Towards an Efficient Algorithm for Automatic Score-to-Audio Synchronization

Conference Paper · October 2004

CITATIONS
68

READS
236

3 authors, including:

Meinard Müller
Friedrich-Alexander-Universität of Erlangen-Nürnberg
377 PUBLICATIONS **11,362** CITATIONS
SEE PROFILE

Tido Röder
12 PUBLICATIONS **1,286** CITATIONS
SEE PROFILE

TOWARDS AN EFFICIENT ALGORITHM FOR AUTOMATIC SCORE-TO-AUDIO SYNCHRONIZATION

Meinard Müller, Frank Kurth, Tido Röder
Universität Bonn, Institut für Informatik III
Römerstr. 164, D-53117 Bonn, Germany
{meinard, kurth, roedert}@cs.uni-bonn.de

ABSTRACT

In the last few years, several algorithms for the automatic alignment of audio and score data corresponding to the same piece of music have been proposed. Among the major drawbacks to these approaches are the long running times as well as the large memory requirements. In this paper we present an algorithm, which solves the synchronization problem accurately and efficiently for complex, polyphonic piano music. In a first step, we extract from the audio data stream a set of highly expressive features encoding note onset candidates separately for all pitches. This makes computations efficient since only a small number of such features is sufficient to solve the synchronization task. Based on a suitable matching model, the best match between the score and the feature parameters is computed by dynamic programming (DP). To further cut down the computational cost in the synchronization process, we introduce the concept of anchor matches, matches which can be easily established. Then the DP-based technique is locally applied between adjacent anchor matches. Evaluation results have been obtained on complex polyphonic piano pieces including Chopin's Etudes Op. 10.

1. INTRODUCTION

Modern digital music libraries consist of large document collections containing music data of diverse characteristics and formats. For a single piece of music, the library may contain the musical score, several compact disc recordings of a performance, and various MIDI files. Inhomogeneity and complexity of such music data make content-based browsing and retrieval in digital music libraries a difficult task with many yet unsolved problems. Here, synchronization algorithms which automatically link data streams of different data formats representing a similar kind of information are of great importance. In this paper, we consider the fundamental case that one data stream, given as a MIDI file, represents the score of a piece of music and the other data stream, given as a WAV file, represents a recorded performance of the same piece of music. The latter data is also simply referred to as *audio*. The *synchronization task* then amounts to associating the note events given by the score data stream with their occurrences in the audio file.

Score and audio data fundamentally differ in their respective structure and content, making score-to-audio synchronization a difficult task. On the one hand, the score consists of note parameters such as pitch, onset times, or note durations, leaving a lot of space for various interpretations concerning, e. g., the tempo, dynamics, or multi-note executions such as trills. On the other hand, the waveform-based CD recording of some performance encodes all the information needed to reproduce the acoustic realization — note parameters, however, are not given explicitly. Therefore, all present approaches to score-to-audio synchronization (see Section 2) proceed in two stages: In the first stage, suitable parameters are extracted from the score and audio data streams making them comparable. In the second stage, an optimal alignment is computed by means of dynamic programming (DP) based on a suitable local distance measure.

The approach discussed in this paper also follows along these lines. However, we put special emphasis on the efficiency of the involved algorithms — concerning running time as well as memory requirements. In contrast to previous approaches, we use a sparse set of highly expressive features, which can be efficiently extracted from the audio data stream. Due to its expressiveness, this feature set allows for an accurate synchronization with high time resolution (around 20 ms). Due to its sparseness, it facilitates a time and memory efficient alignment procedure (based on 0 to 20 feature vectors per second depending on the respective segment of the piece of music). In our research we have concentrated on polyphonic piano music of arbitrary genre and complexity. This allows us to exploit certain characteristics of the piano sound to be used in the feature extraction. Our underlying concept, however, may be transferred to other music as well by modifying the feature set.

In the second stage, we use dynamic programming (DP) to compute the actual score-to-audio alignment. Our suggested matching model differs from the the classical concept of dynamic time warping (DTW) employed in the

synchronization algorithms suggested in [6, 7]. Since we prefer to have missing matches over having bad or wrong matches, we do not force the alignment of all score notes but rather allow note objects to remain unmatched. Furthermore, we present efficiently computable local score functions which relate the audio features to the note parameters of the score data. Here we are led by the following simple but far-reaching principle: The score data will guide us in what to look for in the audio data stream. In other words, all information contained in the extracted audio features, which is not reflected by the score data, remains unconsidered by the local score function.

As for classical DTW, the running time as well as the memory requirements in the second stage are proportional to the product of the lengths of the two sequences to be aligned. In view of efficiency it is therefore important to have sparse feature sets leading to short sequences. The synchronization algorithm can be accelerated considerably if one knows matches prior to the actual DP computation. To account for such kind of prior knowledge, we introduce the notion of *anchor configurations* which may be thought of as note objects having some salient dynamic or spectral properties, e. g., some isolated fortissimo chord with some salient harmonic structure or some long pause. The counterparts of such note objects in the audio data streams can be determined by a linear-time linear-space algorithm which efficiently provides us with so-called anchor matches. The remaining matches can then be computed by much shorter, local DP computations between these anchor matches.

