IMPROVED SCORE-PERFORMANCE ALIGNMENT ALGORITHMS ON POLYPHONIC MUSIC

Chun-Ta Chen, Jyh-Shing Roger Jang, and Wenshan Liou

Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan
{chun-ta.chen, jang}@mirlab.org, wsliou@iii.org.tw

ABSTRACT

Automated symbolic music alignment is a challenging task due to the variation of performance by different performers. It becomes more complicated when dealing with polyphonic music because note events could occur at the same time. The goal of this study is to find an efficient algorithm for aligning two polyphonic symbolic representations (MIDI files, for instance) of the same music. To this end, we design two methods for such score-performance alignment that matches the performance with its corresponding score. The first method applies a string matching algorithm based on dynamic programming. The second method is based on the principle of "divide and conquer" that performs efficient alignment recursively. To evaluate the algorithms, we have collected a set of 21 MIDI pairs of classic piano performance with human corrected note-level mapping as ground truth. We have released the dataset as a public resource. Both the proposed algorithms achieved a precision and recall higher than 96% in our experiment, outperforming the most recently proposed method [7] in the literature. Besides, the execution time of proposed methods is much faster the method of [7].

Index Terms— Music alignment, symbolic score following

1. INTRODUCTION

According to music data formats (audio recordings or symbolic representations), there are mainly 3 categories of music alignment processes: audio-to-audio, audio-to-symbolic, symbolic-to-symbolic [1]. The focus of this paper is symbolic-to-symbolic alignment with polyphonic music. In other words, assume there are two symbolic representations of the same underlying music contents; one is the performance music and the other is the corresponding music score. The performance music may contain some performance errors, ornaments, and temporal deviation of notes from the score [2]. The target of music alignment is to indicate how each note in the performance music is matched to the corresponding note in the score according to their correlation of the music structure as temporal evolution.

Algorithms that match a written score to a human performance are essential to reflect the human performance errors. The matching algorithm was pioneered by Dannenberg [3], whose matcher considers only the case of monophonic mu-sic. For polyphonic music, Honing first developed the strict matcher [4], which takes the order of the score in the score as a strict temporal constraint on the performance. The matching strategy is to match the performance and the score note by note. Large [5] proposed another matcher, which divides the score and the performance into clusters (notes played together). This matcher constructs a table, where every cell represents a particular combination of a score cluster and a performance cluster, and tries to find the globally optimal match. However, neither the strict matcher, not the Large matcher can cope with heavily polyphonic performances in a satisfactory way. Therefore, P. Desain [6] proposed the structure matcher, which is based on the idea that temporal structure annotated in the computer score gives a matcher more clues regarding how to interpret the performance. This matcher is able to cope with extreme expressive timing resulting in deviation in the chronological succession of notes. B. Gingras [7] combined the structural and temporal information to generate the more accurate match even for heavily polyphonic performances. B. Gingras also proposes a heuristic for the identification of ornaments and errors that is based on perceptual principles. In this paper, the proposed methods will be compared with the method of B. Gingras [7], which is state-of-the-art in this research.

There are a variety of applications that make use of music symbolic alignment. According to the note alignment result of the score and its performance, we can evaluate the performance music piece, and indicate which notes are redundant (insertions) or missed (deletions) when comparing with the corresponding score. The alignment algorithm can also be applied to real-time applications including page turning during live performances and automated musical accompaniment [3, 8]. Another application is the symbolic music retrieval system [9, 10], which uses a short excerpt of music (query) to search for the similar music in a large music database. These systems usually compute a numeric score on how well a query matches each piece of the database and rank the music pieces according to this score. Furthermore, for the research of audio-to-audio or audio-to-symbolic matching [11], the score-alignment matching algorithm can provide an efficient way to automatically generate the ground truth of audio alignment. With the information of symbolic music alignment of two symbolic music pieces, we can use it as the ground truth of audio alignment by synthesizing the symbolic music into audio format. The advantages of this method are that the timing of note alignments is the same as the original symbolic score format and we can synthesis into various timbres or instruments by using different audio sources.

In this paper, we propose to investigate the note matching algorithm to symbolically-encoded polyphonic music alignment problem. We are interested in an alignment at the note level, which means that the result is the note index. We detail the approaches of note alignment in the following sections. The rest of the paper is organized as follows. Section 2 provides an overview of the proposed approaches. We introduce the general frameworks for polyphonic music alignment by using the string matching algorithm and the dual scan method. The performance analysis of the symbol-ic alignment algorithms are provided in Section 3. Finally, Section 4 concludes this paper.

