# HANDLING REPEATS AND JUMPS IN SCORE-PERFORMANCE SYNCHRONIZATION

**Christian Fremerey**
Bonn University
Computer Science
Bonn, Germany
fremerey@iai.uni-bonn.de

**Meinard Müller**
Saarland University and
MPI Informatik
Saarbrücken, Germany
meinard@mpi-inf.mpg.de

**Michael Clausen**
Bonn University
Computer Science
Bonn, Germany
clausen@iai.uni-bonn.de

## ABSTRACT

Given a score representation and a recorded performance of the same piece of music, the task of score-performance synchronization is to temporally align musical sections such as bars specified by the score to temporal sections in the performance. Most of the previous approaches assume that the score and the performance to be synchronized globally agree with regard to the overall musical structure. In practice, however, this assumption is often violated. For example, a performer may deviate from the score by ignoring a repeat or introducing an additional repeat that is not written in the score. In this paper, we introduce a synchronization approach that can cope with such structural differences. As main technical contribution, we describe a novel variant of dynamic time warping (DTW), referred to as *JumpDTW*, which allows for handling jumps and repeats in the alignment. Our approach is evaluated for the practically relevant case of synchronizing score data obtained from scanned sheet music via optical music recognition to corresponding audio recordings. Our experiments based on Beethoven piano sonatas show that JumpDTW can robustly identify and handle most of the occurring jumps and repeats leading to an overall alignment accuracy of over $99\%$ on the bar-level.

## 1. INTRODUCTION

Given a score and a performance of the same piece of music, a common task of music information retrieval consists of synchronizing note events or musical sections given by the score representation with time positions or temporal sections of the performance. A useful example application of such a synchronization is to allow users to navigate in a recorded performance of a piece of music by selecting locations of interest from the visual sheet music representation of the synchronized score and simultaneously playback the performance while highlighting the current playback position in the sheet music [1].

Scores and performances can be given in many different forms and formats. For example, scores can be given as scans of printed sheet music, vector graphics generated by a computer typesetting software, optical music recognition results, symbolic score formats such as MusicXML, Humdrum, or Lilypond, or as MIDI files. Performances are usually given as audio recordings or in form of MIDI files generated by electronic instruments. When aligning score and performance representations, challenging problems arise when the two representations reveal differences in their global overall structures. For example, a performer may ignore a repeat that is written in the score or may introduce an extra repeat that is not written in the score (e.g. an additional verse). Furthermore, a performance may include parts that are not written in score at all (e.g., a cadenza or solo part) or may skip certain parts of an underlying score. Structural differences between scores and performances have been encountered in previous work on on-line score following such as [2–4]. In this scenario, the scores and performances that are synchronized are usually monophonic. The most popular approach for this scenario is to use hidden Markov models (HMM) in combination with a training process to determine model parameter that suit the given type of data. For the case of off-line synchronization of polyphonic scores and performances, dynamic time warping (DTW) in combination with chroma features has become a popular approach [5, 6] because it can deliver similar accuracy than HMMs but without the need for creating and training models. Furthermore, efficient multi-scale implementations can easily be realized for this approach [7]. An overview on on-line and off-line score-performance synchronization approaches is found in [8]. In previous work on off-line score-performance synchronization, a basic assumption usually is that there are no structural differences between the two versions to be aligned. In [6], the authors point out that classical DTW can bypass additional segments such as repeated verses, at least to some extent. Raphael [9] remarks in his work that structural differences such as repeats are a common problem in score-performance synchronization. Content-based comparison of scores and performances also plays an important role in retrieval scenarios [10–12]. As pointed out in [12], retrieval methods may also be used to determine the structural differences between a score and a performance. Further related work has focused on performances only, either in the scenario of general partial music synchronization [13] or structural analysis of performances [14, 15].