The rest of this paper is organized as follows. After a brief overview of related approaches in Section 2, we describe the two stages: the feature extraction in Section 3 and the alignment procedure in Section 4. Then, in Section 5, we describe how to improve the efficiency of our synchronization by introducing the concept of anchor matches. The synchronization results as well as the running time behavior of our algorithm for complex polyphonic piano pieces including Chopin's Etudes Op. 10 are presented in Section 6. Section 7 concludes the paper with a summary and perspectives on future work.

2. RELATED WORK

There are several problems in computational musicology which are related to the synchronization problem such as automatic score following, automatic music accompaniment, performance segmentation or music transcription. Due to space limitations the reader is referred to [1, 6] for a discussion and links to the literature. There, one also finds a description of other conceivable applications of score-to-audio alignment. We now summarize recent approaches from the relatively new field of automatic score-to-audio synchronization as described in [1, 6, 7].

All three approaches proceed in the two stages mentioned above. Turetsky et al. [7] first convert the score data (given in MIDI format) into an audio data stream using a synthesizer. Then, the two audio data streams are analyzed by means of a short-time Fourier transform (STFT) which in turn yields a sequence of suitable feature vectors. Based on an adequate local distance measure permitting pairwise comparison of these feature vectors, the best alignment is derived by means of DTW.

The approach of Soulez et al. [6] is similar to [7] with one fundamental difference: In [7], the score data is first converted into the much more complex audio format — in the actual synchronization step the explicit knowledge of note parameters is not used. Contrary, Soulez et al. [6] explicitly use note parameters such as onset times and pitches to generate a sequence of attack, sustain and silence models which are used in the synchronization process. This results in a more robust algorithm with respect to local time deviations and small spectral variations.

Since the STFT is used for the analysis of the audio data stream, both approaches have the following drawbacks: Firstly, the STFT computes spectral coefficients which are *linearly* spread over the spectrum resulting in a bad low-frequency resolution. Therefore, one has to rely on the harmonics in the case of low notes. This is problematic in polyphonic music where harmonics and fundamental frequencies of different notes often coincide. Secondly, in order to obtain a sufficient time resolution one has to work with a relatively large number of feature vectors on the audio side. (For example, even with a rough time resolution of 46 ms as suggested in [7] more than 20 feature vectors per second are required.) This leads to huge memory requirements as well as long running times in the DTW computation.

In the approach of Arifi et al. [1], note parameters such as onset times and pitches are extracted from the audio data stream (piano music). The alignment process is then performed in the score-like domain by means of a suitably designed cost measure on the note level. Due to the expressiveness of such note parameters only a small number of features is sufficient to solve the synchronization task, allowing for a more efficient alignment. One major drawback of this approach is that the extraction of score-like note parameters from the audio data — a kind of music transcription — constitutes a difficult and time-consuming problem, possibly leading to many faultily extracted audio features. This makes the subsequent alignment step a delicate task.

3. COMPUTATION OF SPARSE FEATURE SETS

Before we describe the first stage, the extraction step, it is helpful to recall some facts concerning the music to be aligned. As mentioned before, we consider polyphonic piano music of any genre and complexity. This allows us to exploit certain characteristics of the piano sound. However, dealing with piano music is still a difficult task due to the following facts (see, e. g., [2, 3] for more details):

- Striking a single piano key already generates a complex sound consisting not only of the fundamental pitch and several harmonics but also comprising in-

harmonicities caused by the keystroke (mechanical noise) as well as transient and resonance effects.

- Especially due to the usage of the right (sustaining) pedal, the note lengths in piano performances may differ considerably from the note lengths specified by the score. This results in complex sounds in polyphonic music which are not reflected by the score. Furthermore, pedaling also has a great effect on the timbre (sound spectrum) of a piano sound.

- The piano has a large pitch range as well as dynamic range. The respective sound spectra are not just translated, scaled, or amplified versions of each other but differ fundamentally in their respective structure depending on pitch and velocity.

To make the alignment robust under such spectral, dynamic and temporal variations, we only consider pitch and onset information for our audio features. The extraction of such features is based on the following fact: Striking a piano key results in a sudden energy increase (attack phase). This energy increase may not be significant relative to the entire energy — in particular if the keystroke is soft and the generated sound is masked by the remaining signal. However, the energy increase relative to the spectral bands corresponding to the fundamental pitch and harmonics of the respective key may still be substantial. This observation suggests the following general feature extraction procedure (cf. [8], for a similar approach):

- First decompose the audio signal into spectral bands corresponding to the fundamental pitches and harmonics of all possible piano keys.