2. SCORE-PERFORMANCE ALIGNMENT

2.1. Problem Definition

In this section, we describe formally the score-performance alignment problem. The task is to link note event from a score representation to another symbolic representation of a certain performance of a piece of music, while maximizing the number of matched performance notes and identifies all common types of errors. There are a number of peculiar difficulties inherent in music alignment. They are well identified after years of research [2-7]. These difficulties are related to possible sources of mismatch between the human performance and the corresponding score. Musicians can make errors, i.e. playing something differing from the score, because there is always a certain level of unpredictability in the musical live interpretation. Besides, they may add some ornaments or chord variations, which are not specified in the scores, to enrich their performance. The other important issue is temporal deviation, including variations of note onsets, note durations, and tempos. There are slight differences in timing because of free expression in performance. Figure 1 gives an example of temporal deviation, where (a) is the score of the piano music piece; (b) is the piano-roll plot of the music score; and (c) is an example of the performance of the score. Each red horizontal bar in the figure represents a note, with its height denoting the note pitch and its length denoting the note duration. Alphabets are labeled around each note and its corresponding bar. In this example, the note duration and the note onset of the performance deviated considerably from the score. Besides, the durations are 0.8 seconds and 2 seconds in Figure 1 (b) and 1 (c), respectively, indicating the speed of the performance is relatively slower.

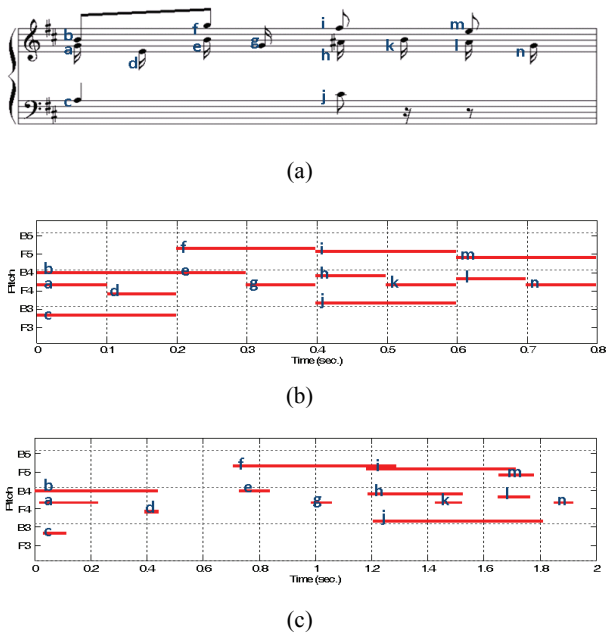

Figure 1. An example of temporal deviation, where (a) is the score of the piano music piece; (b) is the piano-roll plot of the music score; (c) is the piano-roll plot of a person's performance. Each alphabet represents a note.

2.2. Algorithm of Symbolic Music Alignment

In this study, we propose two methods for score-performance alignment and compare their effectiveness and efficiency in the experiments. The first method is based on a string matching algorithm, where the note sequences are converted into strings for alignment. The second method is based on the concept of "divide and conquer", which divides the note sequence data into small segments and aligns these segments recursively. The details of each step are discussed as follows.

2.2.1. Note alignment based on LCS

The first step is to convert the given note sequence into a string. This is achieved by sorting all notes based on their onset time first, and then using the pitch (in terms of semitone) of the notes as the string's elements. This is based on the intuitive concept that the onset of a note is much more important than the offset time. For simplicity, the duration of each note is not considered in the conversion. Our experiments do verify that such conversion is effective and valid for achieving satisfactory performance of the alignment.

For the notes with the same onset, we can simply use the pitch values as the second key for ordering. However, the notes in the score with the same onset may correspond to notes in the performance with different onset. This is likely to happen since a human performer cannot have precisely the same onsets for these same-onset notes in the score. For instance, the first three elements of the string for Figure 1 (a) is cab for Figure 1 (b) and bac for Figure 1 (c). To avoid such difference in onsets, we need to perform onset correction first before converting notes into string. In this study, we adopt a tolerance for onset correction. In other words, any two notes with onset difference less than the tolerance are classified as notes with the same onset. This preprocessing step of onset correction is effective in putting the performance's notes into the right temporal order for alignment.