In this paper, we describe a novel approach that allows for synchronizing score and performance data in the presence of structural differences. The main motivation for our work originates from a problem of high practical relevance arising in the data acquisition and processing pipeline of a digital music library [1]. Here, the score data is typically obtained by first scanning the given printed sheet music

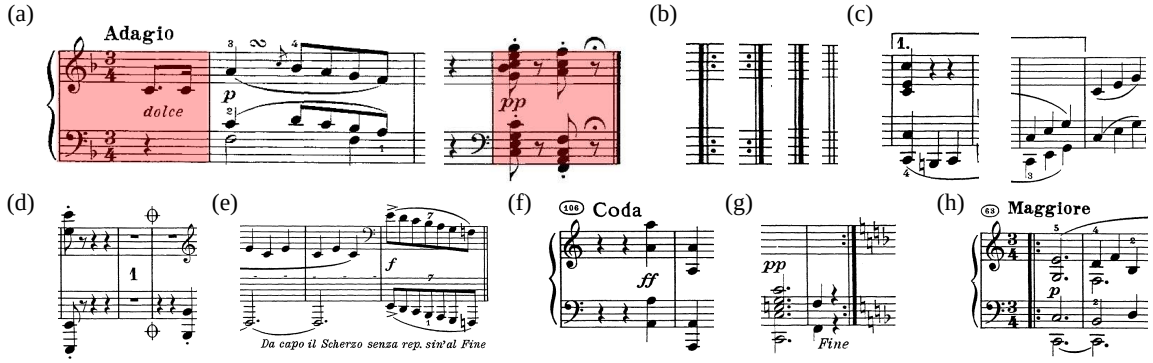


**Figure 1**. Examples for several types of block boundary indicators: (a) beginning and end of movement/song, (b) double bar lines with and without repeat signs, (c) brackets for alternative endings, (d) segno marker, (e) textual jump directive, (f) coda, (g) fine, (h) title heading of new musical section

material and then by converting the digitized images into a symbolic score representation using optical music recognition (OMR). In this process, repeat and jump directives that are written in the printed sheet music (as shown in Fig. 1) are often not recognized reliably by the OMR software. Besides the reasons given above, such missing directives are a major source for structural differences between the resulting score representation and a given audio recording. As the main technical contribution of this paper, we introduce a novel variant of dynamic time warping (DTW), which we refer to as *JumpDTW*. The main idea of our approach is to estimate the repeats and jumps that make the score match the performance and to calculate the actual score-performance alignment within a joint optimization procedure based on a content-based comparison of the score and audio data. The task tackled in this paper is related to the task of computing a possibly large partial alignment of two data streams [13, 16]. However, in contrast to these approaches, our goal is to somehow unfold the score representation to best explain the performance. Furthermore, we assume that the jumps and repeats only occur on musically meaningful positions by exploiting additional structural information given by the score. To this end, the score is searched for structural elements such as double bar lines to divide the score into blocks, see Fig. 1. Then repeats and jumps are allowed only at block boundaries but never inside blocks.

The remainder of this paper is organized as follows. In Sect. 2, we formalize the task of handling repeats and jumps in score-performance synchronization. In Sect. 3, we describe our novel *JumpDTW* algorithm in detail and indicate several extensions. Finally, in Sect. 4, we present experiments performed on a test dataset consisting of piano sonatas by Beethoven and conclude in Sect. 5 with a discussion of future work.

## 2. PROBLEM MODELING

We now assume that we are given one sheet music representation and one performance in form of an audio recording of the same piece of music. After processing the sheet music via OMR, one obtains a symbolic representation referred to as *score* representation. The score is naturally divided into sections that are delimited by either bar lines or the left or right boundary of a grand staff. Even though these sections may differ from the musical bars as they are usually counted in Western sheet music notation, in this paper, we simply refer to each such section as *bar*.

Let $\mathcal{B}$ denote the set of bars appearing in the score and let $K = |\mathcal{B}|$ be the number of bars. Ordering the set of bars by their visual occurrence in the sheet music (canonically ordered by the page number, line number, and left to right within a line), one obtains a sequence $\sigma = (\sigma_1, \ldots, \sigma_K)$, $\sigma_k \in \mathcal{B}$, $k \in [1 : K]$, which we refer to as *score bar sequence*. Note that the score bar sequence does not account for jump and repeat directives, see Fig. 2. Depending on the context, we use the term *bar* to denote either an element of $\mathcal{B}$, the region in the sheet music image that represents the bar, the musical content of the bar, or one of possibly many occurrences of the bar in the performance.

