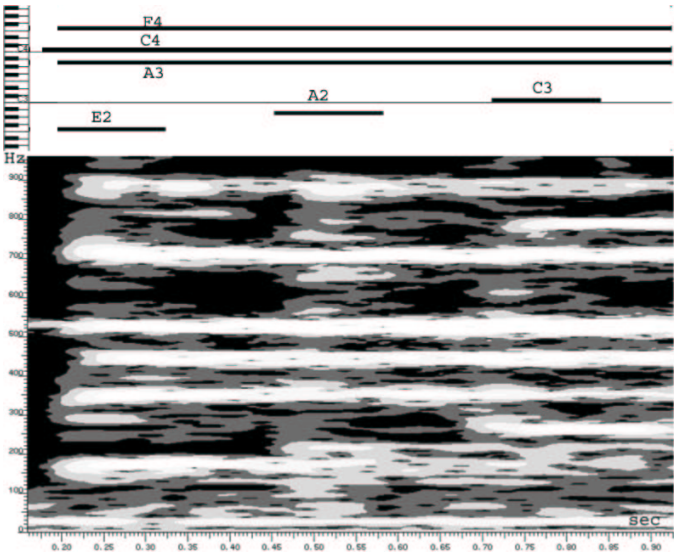
A more accurate indication of a note beginning is given by the variation in the filters. Thus, special score frames using energy variations Δ_i^k in the harmonic filter band i of the note k instead of *PSD* were created at every onset. In these frames, the attack distance AD is given by the sum of the energy variations (in dB) in every tuned filter band i. In the case of simultaneous onsets, the distance AD is computed for every beginning note and averaged out:

$$AD = \text{mean}_k \left(1 - \tanh\left(\alpha\left(\sum_{i=1}^{F_n} \|\Delta_i^k\| - \theta_a\right)\right)\right) \quad (1)$$

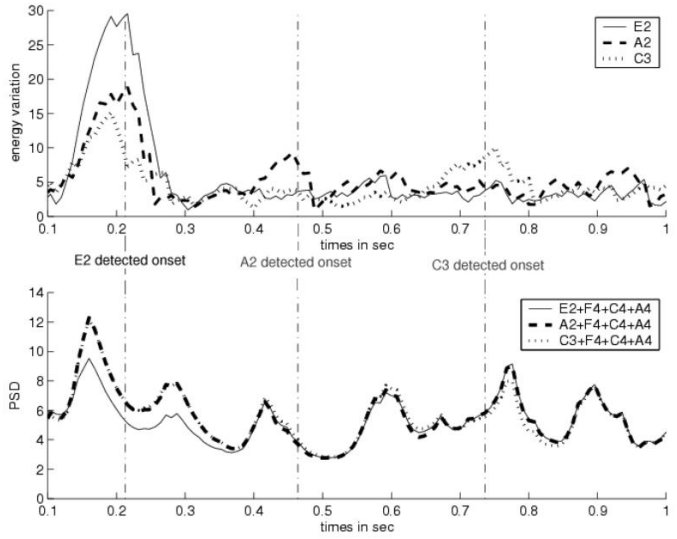
with Δ_i^k the energy difference in dB with the precedent local extremum in the filter band i of note k, θ_a a threshold, and α a scaling factor.

Small note changes during chords seem to be grasped by human perception mostly due to their onsets. Therefore, the local distance AD is amplified by the scaling factor α to favor onset detection over *PSD*. After carrying out some tests, θ_a was set to 6.5 dB and α to 50.

The example in figure 2 is characteristic of the principal problems of the sustain detection: For the first second of this Mozart string and oboe quartet, violins and oboe play a loud continuous note while the cello is playing small notes in their subharmonics. The cello has many common partials with the other notes and global energy variations are due to violin vibrato and not cello onset. As shown by the *PSD* diagram in figure 2(b), detection by use of the sustain model (*PSD*) is not possible. On the contrary, the three notes E2, A2 and C3 can easily be localized on the energy variation diagram as indicated by the vertical dash-dotted lines.



(a) Spectrogram and MIDI roll



(b) $\sum_{i=1}^{F_n} \|\Delta_i\|$ and *PSD* for note E2 A2 C3

Figure 2: First second of Mozart quartet

2.1.3 Silence Model

Short silences due to short rests in the score and non-legato playing are difficult to model, since reverberation has to be taken into an account. We only model rests longer than 100 ms.

Shorter rests are merged with the previous note. The local distance SD for long rests is computed using an energy threshold θ_s:

$$SD(m,n) = \begin{cases} E - \theta_s & \text{if } E \geq \theta_s, \\ 0 & \text{if } E < \theta_s. \end{cases} \quad (2)$$

where E is the energy of the signal in the performance frame m.