After the onset correction, each polyphonic music piece is converted into a one-dimensional linear sequence of the corresponding pitch values of notes. Then we can invoke an approximate string matching algorithm to compute the common pattern between two strings of pitch values. We choose longest common subsequence (LCS for short) [12] for our study, which is a well-known string matching method based on dynamic programming.

Let us define the optimum-value function *LCS(a, b)* as the length of the longest common subsequence between string a and string b. Then the recursive formula for LCS can be defined as follows:

$$LCS(ax, by) = \begin{cases} LCS(a,b) + 1, if\ x = y \\ max(LCS(ax,b), LCS(a,by)), if\ x \neq y \end{cases} \quad (1)$$

The boundary conditions are *LCS(a, []) = 0* and *LCS([], b) = 0*. The alignment path (i.e., note-to-note mapping) can be extracted from by back-tracking the alignment table.

The LCS method is effective but not efficient. It is based on dynamic programming with a time and space complexity of *O(|a||b|)*. That is, the computation requires much more memory and CPU time as the note number increases. To avoid these pitfalls, we shall investigate the use a different strategy by matching two note sequences directly.

2.2.2. Dual scan algorithm

The music performance and its corresponding score are in general very similar, except for certain insertions and deletions in the music performance. Based on this observation, we shall propose an efficient method based on the principle of "divide and conquer".

Suppose that we have already found some correct matched pairs (called pivots) with high confidence. Then both note sequences of score and performance can be divided by these pivots into subsequences on which the next-level alignment can be conducted. The success of this approach hinges on how to find the pivots. Here we propose a strategy called dual scan, which includes forward and backward scans. During each scan, the notes with the same pitch and similar onsets will be matched using the matching rules derived from the strict matcher in [4], with forward scan from the beginning and backward scan from the ending. Because of the disturbance from insertion and deletions, we cannot guarantee the correctness of the matched pairs in either of the two scan processes. Thus we choose their intersection as the pivots. Based on the pivots, we then divide the note sequences of the score and the performance into subsequences for the next level of such scan procedure.

1. All notes in both the score and the performance are labeled as unset.
2. Start the first unset node (denoted as note *i*) in the score and find the first same-pitch unset note (denoted as note *j*) in the performance. The unset note is labeled as insertion/deletion if one of the following conditions holds:
   A. There is no same-pitch node in the performance
   B. The first unset node before note *j* has a time difference more than a timing window (ex., 0.2 seconds) from note *j*.
3. Otherwise label both note *i* and *j* as matched.
4. Reverse the role of the score and the performance, and go back to step 2. The procedure continues until we reach the end of both note sequences.

To explain our algorithm clearly, we have introduced 4 global variables in the pseudo code, including *Notes1*, *Notes2*, *match1*, *match2*. *Notes1* and *Notes2* are the note sequences of the score and the performance, respectively. The matched note indices of forward scan and backward scan are stored in arrays *match1* and *match2*, respectively. Each element in these arrays represents a match. More specifically, *match1[i]=j* indicates that note *i* in *Notes1* is matched with node *j* in *Notes2*. The notation *Notes1[a:b]* represents a note subsequence whose indices are from *a* to *b*. The function *Num()* returns the number of elements in an array. The following is the pseudo code for the dual scan method:

```
dual_scan(Pivots)
    for i=1 to Num(Pivots)-1
        /* The subsequence starts from pivot1s to pivot1e. */
        pivot1s ← Pivots[i]
        pivot1e ← Pivots[i+1]

        /* Termination condition:
           When the length of the subsequence is less than or
           equal   to 1, jump to the next iteration. */
        if pivot1e<=pivot1s+1
            continue
        endif

        /* The alignment results of forward scan and backward
           scan will be stored in the index range (pivot1s, pivot1e)
           of match1 and match2, respectively. Their intersection
           will be used as pivots in the next recursive call. */
        forward_scan(pivot1s, pivot1e)
        backward_scan(pivot1s, pivot1e)
        new_Pivots ← match1[pivot1s:pivot1e]
                     ∩ match2[pivot1s:pivot1e]

        /*When there is no new pivots, we need another way to
           determine the matching of the subsequences. */
        if Num(new_Pivots) <= 2
            do LCS to the subsequence (pivot1s:pivot1e)
            continue
        endif

        /* The current subsequence will be divided by new_Pivot
           by recursively calling to dual_scan function with
           new_Pivots  as its input argument. */
        dual_scan(new_Pivots)
    endfor
end
```

The input argument *Pivots* is an array which contains the intersection of the match sets in *match1* and *match2*, this is, *i* is in *Pivots* if *match1[i] = match2[i]* and *i* is the note index of the score. This program iteratively segments note subsequences by these pivots until the length of each subsequence is equal to 0. Moreover, if there is no pivot at all (when the intersection of the sets from two scan processes is empty), we can use LCS to match the subsequences. Here, we just choose the result with more matches from the two scan processes. The other details are shown in the comments of the pseudo code.