As discussed before, sheet music may contain jump and repeat directives such as repeat signs, alternative endings, dacapos or segnos, see Fig. 1. Because of these directives, the given performance often deviates from the score bar sequence $\sigma$. The musician may even choose to ignore or add some of the displayed repeats or may introduce shortcuts. This leads to a possibly different sequence $\pi = (\pi_1, \ldots, \pi_J)$, $\pi_j \in \mathcal{B}$, $j \in [1 : J]$, which we call *performance bar sequence*, see Fig. 2. Note that in the scenario discussed in this paper, the performance bar sequence $\pi$ is unknown. One application of the approach introduced in the remainder of this paper is to determine this sequence $\pi$.

To relate the score bar sequence $\sigma$ and the performance bar sequence $\pi$, intuitively, the score bar sequence, which represents the source material, has to be suitably *unfolded* to best explain the performance. Here, the *unfolding* typically appears at the jump and repeat directives indicated by the sheet music. Making use of this fact, the problem of unfolding sequences of bars can be reduced to the easier task of unfolding much shorter sequences of so-called *blocks* which are obtained by concatenating suitable subsequences of bars during which no repeats or jumps are expected to occur. To this end, the score is searched for *block boundary indicators* that indicate bars in the score that might serve as source or target for jumps and repeats. Examples of these indicators are depicted in Fig. 1.

Let $k_0 = 0 < k_1 < \ldots < k_{I-1} < k_I = K$ be boundary indices corresponding to the jump and repeat directives. Then, we define the block

$$\beta_i = (\sigma_{k_{i-1}+1}, \ldots, \sigma_{k_i}) \tag{1}$$

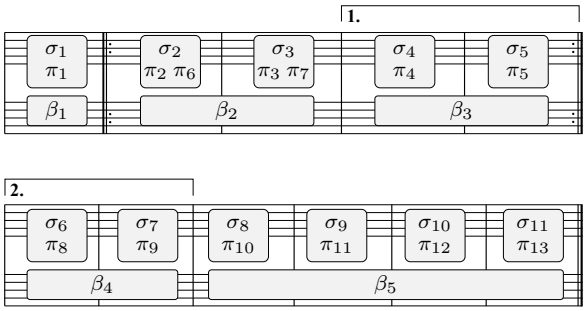


**Figure 2**. Illustration of the score bar sequence $\sigma$, the performance bar sequence $\pi$ and the score block sequence $\beta$.

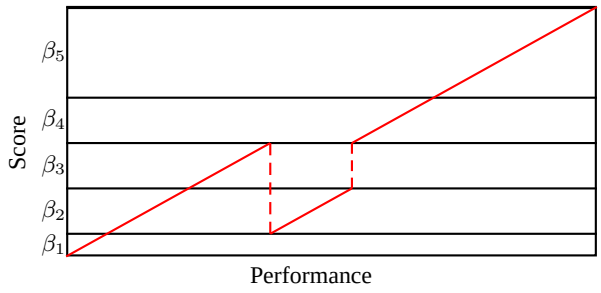


**Figure 3**. Visualization of a score-audio synchronization result with score block sequence $b = (1, 2, 3, 2, 4, 5)$ for the score and performance bar sequences shown in Fig. 2. The red line indicates an alignment path with jumps.

of length $|\beta_i| = k_i - k_{i-1}$ for $i \in [1 : I]$. The resulting *score block sequence* $\beta := (\beta_1, \ldots, \beta_I)$ is a partition of $\sigma$, see Fig. 2. Now, the task of finding the performance bar sequence $\pi$ is reduced to finding a sequence of block indices $b = (b_1, \ldots, b_G)$, $b_g \in [1 : I]$, $g \in [1 : G]$, such that $(\beta_{b_1}, \ldots, \beta_{b_G})$ is as close as possible to the performance bar sequence $\pi$. The task of finding such a sequence $b$ is discussed in the next section. For an example, we refer to Fig. 3. Note that, depending on the context, we will later use the term *block* not only to denote elements of the score block sequence $\beta$, but also to refer to elements of the block index sequence $b$.