2.2 Dynamic Time Warping

DTW is a consolidated technique for the alignment of sequences, the reader may refer to (Rabiner & Juang, 1993) for a tutorial. Using dynamic programming, DTW finds the best alignment between two sequences according to a number of constraints. The alignment is given in the form of a path in a local distance matrix where each value $ldm(m,n)$ is the likeness between the score frame m and the performance frame n. If a path goes through $[m,n]$, the frame m of the performance is aligned with frame n of the score. The following constraints have been applied: The end points are set to be $[1,1]$ and $[M,N]$, where M and N are the number of frames of the performance and of the score, respectively. The path is monotonic in both dimensions. The score is stretched to approximately the same duration as the performance ($M \approx N$). The optimal path should then be close to the diagonal, so that favoring the diagonal would prevent deviating paths.

Three different local neighborhoods of the DTW have been tested. Several improvements have been added to the classical DTW algorithm in order to lower processing time or memory requirements and thus allow long performances to be analyzed. The most important of these improvements are the path pruning and the short cut path implementation.

2.2.1 Local Constraints

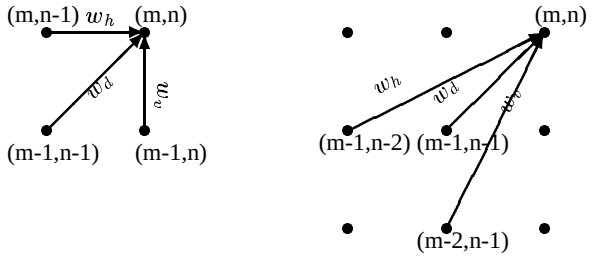
The DTW algorithm calculates first the augmented distance matrix $adm(m,n)$ which is the cost of the best path up to the point $[m,n]$. To compute this adm matrix, different types of local constraints have been implemented in which the weights along the local path constraint branches can be tuned in order to favor one direction. These weights [w_v w_h w_d] are explained in the figure 3. The different type names, I, III and V follow the notation in (Rabiner & Juang, 1993) and are calculated as follows, with $ldm(m,n)$ abbreviated to λ:

Type I :

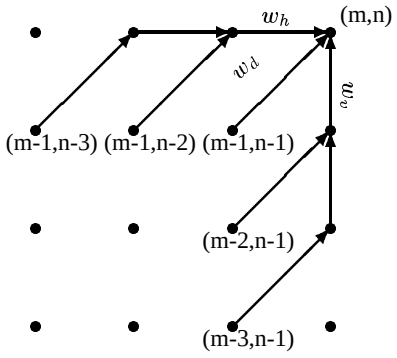
$$adm(m,n) = \min \begin{Bmatrix} adm(m-1,n-1) + w_d\,\lambda \\ adm(m-1,n) + w_h\,\lambda \\ adm(m,n-1) + w_v\,\lambda \end{Bmatrix} \quad (3a)$$

Type III :

$$adm(m,n) = \min \begin{Bmatrix} adm(m-1,n-1) + w_d\,\lambda \\ adm(m-2,n-1) + w_v\,\lambda \\ adm(m-1,n-2) + w_h\,\lambda \end{Bmatrix} \quad (3b)$$



Type I Type III



Type V

Figure 3: Neighborhood on point (m,n) in type I, III and V

Type V :

$$adm(m,n) = \min \begin{Bmatrix} adm(m-1,n-1) + w_d\,\lambda \\ adm(m-2,n-1) + w_v\,\lambda \\ \quad + w_d\,ldm(m-1,n) \\ adm(m-1,n-2) + w_h\,\lambda \\ \quad + w_d\,ldm(m,n-1) \\ adm(m-3,n-1) + w_v\,\lambda \\ \quad + w_d\,ldm(m-2,n) \\ \quad + w_v\,ldm(m-1,n) \\ adm(m-1,n-3) + w_h\,\lambda \\ \quad + w_h\,ldm(m,n-1) \\ \quad + w_d\,ldm(m,n-2) \end{Bmatrix} \quad (3c)$$

The constraint type I is the only one allowing horizontal or vertical paths and thus admitting extra or forgotten notes. Since it allows for vertical or horizontal paths, the drawback of this constraint type is as follows: The path can be stuck in a frame of a given axis with erroneous small local distance with successive frames of the other axis. It leads to bad results in the polyphonic case by detecting too many extra or forgotten notes.

The types III and V constrain the slope to be respectively between 2 and $\frac{1}{2}$ or 3 and $\frac{1}{3}$. Since it is very rare to hear a performance with passages played more than three times faster or

slower than the score, it gives good alignment but will accept neither vertical nor horizontal paths and thus does not directly handle forgotten or extra notes. These constraints III and V give approximately the same result, the type V takes more resources and more time but gives more freedom to the path allowing greater slope. Using Type V is preferable but type III can still be used for long pieces.