3. EXPERIMENTS

3.1. Dataset

For this study, we have collected a heavily polyphonic dataset consisting of 21 pairs of classic piano MIDI files from a variety of MIDI websites. Each pair represents a repertoire and contains one music performance and its counterpart, the corresponding standard score. The total duration is 135 minutes, with more than 51 thousands notes. Because each pair of MIDIs are played by different persons and everyone has his own interpretation of the music, the note numbers of the performance and the score are different from each other for each repertoire, with many insertions, deletions, and swaps of notes. Besides, each MIDI has onset deviations, duration variations, and tempo variations from its counterpart. To get the ground truth of the note-level mapping, we first ran the LCS string matching algorithm to obtain a preliminary note alignment for each pair of MIDI files, and then manually examined the results and corrected them if necessary. The correction work was preceded carefully and double-checked by different persons. We believe this dataset is reliable in the relative research. The complete dataset including the labeled ground truth is available at http://mirlab.org/dataset/public. The ground truth is labeled as two columns, where the first column is the note index in the score file and the second column is the note index in the performance file. Each row [*i* ,*j*] in the ground truth file represents a note mapping, that is, the *i*-th note of the score file is matched to the *j*-th note of its performance file.

3.2. Experimental Results

We shall compare both the proposed methods with one of the most recently proposed method by B. Gingras [7]. Figure 3 shows a typical mapping curve after score-performance alignment. Each match of two notes is represented by a dot in the plane. The note sequences of the score and the performance correspond to the x-axis and the y-axis, respectively. In this figure, blue dots are the correct matches which are the same as the ground truth. Because

the score and the performance are the same repertoire, the distribution of the blue dots is close to a straight line of 45 degrees. False matches, such as the two circled dots in the enlarged plot, usually deviate from the 45-degree line.

Precision and recall are used to evaluate the effectiveness of the proposed methods:

$$Precision = \frac{tp}{tp+fp} \quad (2)$$

$$Recall = \frac{tp}{tp+fn} \quad (3)$$

Terms in the above equations are explained next.

- *tp* is the number of true positives, which are the intersections of the alignment result and the ground truth.
- *fp* is the number of false positives, which are the note-matching set of the alignment result that are absent in the ground truth. In other words, the false positives are the set that are erroneously judged by the alignment algorithm.
- *fn* is the number of false negatives, which are the note-matching set that are not correctly indicated by the alignment algorithm.

Therefore, precision is the fraction of retrieved note matching set that are correct answers, while recall is the fraction of correct note matching set that are retrieved.

The experimental results are listed in Table 1. There are 2 categories for evaluation, piecewise and overall. Piecewise evaluation is based on the average of individual precision/recall of each repertoire. Overall evaluation sums the total true positives, false positives, and false negatives first, and then computes the overall precision and recall. In each category, we compare 3 methods:

- The method proposed by B. Gingras [7], which is one of the most recently proposed method in the literature
- The proposed LCS with onset correction step (OC-LCS for short)
- The proposed dual scan method

From the table, it is obvious that both the proposed methods can achieve a precision and recall higher than 96%, outperforming B. Gingras' method by a substantial margin. Because B. Gingras' method follows an iterative process to optimize the quality of the match over several cycles, it also takes the most computation time. (Note that the computation time in the table is based on MATLAB, so the absolute values are not essential and we should focus on their ratios instead.) Moreover, the precision and the recall of the dual scan method are only about 1% lower than those of LCS, but the execution time is 85 times faster, demonstrating the superb efficiency of the dual scan method. Such remarkable efficiency makes the dual scan method highly suitable for tasks involving massive computation, such as music retrieval from a huge database.