## 3. PARTIAL SYNCHRONIZATION WITH JUMPS

Most procedures for score-performance synchronization first convert the two data streams to be aligned into suitable feature representations. Then, based on a local cost measure that allows for comparing features, a global alignment path between the feature sequences is computed using dynamic time warping (DTW). This procedure only works well if the score and the performance are in global correspondence and do not differ in their overall structure.

To account for structural differences as occurring in our scenario, we extend the classical DTW approach to enable jumps in the alignment path. Our idea of allowing jumps is inspired by the way a piece of music is often modeled using a Hidden Markov Model (HMM). Here, the note events of a score are modeled by states which are left-to-right connected to enforce that the music can only move forward but not backward. To account for possible repeats and jumps at certain block boundaries, one then simply adds further connections that connect states representing possible jump sources to states representing possible jump targets. After a short review of classical DTW (Sect. 3.1), we show how the jump directives can be incorporated (Sect. 3.2) and then indicate further DTW variants (Sect. 3.3).

### 3.1 Classical DTW

Introducing some notation, we now summarize the classical DTW approach using a slight reformulation. Let $x = (x_1, \ldots, x_N)$ and $y = (y_1, \ldots, y_M)$ be the feature sequences obtained from the score and performance representation, respectively. Furthermore, let $c$ denote the local cost measure used to compare two features. Then the *local cost matrix* $C$ of dimension $N \times M$ is defined by

$$C(n, m) := c(x_n, y_m) \tag{2}$$

for $(n, m) \in Z$, where $Z := [1 : N] \times [1 : M]$ is referred to as the set of *cells*. A (global) *alignment path* between $x$ and $y$ is a sequence $p = (p_1, \ldots, p_L)$ with $p_\ell = (n_\ell, m_\ell) \in Z$ for $\ell \in [1 : L]$ satisfying the boundary condition $p_1 = (1, 1)$ and $p_L = (N, M)$ and the step condition $p_\ell - p_{\ell-1} \in \Sigma$ for $\ell \in [2 : L]$ Here, $\Sigma := \{(1, 0), (0, 1), (1, 1)\}$ denotes the set of possible steps. The cost of the path $p$ is defined by $\sum_{\ell=1}^{L} C(p_\ell)$. An *optimal alignment path* is defined to be an alignment path having minimal cost over all possible alignment paths.

An optimal alignment path can be computed using dynamic time warping (DTW). First, for a given cell $(n, m) \in Z$, one defines the set $Z_{n,m}$ of possible *predecessors* by

$$Z_{n,m} := \{(n, m) - z \mid z \in \Sigma\} \cap Z. \tag{3}$$

Then, one computes an *accumulated cost matrix* $D$ of dimension $N \times M$. First, one sets $D(1, 1) := C(1, 1)$ and then recursively defines

$$D(n, m) := C(n, m) + \min\big\{D(z) \mid z \in Z_{n,m}\big\} \tag{4}$$

for $(n, m) \in Z \setminus \{(1, 1)\}$. The value $D(N, M)$ represents the cost of an optimal alignment path. Such an optimal path can be constructed based on a simple back tracking algorithm using $D$. For details, we refer to [17].

### 3.2 JumpDTW

To account for structural differences between the score and the performance caused by repeats and jumps, we now extend the concept of an alignment path and the classical DTW approach. Recall that we assume that the jumps occur from ends to beginnings of the blocks $\beta_i$, $i \in [1 : I]$. With regard to the feature representation $x = (x_1, \ldots, x_N)$ of the score, we assume that the beginning of $\beta_i$ corresponds to index $s_i \in [1 : N]$ and the end to index $t_i \in [1 : N]$, where $s_i < t_i$. Furthermore, we assume that the beginning of block $\beta_{i+1}$ immediately follows the end of block $\beta_i$, i.e., $s_{i+1} = t_i + 1$. Let $S := \{s_i \mid i \in [1 : I]\}$ and $T := \{t_i \mid i \in [1 : I]\}$.

Next, an *alignment path with jumps* with respect to the sets $S$ and $T$ is defined to be a sequence $p = (p_1, \ldots, p_L)$ with $p_\ell = (n_\ell, m_\ell) \in Z$ for $\ell \in [1 : L]$ satisfying the boundary condition as before. However, this time we modify the step condition by requiring that either $p_\ell - p_{\ell-1} \in \Sigma$ (as before) or

$$m_{\ell-1} = m_\ell - 1 \ \wedge \ n_{\ell-1} \in T \ \wedge \ n_\ell \in S. \tag{5}$$

In other words, besides the regular steps, we also permit jumps in the first coordinate (corresponding to the score) from the end of any block (given by $T$) to the beginning of any other block (given by $S$), see also Fig. 3.