- Then compute the positions of significant energy increases for each band. These positions constitute candidates for note onsets.

Note that, opposed to the approach in [1], we do not try to extract further note parameters from the audio file. The alignment will purely be based on these onset candidates.

We now describe our feature extraction in detail. For convenience, we identify the notes A0 to C8 of a standard piano with the MIDI pitches $p = 21$ to $p = 108$. For example, we speak of the note A4 (frequency 440 Hz) and simply write $p = 69$. Besides the fundamental pitch of a note p, we also consider the first two harmonics, which can be approximated by the pitches $p+12$ and $p+19$. The generalization of our concepts to a constant or variable, note-dependent number of higher harmonics is straightforward (cf. Section 4.2).

3.1. Subband Decomposition

In decomposing the audio signal we use a filter bank consisting of 88 bands corresponding to the piano notes $p = 21$ to $p = 108$. Since a good signal analysis is the basis for our further procedure, the imposed filter requirements are stringent: To properly separate adjacent notes, the passbands of the filters should be narrow, the cutoffs should be sharp, and the rejection in the stopband should be high. In addition, the filter orders should be small to allow for efficient computation. In order to design a set of filters satisfying these requirements for all MIDI notes in question, we work with three different sampling rates: 22050 Hz for high frequencies ($p = 93, \ldots, 108$), 4410 Hz for medium frequencies ($p = 57, \ldots, 92$), and 882 Hz for low frequencies ($p = 21, \ldots, 56$). Each filter is implemented using an eighth-order elliptic filter with 1 dB passband ripple and 50 dB rejection in the stopband. To separate the notes we use a Q factor (ratio of center frequency to bandwidth) of $Q = 25$ and a transition band half the width of the passband. Figure 1 shows the magnitude response of some of these filters.

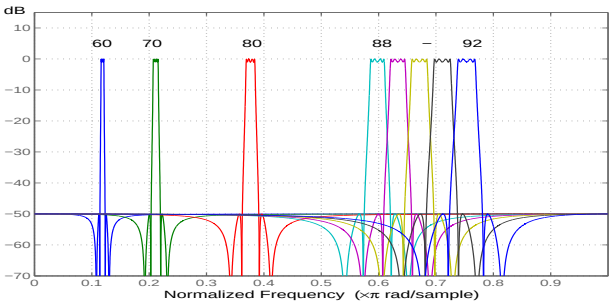

Figure 1. Magnitude responses for the elliptic filters corresponding to the MIDI notes 60, 70, 80, and 88 to 92 (sampling rate 4410 Hz).

Elliptic filters have excellent cutoff properties as well as low filter orders. However, these properties are at the expense of large phase distortions and group delays. Since in our offline scenario the audio signals are entirely known prior to computations, one can apply the following trick: After filtering in the forward direction, the filtered signal is reversed and run back through the filter. The resulting output signal has precisely zero phase distortion and a magnitude modified by the square of the filter's magnitude response. Further details may be found in standard text books on digital signal processing such as [5].

We have found this filter bank to be robust enough to work for a reasonably tuned piano. For out-of-tune pianos one may easily adjust the center frequencies and bandwidths as suggested in [8].

3.2. Onset Detection

After filtering the audio signal, we compute the short-time root-mean-square (STRMS) power for each of the 88 subbands. To this end, we convolve each squared subband signal with a Hann window of suitable length. In our experiments, we picked the three different window sizes of 101, 41, and 21 samples depending on the sampling rates 22050, 4410, and 882 Hz, respectively. The resulting curves are further lowpass-filtered and downsampled by factors 50, 10, and 10, respectively. Finally, the first-order difference function is calculated and half-wave rectified (i. e., taking only the positive part of the function).

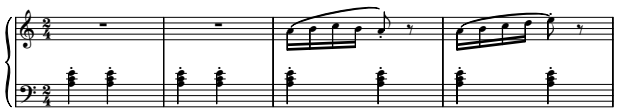

Figure 2. First four measures of Op. 100, No. 2 by Friedrich Burgmüller.

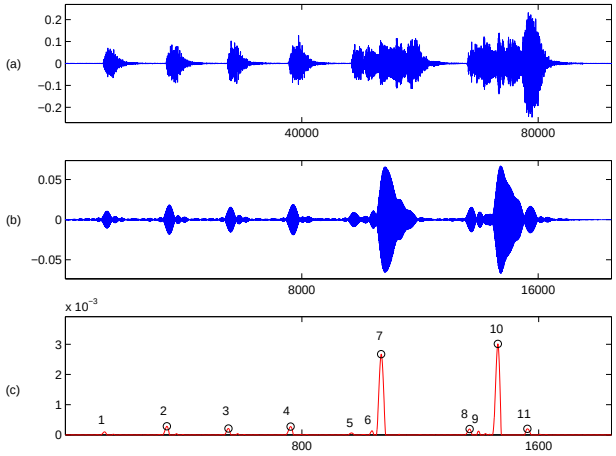

Figure 3. (a) Audio signal of a performance of the score shown in Figure 2. (b) Filtererd audio signal w.r.t. to the pitch $p = 72$. (c) Onset signal OS_{72} with detected peaks indicated by circles.