4. CONCLUSIONS AND FUTURE WORK

In this paper, we have proposed two methods for score-performance alignment based on polyphonic symbolic representations. The first method is based on the identification of the longest common subsequence, which usually provides good accuracy but bad efficiency. To reduce the computational complexity, we proposed a more efficient alternative called the dual scan method, which divides the note sequence based on pivots (the intersection of the matched sets generated by the forward and backward scans) and performs recursive alignment. This method

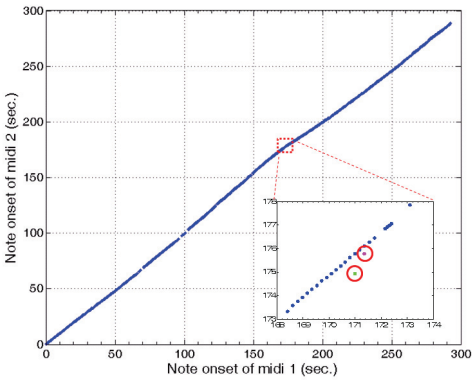

Figure 3. A typical mapping curve of the alignment result of one repertoire in the dataset. The circled dots in the subplot are the false matches.

Table 1. The comparison of B. Gingras' method [7], LCS method with the onset correction (OC-LCS) and the dual scan method. (a) is the piecewise case while (b) is the total case (as explained in the test).

Methods	Precision (%)	Recall (%)	Avg. time (sec)
Gingras [11]	85.43	71.43	168.35
OC-LCS	98.42	97.27	16.32
Dual scan	97.50	96.57	**0.19**

(a) Piecewise evaluation

Methods	Precision (%)	Recall (%)	Total time (sec)
Gingras [11]	91.93	77.49	3535.36
OC-LCS	98.51	97.73	342.70
Dual scan	97.80	96.46	**3.95**

(b) Total evaluation

achieves alignment precision/recall slightly lower than that of LCS, but with a huge saving in computation. Both the proposed methods compare favorably (in terms of efficiency and effectiveness) than one of the recently proposed method in the literature.

Future work of this research will be focused on improving of the alignment algorithms in different aspects. For one thing, the current methods simply regard the ornaments and chord variations as insertions or deletions in the performance, which is kind of over simplistic. We would like to design an efficient algorithm that is able to identify these variations with a better treatment. The final goal is to develop an automatic evaluation system of polyphonic music performances.

6. REFERENCES

[1] Meinard Müller: *Information Retrieval for Music and Motion*, Springer, 2007.

[2] Heijink, H., Desain, P., Honing, H., & Windsor, L.: "Make me a match: An evaluation of different approaches to score-performance matching," *Computer Music Journal*, 24(1), pp. 43–56, 2000.

[3] Dannenberg, R.: "An on-line algorithm for real time accompaniment," *Proceedings of the 1984 International Computer Music Conference*, pp. 193-198, 1984.

[4] Honing, H.: "POCO: An environment for analyzing, modifying, and generating expression in music," *Proceedings of the International Computer Music Conference*, pp. 364–368, 1990.

[5] Large, E.W.: "Dynamic programming for the analysis of serial behaviors," *Behavior Research Methods Instruments & Computers*, 25(2), pp. 238–241, 1993.

[6] Desain, P., Honing, H., and Heijink, H.: "Robust Score-Performance Matching: Taking Advantage of Structural Information," *Proceedings of the 1997 International Computer Music Conference*, pp. 337–340, 1997.

[7] Bruno Gingras and Stephen McAdams: "Improved Score-performance Matching Using Both Structural and Temporal Information from MIDI Recordings," *Journal of New Music Research*, , Volume 40, Issue 1, pp. 43–57, 2011.

[8] Dannenberg, R., & Mukaino, H.: "New techniques for enhanced quality of computer accompaniment," *Proceedings of the 1988 International Computer Music Conference*, pp. 243–249, 1988.

[9] Shyamala Doraisamy and Stefan Ruger: "Robust Polyphonic Music Retrieval with N-grams," *P Journal of Intelligent Information Systems*, Volume 21, Number 1, pp. 53–70, 2003.

[10] Julien Allali, Pascal Ferraro, Pierre Hanna and Matthias Robine: "Polyphonic Alignment Algorithms for Symbolic Music Retrieval," *Lecture Notes in Computer Science*, Volume 5954, pp. 466–482, 2010.

[11] Orio, N., Lemouton, S., Schwarz, D., and Schnell, N.: "Score Following: State of the Art and New Developments," *Proc. of the Conference of New Interfaces for Musical Expression*, pp. 36-41, 2003

[12] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein: *Introduction to Algorithms*, PHI Learning, 3rd ed, 2010.