We now introduce a modified DTW version, referred to as *JumpDTW*, that allows for computing an optimal alignment path with jumps. Recall that, in classical DTW, the set $Z_{n,m}$ of possible predecessor cells encodes all cells from which one can reach the cell $(n, m)$ by applying a single step from $\Sigma$, see (3). The main idea of our modification is to add further predecessor cells that model possible jumps between the block boundaries. To this end, we extend all sets $Z_{n,m}$ for $n \in S$ by setting

$$\tilde{Z}_{n,m} := Z_{n,m} \cup \big(\{(t, m-1) \,|\, t \in T\} \cap Z\big). \quad (6)$$

Furthermore, we set $\tilde{Z}_{n,m} := Z_{n,m}$ for all other $n \in [1 : N] \setminus S$. Intuitively, the additional predecessor cells in $\tilde{Z}_{n,m} \setminus Z_{n,m}$ permit jumps from the end of any block to the beginning of any other block. As in the classical case, one then computes an accumulated cost matrix simply by replacing the sets $Z_{n,m}$ by the sets $\tilde{Z}_{n,m}$ obtaining a matrix $\tilde{D}$. More precisely, we set $\tilde{D}(1,1) = C(1,1)$ and

$$\tilde{D}(n,m) := C(n,m) + \min\big\{\tilde{D}(z) \,|\, z \in \tilde{Z}_{n,m}\big\} \quad (7)$$

for $(n, m) \in Z \setminus \{(1, 1)\}$. Note that for a given $(n, m) \neq (1, 1)$, the set $\tilde{Z}_{n,m}$ only contains cells of the form $(n - 1, m)$ or $(k, m - 1)$ for some $k \in [1 : N]$. In other words, $\tilde{Z}_{n,m}$ only contains cells that lie below or to the left of the current cell $(n, m)$ when the axes are chosen as in Fig. 3. Therefore, $\tilde{D}$ can still be computed recursively in a column-wise fashion. The matrix entry $\tilde{D}(N, M)$ yields the cost of an optimal alignment path with jumps. As for the classical case, such an optimal path can then be constructed based on a simple back tracking algorithm using $\tilde{D}$.

From an optimal warping path with jumps one can derive the underlying sequence of block indices $b = (b_1, \ldots, b_G)$, $b_g \in [1 : I]$, $g \in [1 : G]$, in a canonical way. Starting with the first block, one either enters the subsequent block via a step from $\Sigma$ or enters a different block via a jump. For example, in the case of a jump from $p_{\ell-1} = (t_j, m - 1)$ to $p_\ell = (s_i, m)$ for some $\ell \in [2 : L]$, one obtains $b_{g-1} = j$ and $b_g = i$ for some $g \in [2 : G]$, see also Fig. 3 for an illustration. Having determined the sequence of block indices $b$, one can easily derive the performance sequence $\pi$ by expanding blocks to bars.

### 3.3 Further DTW Variants

Because of the boundary condition, an alignment path starts at $p_1 = (1, 1)$ and ends at $p_L = (N, M)$. Therefore, the score block sequence $b$ is also restricted to start with the first block $b_1 = 1$ and to end with the last block $b_G = K$. In practice, however, a performance may end with a different block. For example, this happens in the presence of a "dacapo", where the piece ends at a block marked with the keyword "fine." To account for this possibility, one can easily modify the JumpDTW algorithm. Instead of looking at the entry $\tilde{D}(N, M)$, one simply has to determine the index

$$n^* := \operatorname{argmin}\big\{\tilde{D}(n, M) \,|\, n \in T\big\}. \quad (8)$$

Then, the alignment path with jumps is computed via backtracking starting with the cell $(n^*, M)$ instead of $(N, M)$. Similarly, one can relax the condition that one has to start with the first block, see [17] for details. Note that further constraints on the jumps can easily be handled by suitably modifying the sets $\tilde{Z}_{n,m}$ of predecessor cells. For example, to restrict the jump possibilities for a given block $\beta_i$, one simply restricts the set $T$ to a suitable subset $T' \subset T$ and then uses $\tilde{Z}_{s_i,m} := Z_{s_i,m} \cup \big(\{(t, m-1) \,|\, t \in T'\} \cap Z\big)$.