Altogether, we obtain for each note p a rectified difference signal, also denoted as *onset signal* and written as OS_p, which expresses the local energy increase in the subband corresponding to pitch p. The time resolution depends on the sampling rate. In our case, each sample of the onset curve corresponds to $50/22050 = 2.3$ ms, $10/4410 = 2.3$ ms, and $10/882 = 11.3$ ms, respectively. As an illustration, Figure 3 shows in (a) the waveform of some audio signal representing a performance of the score depicted in Figure 2. The filtered audio signal with respect to the pitch $p = 72$ (C5, 523 Hz) is shown in (b) and the corresponding onset curve OS_{72} in (c).

3.3. Peak Picking

The local maxima or *peaks* of the onset signal OS_p indicate the positions of locally maximal energy increases in the respective band. Such peaks are good candidates for onsets of piano notes of pitches p, $p-12$, or $p-19$. (Recall that besides the fundamental pitch we consider two harmonics.) In theory, this sounds easy. In practice and for complex piano pieces, however, one has to cope with "bad" peaks not generated by onsets: Resonance and beat effects (caused by the interaction of strings) often lead to additional peaks in the onset signals. Furthermore, a strongly played piano note may generate peaks in subbands that do not correspond to harmonics (e. g., caused by mechanical noise). The distinction of such "bad" peaks and peaks coming from onsets is frequently impossible and the peak picking strategy becomes a delicate problem. Since in general the "bad" peaks are less significant then the "good" ones, we use local thresholds (local averages) to discard the peaks below these thresholds.

In (c) of Figure 3 the peaks of the onset signal OS_{72} are indicated by a number, circles indicating which peaks were chosen by the peak picking strategy. Peak 7 and 10 correspond to the note C5 ($p = 72$) played in the right hand. Peaks 2, 3, 4, 8, and 11 correspond to the first harmonics of the note C4 ($p = 60$) played in the left hand. It can be seen that the first harmonics of the first and fifth C4 in the left hand caused the two peaks 1 and 5, which were rejected by our local threshold constraints. This also holds for the "bad" peaks 6 and 9.

After a suitable conversion, we obtain a list of peaks for each piano note p. Each peak is specified by a triple (p, t, s) where p denotes the pitch corresponding to the subband, t the time position in the audio file, and s the size expressing the significance of the peak (or velocity of the note). For computing the score-to-audio alignment, only these peak sequences are required — the audio file as well as the subbands are no longer needed. This considerably reduces the amount of data (e. g., a mono audio signal of sampling rate 22050 Hz requires 30–50 times more memory than the corresponding peaks).

4. SYNCHRONIZATION ALGORITHM

As a preparation for the actual synchronization step, we divide the notes of the score into *score bins*, where each score bin consists of a set of notes with the same musical onset time. For example, for the score in Figure 2 the first score bin is $S_1 := \{48, 52, 55\}$ containing the first three notes, and so on. Similarly, we divide up the peak lists into *peak bins*. To this end, we evenly split up the time axis into segments of length 50 ms. Then we define peak bins by assigning each peak to the segment corresponding to its time position. Finally, we discard all empty peak bins. Altogether we obtain a list $S = (S_1, S_2, \ldots, S_n)$ of score bins and a list $P = (P_1, P_2, \ldots, P_m)$ of peak bins where n and m denote the respective number of bins. The division into peak bins seems to introduce a time resolution of 50 ms. As we will see in Section 4.3, this is not the case since we further process the individual notes after the bin matching procedure.

4.1. Matching Model

The next step of the synchronization algorithm is to match the sequences S of score bins and the sequence P of peak bins. Before doing so, we have to specify a suitable matching model.

Due to note ambiguities in the score such as trills or arpeggios as well as due to missing and wrong notes in the performance, not every note object of the score needs to have a realization in the audio recording. There also may be "bad" peaks extracted from the audio file. Therefore, as opposed to classical DTW, we do not want to force every note bin to be matched with a peak bin and vice versa.

As in our alignment we only consider note onsets, where a note given by the score is associated with the onset time of the corresponding physical realization, each note of the score should be aligned with at most one time position in the audio data stream. Furthermore, notes with the different musical onset times should be assigned to different physical onset times. These requirements lead to the following formal notion of a match:

Definition: A *match* between S and P as defined above is a partial map $\mu : [1 : n] \to [1 : m]$ that is strictly monotonously increasing.