The standard values for the local path constraints [w_v w_h w_d] = [1 1 2] for type I and V or [3 3 2] for type III, do not favor any direction and are used in our method. Note that our experiments showed that lowering w_d favors the diagonal and prevents extreme slopes.

2.2.2 Path Pruning

As the frame size is usually around 5.8 ms, three minute long performances contain about 36000 frames, so that about $1.3 \cdot 10^9$ elements need to be computed in the local distance matrix and as many for the augmented distance matrix. The memory required to store them is 2.5 GB. To reduce the computation time and the resources needed, at every iteration m, only the best paths are kept, by pruning the paths with an augmented distance $adm(m,n)$ over a threshold θ_P. This threshold is dynamically set using the minimum of the previous adm row. After various experiments this threshold was set to:

$$\theta_P(m) = 1.1 \min\left(adm(m-1)\right) \quad (4)$$

However, the paths between the corridor of selected paths and the diagonal are not pruned to leave more possible paths. Usually the corridor width is about 400 frames.

2.2.3 Shortcut Path

Most applications only need to know the note start and end points, and not the alignment within the note. Therefore, only a shortcut path, linking all the score events in the path, is stored as explained in (Orio & Schwarz, 2001). As the local constraint types III or V need computation with a depth of 3 or 4 frames respectively, only 2 or 3 frames per performance frame are stored for each score event reducing memory requirements by about 95%.

3 Results

All tests were performed with a default frame hop size of 5.8 ms (usually 256 points) which is a good compromise between precision and number of frames to compute. This hop-size can be lower for a better resolution when considering small recordings or higher for quick preview of the alignment.

Due to the absence of previously aligned databases and the difficulty of building one by human alignment, quantitative statistics were done on a small database. However, many qualitative tests were performed by listening to performances and their reconstituted MIDI files, which permitted the evaluation of global alignment. These tests were performed with various types of music (classical, contemporary, songs without percussion, for instance Bach, Chopin, Boulez, Brassens, etc.) achieving very good results. Even with difficult signals such as voices, very fast violin or piano sections, trills, vibrato, poly-instrumental pieces, the algorithm showed good results and good robustness with only few imprecisions on onset for multi-instrumental pieces.

3.1 Limits

Notes shorter than 4 frames (23 ms) are very difficult to detect and often lead to errors for neighbor notes. Therefore, all the events that are too short, are merged in a chord with the next event. This technique makes it possible to handle unquantised chords from MIDI files recorded on a keyboard. Alignment is efficient for pieces with less than five harmonic instruments such as singing voice, violin, piano, etc. As the memory requirement is still too high, only pieces shorter than six minutes and with about four thousand or less score events are currently treatable (a little less with local constraint V), but this is enough to align most pieces. The longest successful test was performed on a five minute and twelve second long jazz performance of 4200 score events with time resolution of 5.8 ms (53926 frames) taking about 400 MB of RAM and 146 minutes on a Pentium IV 2.8 GHz running C++ and Matlab® routines.

3.2 Automatic Evaluation

As performers rarely play with sudden variations in tempo, extreme slopes of alignment path, with large variation, usually indicate score–performance mismatching. Thus, the path slope can be a good error indicator. If the slope is $\frac{1}{3}$ for several notes, it is very likely that some notes are missing in the performance. On the other hand if the slope is 3, there are certainly extra notes in it.

This indicator was able to find with precision the position of an unknown extra measure in a score of Bach's prelude, as can be seen in figure 4.

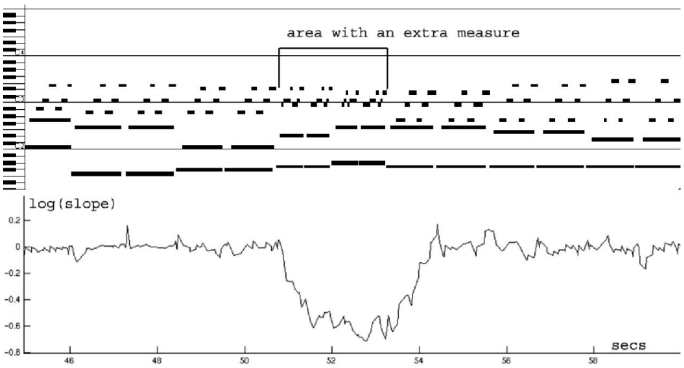


Figure 4: Piano roll representation of aligned MIDI, and path slope in log units in the Bach's prelude between 45 sec and 60 sec.

3.3 Robustness

Tests with audio recording that do not exactly coincide with the MIDI files showed very strong robustness and a very good global alignment. For instance, alignment of the first prelude for piano of Bach (80 sec and 629 score events) with an extra measure at the 51st second was correctly aligned until the 50th and after the 55th, and another test with a Bach sonata for violin showed a very good global alignment even though a passage of 52 notes was missing in the score!