## 4. EXPERIMENTS

To evaluate the usefulness of JumpDTW in a practically relevant application, experiments are conducted on the first 15 piano sonatas by Beethoven including a total of 54 individual movements. The score data is obtained from OMR results of a printed sheet music edition, and the performances are given as audio CD recordings. Since the score data does not include any tempo information, a mean tempo is estimated for each movement using the number of bars and the duration of the corresponding performance. For each movement, the score bar sequence $\sigma$ is known and the score block sequence $\beta$ is obtained using block boundary indicators extracted from the score. Note that this may include block boundary indicators where actually no jump or repeat occur in the performance. The performance bar sequence $\pi$ is given as ground truth and is used to derive a ground truth block index sequence $b$ with respect to $\beta$. For our test data set, the total number of score blocks appearing in the sequences $\beta$ of the 54 movements is 242. The total number of score bars is 8832. Note that, because of repeats and jumps, a score block may occur more than once in the performance. Therefore, the total number of blocks appearing in the sequences $b$ is 305 which corresponds to a total of 11836 bars being played in the performance. The total duration of the performance amounts to 312 minutes.

JumpDTW is performed on the data using $\beta$ as described in Section 3.2. From the resulting warping path with jumps, an output block index sequence $b' = (b'_1, \ldots b'_{G'})$ is obtained. In the optimal case, this block index sequence $b'$ would be equal to the ground truth block index sequence $b$. Table 1 shows the results of comparing $b'$ to $b$ using several different evaluation measures. Each row shows the results for different sets of $b'$ obtained using a different JumpDTW variant. Each entry in the table summarizes the results for all 54 movements. The first row, tagged `nojumps`, represents the results when using classical DTW as described in Sect. 3.1, which serves as bottom line in our evaluation. The second row, tagged `s1_plain`, represents the basic JumpDTW algorithm as described in Section 3.2 including the relaxed boundary condition for dacapo/fine cases as described in 3.3.

The numbers plotted in the first six columns are based on a direct comparison of the sequences $b'$ and $b$ and measure how many blocks (abbreviated as `blk`) or performance bars (`bar`) match between the two sequences (`mch`), have been erroneously inserted into $b'$ (`ins`), or have been erroneously omitted in $b'$ (`omt`) with respect to the ground truth $b$. To this end, we calculate an optimum alignment between the two block index sequences using a variant of the edit distance that only allows insertions and deletions (but not replacements). To find an alignment between the two block index sequences that is optimal with respect to the amount of inserted and omit-

| | mch blk % (#) | ins blk % (#) | omt blk % (#) | mch bar % (#) | ins bar % (#) | omt bar % (#) | prf % (#) |
|---|---|---|---|---|---|---|---|
| nojumps | 70.2 (214) | 0.3 (1) | 29.8 (91) | 74.6 (8831) | 0.0 (1) | 25.4 (3005) | 69.8 (8258) |
| s1_plain | 93.4 (285) | 9.2 (28) | 6.6 (20) | 99.2 (11740) | 0.9 (105) | 0.8 (96) | 98.5 (11661) |
| s2_add_special_states | 93.4 (285) | 5.6 (17) | 6.6 (20) | 99.3 (11759) | 0.7 (82) | 0.7 (77) | 98.8 (11692) |
| s3_penalize_0.5_100 | 94.4 (288) | 9.5 (29) | 5.6 (17) | 99.4 (11767) | 0.4 (51) | 0.6 (69) | 99.1 (11725) |

**Table 1**. Evaluation results for classical DTW and different variants of JumpDTW.

ted bars (instead of blocks), each block index entry in the sequences is weighted by the length of the corresponding score block. Each entry in Table 1 is given as a percentage with respect to the total number of blocks/bars in the performance followed by the absolute number in parentheses. For example, the entry $70.2(214)$ in the first row and column means that $214$ blocks of $b'$ have a matching counterpart in $b$, which is $214/305 = 70.2\%$ of the total number of blocks in $b$. Similarly, the entry $74.6(8831)$ for matching bars means that the $214$ matching blocks have a total length of $8831$ bars, which is $8831/11836 = 74.6\%$ of the total length of $b$ in bars.