The fact that objects in S or P may not have a counterpart in the other data stream is modeled by defining μ as a partial function and not as a total one. The monotony of μ reflects the requirement of faithful timing: if a note bin in S precedes a second one this should also hold for the μ-images of these bins. μ being a function and strictness of μ ensures that each note bin is assigned to at most one peak bin and vice versa.

4.2. Score Measures

In general there are many possible matches between S and P. To compute the "best" match we need some measure to assign a quality to a match. Similar to DTW, we introduce a local score d measuring the similarity between a note bin S_i and a peak bin P_j. There are many possibilities for adequate score functions depending upon which aspects of the match are to be emphasized. Recall that the note bin S_i is the set of notes of the same musical onset time, where each note is given by its pitch p. The peak bin P_j consists of peaks corresponding to its time segment. Each peak is specified by a triple (q, t, s), where q denotes the pitch of the corresponding subband, t the time position, and s the size of the peak. Then we define the local score $d(i, j) := d(S_i, P_j)$ by

$$d(i,j) := \sum_{p \in S_i} \sum_{(q,t,s) \in P_j} (\delta_{p,q} + \delta_{p+12,q} + \delta_{p+19,q}) \cdot s,$$

where $\delta_{a,b}$ equals one if $a = b$ and zero if $a \neq b$ for any two integers a and b. Note that the sum $\delta_{p,q} + \delta_{p+12,q} + \delta_{p+19,q}$ is either one or zero. It is one if and only if the peak (q, t, s) appears in a subband pertaining to the fundamental pitch or either one of the harmonics of the note p. In this case, the peak (q, t, s) contributes to the score $d(i, j)$ according to its size s. In other words, the local score $d(i, j)$ is high if there are many significant peaks in P_j pertaining to notes of S_i. Note that the peaks not corresponding to score notes are left unconsidered by $d(i, j)$, i. e., the score data indicates which kind of information to look for in the audio signal. This principle makes the score function robust against additional or erroneous notes in the performance as well as "bad" peaks. Since the note and peak bins typically contain only very few (around 1 to 10) elements, $d(i, j)$ can be computed efficiently.

Finally, we want to indicate how to modify the definition of d to obtain other local score functions. In an obvious way, one can account for a different number of harmonics. Moreover, one can introduce note-dependent weights to favor certain harmonics over others. For example, the fundamental pitch dominates the piano sound spectrum over most of its range, except for the lower two octaves where most of the energy is in the first or even second harmonics. This suggests to favor the fundamental pitch for the upper notes and the first or second harmonics for the lower ones. Omitting the factor s in the above definition of $d(i, j)$ leads to a local score function which, intuitively spoken, is invariant under dynamics, i. e., strongly played notes and softly played notes are treated equally.

4.3. Dynamic Programming and Synchronization

Based on the local score function d, the global score of a match μ between S and P is given by the sum

$$\sum_{(i,j):j=\mu(i)} d(i, j).$$

To compute the score-maximizing match between S and P, we use dynamic programming (DP). To this end, we recursively define the global score matrix $D = (D_{i,j})$ by

$$D_{i,j} := \max\{D_{i,j-1}, D_{i-1,j}, D_{i-1,j-1} + d(i, j)\}$$

and $D_{0,0} := D_{i,0} := D_{0,j} := 0$ for $1 \leq i \leq n$ and $1 \leq j \leq m$. Then the score-maximizing match can be constructed from D by the following procedure:

$i := n$, $j := m$, μ `defined on` $\emptyset$
`while` $(i > 0)$ `and` $(j > 0)$ `do`
`if` $D(i, j) = D(i, j - 1)$ `then` $j := j - 1$
  `else if` $D(i, j) = D(i - 1, j)$ `then` $i := i - 1$
  `else` $\mu(i) := j$, $i := i - 1$, $j := j - 1$
`return` μ

Note that this procedure indeed defines a match μ in the sense of our matching model defined in Section 4.1. After matching the note bins with the peak bins, we individually align the notes of S_i to time positions in the audio file, improving the time resolution of 50 ms imposed by the peak bins: For a note $p \in S_i$, consider the subset of all peaks $(q, t, s) \in P_{\mu(i)}$ with $q = p$, $q = p + 12$, or $q = p+19$. If this subset is empty, the note p is left unmatched. Otherwise, assign the note $p \in S_i$ to the time position t belonging to the peak (q, t, s) of maximal size s within this subset. The final assignment of the individual notes constitutes the synchronization result.

As an example, Figure 4 illustrates the synchronization result of the score data shown in Figure 2 and the audio data shown in part (a) of Figure 3. Observe that notes with the same musical onset time may be aligned to distinct physical onset times. (This takes into account that a pianist may play some notes of a chord a little earlier in order to accentuate these notes.) Finally, we want to point out that the assigned time positions generally tend to be slightly delayed. The reason is that it takes some time to build up a sound after a keystroke and that we actually measure the maximal increase of energy. In general, this delay is larger for lower pitches than for higher pitches.