Vibratos and trills can be aligned very efficiently as well, as shown in the very large vibrato section of *Anthèmes 2* by Boulez.

3.4 Error Rate

Quantitative tests were performed on several jazz piano improvisations played by 3 different pianists, where sound and MIDI were both recorded. These are very fast (an attack every 70 ms on the average) and long pieces (about four minutes) with many trills and a wide dynamical range.

As reverberation prevents precise note end determination, we focused on note onset detection. Only a good global alignment was looked for. A correct pairing between score and performance means that the detected note onset is closer to its corresponding onset in the performance than any other. With this criterion, tests showed a 9.7% error rate of onset detection over the 9024 considered onsets, about 65% of these errors were made on notes shorter than 80 ms, corresponding to a rate of 12 notes per second. These results need several comments:

1. Due to the MIDI recording system used, the MIDI file, though recorded from the keyboard simultaneously with the audio seems to be relatively imprecise when compared to the audio.
2. During the MIDI parsing, every note shorter than 4 frames (usually 23 ms) is merged with the preceding note, increasing error rate of small notes (numerous in our tests).
3. The hop size gives 5.8 ms maximum resolution between each possible detection.
4. Finally, as audio features are extracted from a short time fast Fourier transform computed on a 93 ms (4096 points) window, the center of this window is taken to determine frame position in the recording. A better solution would be to take the center of gravity of energy in this window, but this function is not yet implemented.

As a consequence, tests showed a 23.8 ms standard deviation between the score onset and the detected one. This result can easily be improved in the near future, by a second stage of precise time alignment within the vicinity of the alignment mark. The precise alignment was not the goal pursued in this present work.

4 Conclusion and Future Work

Our method, which is being used at IRCAM for research in musicology, can efficiently perform alignment on difficult signals such as multi-instrumental music (of less than five instruments), trills, vibrato, accentuated or fast sequences, with an acceptable error rate.

We are currently working on an onset detector which reanalyzes the signal around the alignment mark, thus improving the resolution for applications which need better precision. Furthermore, a percussion detection process is being worked on to be included soon in the alignment process.

One of the fundamental problems remaining is the inadequacy of the score representation. MIDI files contain very little information compared to real musical scores and so too few features can be used in the alignment.

Acknowledgments

Many thanks to E. Vincent who was a precious adviser during the preparation of this article.

References

Durbin, R., et al.. (1998). *Biological sequence analysis: Probabilistic models of proteins and nucleic acids.* Cambridge University Press.

Meron, Y. (1999). *High quality singing synthesis using the selection-based synthesis scheme.* Unpublished doctoral dissertation, University of Tokyo.

Orio, N., & Déchelle, F. (2001). Score Following Using Spectral Analysis and Hidden Markov Models. In *Proceedings of the International Computer Music Conference (ICMC).* Havana, Cuba.

Orio, N., Lemouton, S., Schwarz, D., & Schnell, N. (2003). Score Following: State of the Art and New Developments. In *Proceedings of the international conference on new interfaces for musical expression (nime).* Montreal, Canada.

Orio, N., & Schwarz, D. (2001). Alignment of Monophonic and Polyphonic Music to a Score. In *Proceedings of the International Computer Music Conference (ICMC).* Havana, Cuba.

Rabiner, L. R., & Juang, B.-H. (1993). *Fundamentals of speech recognition.* Englewood Cliffs, NJ: Prentice Hall.

Raphael, C. (1999). Automatic Segmentation of Acoustic Musical Signals Using Hidden Markov Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 21*(4), 360–370.

Schwarz, D. (2000). A System for Data-Driven Concatenative Sound Synthesis. In *Digital Audio Effects (DAFx)* (pp. 97–102). Verona, Italy.

Schwarz, D. (2003a). New Developments in Data-Driven Concatenative Sound Synthesis. In *Proceedings of the International Computer Music Conference (ICMC).* Singapore.

Schwarz, D. (2003b). The CATERPILLAR System for Data-Driven Concatenative Sound Synthesis. In *Digital Audio Effects (DAFx).* London, UK.

Shalev-Shwartz, S., Dubnov, S., Friedman, N., & Singer, Y. (2002). Robust temporal and spectral modeling for query by melody. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 331–338). ACM Press.

Turetsky, R. (2003). *MIDIAlign: You did* what *with MIDI?* Retrieved August 8, 2003, from `http://www.ee.columbia.edu/~rob/midialign`.

Vinet, H., Herrera, P., & Pachet, F. (2002). The Cuidado Project: New Applications Based on Audio and Music Content Description. In *Proceedings of the International Computer Music Conference (ICMC).* Gothenburg, Sweden.