A further evaluation measure (`prf`), which is plotted in the last column of Table 1, expresses the alignment accuracy on the bar-level. This measure is motivated by the application of visually presenting sheet music that is linked on a bar-wise level to a given recorded audio performance. For this application, we want to measure for how many of the performance bars the alignment computed via JumpDTW is suitably accurate. To this end, the ground truth block index sequence $b$ is used to create a feature sequence $x$ from the score data that matches the repeats and jumps of the performance. Then, this feature sequence is synchronized to a feature sequence $y$ obtained from the performance using classical DTW. From the output warping path, we derive a bar-wise score-performance synchronization that maps each performance bar $\pi_j \in \pi$ to a temporal region with center time $d_j$ in the performance, see Fig. 4. Furthermore, this synchronization delivers a mapping $\phi : [0, D] \to [1 : K]$, with $D$ being the duration of the performance, that for each time positions $d \in [0, D]$ in the performance returns an index $k \in [1 : K]$ indicating that bar $\sigma_k$ is played at time $d$, see also Fig. 4. Since, from manual inspection, the synchronization results obtained when using the ground truth block index sequence $b$ are known to be suitably accurate on a bar-wise level, they are used as a reference for finding deviations in the synchronization results obtained using $b'$. For each performance bar $\pi_j$, we take the bar center time $d_j$ and input it into the mapping $\phi'$ obtained from the synchronization results using $b'$. The performance bar is counted as correctly matched if $\phi'(d_j) = \phi(d_j)$, which means that in the synchronization obtained using $b'$, the time position $d_j$ points to the same bar $\sigma_k$ as in the reference synchronization. Unlike the mere number of matched bars listed in the column `mch bar`, this measure takes into account the extra confusion that is caused in the synchronization by erroneously inserted or omitted bars.

From the results using classical DTW (`nojumps`) one can see that about 70–75% of the blocks and bars of the performance are covered by the plain score bar sequence. The remaining 25–30% are repeats that are omitted in this sequence. The synchronization-based measure indicates a similar result: $69.8\%$ of the center time positions of the bars in the performance were aligned to the correct bar in the score. These results are improved significantly,

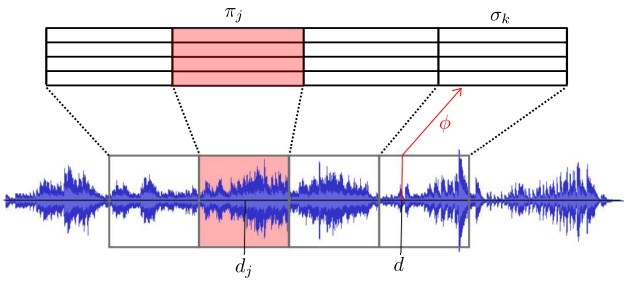


**Figure 4**. Illustration of a bar-wise score-performance synchronization. Each performance bar $\pi_j$ is synchronized to a temporal region of a performance with bar center time $d_j$. Furthermore, a mapping $\phi$ can be derived that for a given time position $d$ in the performance outputs the index $k$ of the corresponding score bar $\sigma_k$.

when using JumpDTW (`s1_plain`). Here, $93.4\%$ of the blocks and $99.2\%$ of the bars are matched correctly. In the synchronization-based measure, $98.5\%$ of the performed bars match the reference synchronization. Even though $28$ blocks have been erroneously inserted and $20$ blocks have been omitted, this amounts to only $105$ inserted bars and $96$ omitted bars, revealing that the mean length of inserted and omitted blocks is only about $4.2$ bars.