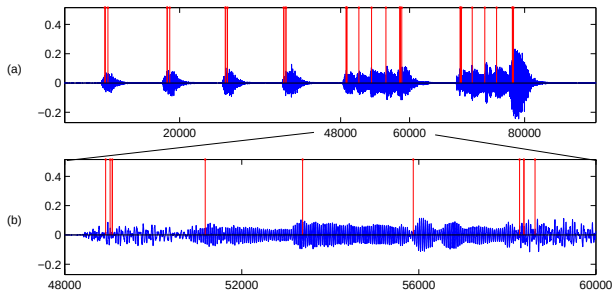

Figure 4. (a) Synchronization result for the audio file of Figure 3. The matched notes are indicated by the vertical lines. (b) Enlargement of a segment of (a).

5. EFFICIENCY AND ANCHOR MATCHES

Recall that running time as well as memory requirements of DP are proportional to the *product* of the number of score and peak bins to be aligned. This makes DP inefficient for long pieces (cf. Section 6). Classical techniques for speeding-up DP computations are based on introducing global constraints which, however, does not improve the complexity substantially.

The best possible complexity for a synchronization algorithm is proportional to the *sum* of the number of score and peak bins. Such a result may be achieved by using techniques employed in areas such as *score following* (see, e. g., [4]) which may be regarded as a kind of online synchronization. Such algorithms, however, are extremely sensible towards wrong or missing notes, local time deviations, or erroneously extracted features, which can result in very poor synchronization results for complex, polyphonic music.

The quality of the computed alignment and the robustness of the synchronization algorithm are of foremost importance. Consequently, increasing efficiency of the algorithm should not degrade the synchronization result. To substantially increase the efficiency, we suggest the following simple but powerful procedure: First, identify in the score certain configurations of notes, also referred to as *anchor configurations*, which possess salient dynamic and/or spectral properties. Such a configuration may be some isolated fortissimo chord, a note or chord played after or before some long pause, or a note with a salient fundamental pitch. Due to their special characteristics, anchor configurations can be efficiently detected in the corresponding audio file using a linear-time/linear-space algorithm. From this, compute score-to-audio matches, referred to as *anchor matches*, for the notes contained in an anchor configuration. Finally, align the remaining notes. This can be done *locally* by applying our DP-based synchronization algorithm on the segments defined by two adjacent anchor matches. The acceleration of the overall procedure will depend on the distribution of the anchor matches. The best overall improvements are obtained with evenly distributed anchor matches. For example, $n-1$ anchor matches dividing the piece into equally long segments speeds up the accumulated running time for all local DP computations by a factor n. The memory requirements are even cut down by a factor of n^2 since only the score matrix of the active local DP computation has to be stored.

Of course, finding suitable anchor configurations is a difficult research problem by itself (cf. Section 7). For the moment, we use a semiautomatic ad-hoc approach in which the user has to specify a small number of suitable anchor configurations for a given piece of music. We have implemented several independent detection algorithms for different types of anchor configurations which are applied concurrently in order to decrease the detection error rate. Pauses in the audio data as well as isolated fortissimo chords are detected by suitably thresholding the ratio between short-time and long-time signal energy computed with a sliding window. Additionally, since pauses as well as long isolated chords correspond to segments with a small number of note onsets, such events can be detected in our peak lists from the extraction step (see Section 3.3) by means of a suitable sparseness criterion. Notes of salient fundamental pitch, i.e., notes whose fundamental pitch does not clash with harmonics of other notes within a large time interval, may be detected by scanning through the corresponding subband using an energy-based measure. To further enhance detection reliability, we also investigate the neighborhoods of the detected candidate anchor matches comparing notes before and after the anchor configuration to the corresponding subband peak information. Then we discard candidate anchor matches exhibiting a certain likelihood of confusion with the surrounding note objects or peak events. The resulting anchor matches may be presented to the user for manual verification prior to the local DP matching stage.

6. EXPERIMENTS AND RESULTS

A prototype of our synchronization algorithm has been implemented in MATLAB. For the evaluation we used MIDI files representing the score data and corresponding CD recordings by various interprets representing the audio data. Our test material consists mainly of classical polyphonic piano pieces of various lengths ranging from several seconds up to 10 minutes. In particular, it contains complex pieces such as Chopin's Etudes Op. 10 and Beethoven's piano sonatas.

It has already been observed in previous work that the evaluation of synchronization results is not straightforward and requires special care. First, one has to specify the granularity of the alignment, which very much depends on the particular application. For example, if one is interested in a system that simultaneously highlights the current measure of the score while playing a corresponding interpretation (as a reading aid for the listener), an alignment deviation of a note or even several notes might be tolerable. However, for musical studies or when used as training data for statistical methods a synchronization at note level or even onset level might be required.

Intuitive objective measures of synchronization qual-

ity are the percentage of note events correctly matched, the percentage of mismatched notes, or the deviation between the computed and optimal tempo curve. (The output of a synchronization algorithm may be regarded as a tempo deviation or *tempo curve* between the two input data streams.) However, such a measure will fail if the note events to be aligned do not exactly correspond (such as for trills, arpeggios, or wrong notes). In this case, the measure might give a low grade (bad score), which is not due to the quality of the algorithm but due to the nature of the input streams. One would then rate a synchronization as "good" if the musically most important note events are aligned correctly. Unfortunately, such an evaluation requires manual interaction, making the procedure unfeasible for large-scale examinations. Similarly, the measurement of tempo curves requires some ground truth about the desired outcome of the synchronization procedure. The design of suitable objective measures, which allow a systematic and automatic assessment of the synchronization results, is still an open research problem and out of our scope.

In this paper, we evaluate our synchronization results mainly via *sonification* as follows: Recall that the input of our synchronization algorithm is a MIDI file representing the score and a WAV file representing the audio data. The algorithm aligns the musical onset times given by the score (MIDI file) with the corresponding physical onset times extracted from the audio file. According to this alignment, we now modify the MIDI file such that the musical onset times correspond to the physical onset times. In doing so we only consider those notes of the score that are actually matched and disregard the unmatched notes. Then we convert the modified MIDI file into an audio file by means of a synthesizer. If the synchronization result is accurate, the thus synthesized audio data stream runs synchronously with the original performance. To make this result comprehensible (audible) we produce a stereo audio file containing in one channel the mono version of the original performance and in the other channel a mono version of the synthesized audio file. Listening to this stereo audio file will exhibit, due to the sensibility of the human auditory system, even smallest temporal deviations of less than 50 ms between note onsets in the two version. To demonstrate our synchronization results we made some of the material available at

`www-mmdb.iai.uni-bonn.de/download/sync/,`

where we provide the score data (as a MIDI file), the audio data as well as the sonification of the synchronization result of several classical piano pieces including the 25 Etudes Op. 100 by Burgmüller, the 12 Etudes Op. 10 by Chopin, and several sonatas by Beethoven. Even for these complex pieces, our synchronization algorithm computes accurate global alignments, which are more than sufficient for applications such as the retrieval scenario, the reading aid scenario or for musical studies. Moreover, most onsets of individual notes are matched with high accuracy — even for passages with short notes in fast succession being blurred due to extensive usage of the sustain pedal. (Listen, e. g., to the synchronization result of the Revolution Etude Op. 10, No. 12, by Chopin). Furthermore, aligning sudden tempo changes such as ritardandi, accelerandi, or pauses generally poses no problem to our algorithm.

Our current algorithm is sensitive towards some specific situations, where it may produce some local mismatches or may not be able to find any suitable match. Firstly, pianissimo passages are problematic since softly played notes do not generate significant energy increases in the respective subbands. Therefore, such onsets may be missed by our extraction algorithm. Secondly, a repetition of the same chord in piano and forte is problematic. Here, the forte chord may cause "bad" peaks (see Section 3.3), which can be mixed up with the peaks corresponding to the softly played piano chord. Such problematic situations may be handled by means of a subsequent algorithm which is based on spectral features rather than based on onset features. A direct comparison to the approach in [1] showed that our algorithm is not only more robust and more efficient — concerning the extraction step as well as the synchronization step — but also results in a more accurate alignment.

We now give some examples to illustrate the running time behavior and the memory requirements of our MATLAB implementation. Tests were run on an Intel Pentium IV, 3 GHz with 1 GByte RAM under Windows 2000. Table 1 shows the running times for several pieces where the pieces are specified by the first column. Here, "Scale" consists of a C-major scale played four times in a row in different tempi, "Bu02" is the Etude No. 2, Op. 100, by F. Burgmüller (see also Figure 2). "Ch03" and "Ch12" are Etude No. 3, Op. 10 ("Tristesse") and Etude No. 12, Op. 10 ("Revolution") by F. Chopin. Finally, "Be01" and "Be04" are the first and fourth movement of Beethoven's sonata Op. 2, No. 1. The second column shows the number of notes in the score of the respective piece and the third column the length in seconds of some performance of that piece. In the fourth and fifth columns one finds the number of note bins and peak bins (see Section 4). The next column shows that the running time for the peak extraction, denoted by $t(\text{peak})$, is about linear in the length of the performance. Finally, the last column illustrates that the actual running time $t(\text{DP})$ of the DP algorithm is, as expected, roughly proportional to the product of the number of note bins and peak bins. The running time of the overall synchronization algorithm is essentially the sum of $t(\text{peak})$ and $t(\text{DP})$. The sonifications of the corresponding synchronization results can be found on our web page mentioned above.

Table 2 shows how running time and memory requirements of the DP computations decrease significantly when using suitable anchor configurations (see Section 5). The third column of Table 2 shows the respective list of anchor matches which were computed prior to DP computation. Here an anchor match is indicated by its assigned time position within the audio data stream. The computation time of these anchor matches is negligible relative

Piece	#notes	len. (sec)	#bins (notes)	#bins (peaks)	t(peak) (sec)	t(DP) (sec)
Scale	32	20	32	65	3	0.4
Bu02	480	45	244	615	22	37
Ch03	1877	173	618	1800	114	423
Ch12	2082	172	1318	2664	116	714
Be01	2173	226	1322	2722	149	716
Be04	4200	302	2472	3877	201	2087

Table 1. Running time of our synchronization algorithm for various piano pieces.

Piece	len. (sec)	list of anchor matches (positions given in sec)	t(DP) (sec)	MR (MB)
Ch03	173	-	423	8.90
		98.5	222	3.20
		42.5, 98.5, 146.2	142	1.45
		42.5/74.7/98.5/125.3/146.2	87	0.44
Be01	226	-	716	28.79
		106.5	363	8.24
		53.1/106.5/146.2/168.8/198.5	129	1.54
Be04	302	-	2087	76.67
		125.8	1042	20.4
		55.9/118.8/196.3/249.5	433	5.09

Table 2. Accumulated running time and memory requirements of the local DP computations using anchor matches.

to the overall running time. The fourth column shows the accumulated running time for all local DP computations. As can be seen, this running time depends heavily on the distribution of the anchor matches. For example, in the "Ch03" piece, one anchor match located in the middle of the pieces roughly accelerates the DP computation by a factor of two. Also the memory requirements (MR), which are dominated by the "largest" local DP computation, decrease drastically (see the last column of Table 2).

7. CONCLUSIONS

In this paper we have presented an algorithm for automatic score-to-audio synchronization for polyphonic piano music. In view of efficiency and accuracy, we extracted from the audio files a sparse but expressive set of features encoding candidates for note onsets separately for all pitches. Using note and peak bins, we further reduced the number of objects to be matched. The actual alignment was computed by dynamic programming based on a suitable matching model, an efficiently computable local score measure, and subsequent individual note treatment. The synchronization results, evaluated via sonification, are accurate even for complex piano music. To increase the efficiency of the synchronization algorithm without degrading the alignment quality, we introduced the concept of anchor matches which can be efficiently computed by a semi-automatic approach.

Despite considerable advances, there are still many open research problems in automatic score-to-audio alignment. One of our goals is to design robust linear-time/linear-space synchronization algorithms producing high-quality alignments. To this end, one could try to automatically extract anchor configurations by means of, e. g., statistical methods and by using additional dynamics parameters. For relatively short segments one could then try to use linear-time score-following techniques instead of DP.

Our sonification only gives an overall feeling of synchronization quality. For the future, it would be important to design objective quality measures and to build up a manually annotated evaluation database, allowing the measurement of technology progress and overall performance.

Automatic music processing is extremely difficult due to the complexity and diversity of music data. One generally has to account for various aspects such as the data format (e. g., score, MIDI, PCM), the genre (e. g., pop music, classical music, jazz), the instrumentation (e. g., orchestra, piano, drums, voice), and many other parameters (e. g., dynamics, tempo, or timbre). Therefore, a universal algorithm yielding optimal solutions for all kinds of music is unrealistic. For the future it seems to be promising to build up a system that incorporates different, competing strategies instead of relying on one single strategy in order to cope with the richness and variety of music.

8. REFERENCES

[1] Arifi, V., Clausen, M., Kurth, F., Müller, M.: "Automatic Synchronization of Musical Data: A Mathematical Approach", In W. Hewlett and E. Selfridge-Fields, editors, *Computing in Musicology*. MIT Press, in press, 2004.

[2] Blackham, E.D.: *Klaviere.* Die Physik der Musikinstrumente, 2. Auflage, Spectrum, Akademischer Verlag, 1998.

[3] Fletcher, N. H., Rossing, T. D.: *The Physics of Musical Instruments.* Springer-Verlag, 1991.

[4] Orio, N., Lemouton, S., Schwarz, D.: "Score Following: State of the Art and New Developments", *Proc. Conf. of New Interfaces for Musical Expression NIME*. Montreal, 36–41, 2003.

[5] Proakis, J.G., Manolakis D.G.: *Digital Signal Processsing*. Prentice Hall, 1996.

[6] Soulez, F., Rodet, X., Schwarz, D: "Improving polyphonic and poly-instrumental music to score alignment", *Proc. ISMIR*. Baltimore, USA, 2003.

[7] Turetsky, R. J., Ellis, D. P., "Force-Aligning MIDI Syntheses for Polyphonic Music Transcription Generation", *Proc. ISMIR*. Baltimore, USA, 2003.

[8] Scheirer, E. D.: "Extracting Expressive Performance Information from Recorded Music", M. S. thesis, MIT Media Laboratory, 1995.