Manual inspection of the results for the individual movements reveals that in many cases an extra block is inserted at the beginning or the end of the sequence to cover for silence at the beginning or end of the performance. In one case, this even leads to the last block of the sequence being confused with an incorrect one. To encounter this issue, we extend the JumpDTW algorithm by adding special states to the score representation that model silence at the beginning or end of the performance. The results for this modification are listed in the line labeled `s2_add_special_states` and show slightly improved numbers. An in-depth analysis of the results shows that this modification solved all of the previously mentioned problems caused by initial or trailing silence in the performance. Furthermore, it turned out that $130$ of the $82 + 77 = 159$ inserted and omitted bars occur in just $3$ of the $54$ movements. The performance of the first movement of "Sonata 8, Op. 13, *Pathetique*" contains extreme tempo changes with slow sections of roughly $20$ BPM (beats per minute) alternating with fast sections of about $300$ BPM. This results in a large difference between the estimated mean tempo of the score and the tempo of the slow sections in the performance. The JumpDTW algorithm reacts by erroneously inserting more or less random blocks to cover the unexpectedly slow sections of the performance. A different kind of problem occurs in "Sonata 12, Op. 26, Andante con variazioni". Here, the second block is a variation of the first block that has virtually the same harmonic progression. The JumpDTW erroneously treats this second block in the performance as a repeat of the first block in the score. This behavior is not very surprising considering that

the content-based comparison of score and performance is somewhat noisy and for the chroma-based features used, sections with the same harmonic progression are almost indistinguishable. In "Sonata 13, Op. 27 No. 1, Andante–Allegro" it is again a significant change in the tempo that causes a problem. Here, a repeat of a block (length = $9$ bars) of the faster Allegro section is omitted by JumpDTW, which is provoked by the estimated tempo of the score being significantly slower than the tempo of the corresponding section of the performance. For all of the remaining movements, only blocks of length $2$ or lower are inserted or omitted.

To encounter the problems discussed above, we further extend the JumpDTW approach by introducing a penalty cost for performing jumps in the warping path that is added to the accumulated cost. The cost value is set to $0.5 \cdot \frac{N}{100}$, with $N$ being the length of the score feature sequence. The particular formula is motivated by the idea of choosing a cost value that is close to the cost of matching $1/100$-th of the score to a section of the performance that is not considered similar. Since in our implementation, we use normalized chroma features with a cosine measure for the local cost, a local cost value of $0.5$ is already considered not similar. The results for this modification are listed in the row `s3_penalize_0.5_100`. A closer analysis shows that adding the penalty solves the confusion for the "Andante con Variazioni" and lowers the amount of inserted bars for the slow sections of the "*Pathetique*", which leads to a better overall result. However, the penalty also causes degradation for many of the other movements because short blocks for alternative endings are no longer skipped. Tuning the penalty cost to higher or lower values did not improve the situation. An increased penalty led to an increased amount of erroneously skipped short blocks while a decreased penalty no longer solved the confusion for the two movements discussed above.

## 5. CONCLUSIONS

In this paper, we have formally modeled the task of score-performance synchronization in the presence of structural differences induced by jumps and repeats. To handle such differences, we introduced a novel DTW variant referred to as JumpDTW. The results of the experiments presented in Section 4 show that the JumpDTW approach can successfully align about $99\%$ of the bars played in the performance on the given test dataset with less than $1\%$ of bars being omitted and less than $1\%$ of extra bars being inserted. This positive result suggests that the approach may be useful for the large-scale automatic alignment of OMR data and audio recordings in a digital music library scenario.

Introducing penalty cost for performing jumps did fix some problems occuring on the test dataset but also caused additional errors. Further improvements of our approach are needed in situations where one has large differences (more than a factor of two) in the estimated tempo of the score and the tempo of the actual performance. Also, when using chroma features, blocks that reveal a similar harmonic progression are prone to confusion. Here, combinations with other feature types may help to resolve this problem. Note that, besides the segmentation of the score data into blocks, the JumpDTW approach completely relies on content-based comparison of notes and acoustic data. If further structural information from the score can be incorporated, as for example tempo directives or jumps and repeats as suggested by the notation, many of the remaining issues and inaccuracies might be solved. Besides this, another direction of future work may be to incorporate the case of cadenzas, where the performance contains sections that are not written in the score.

**Acknowledgement.** This work was supported by the German Research Foundation (DFG, CL 64/6-1) and by the Cluster of Excellence on Multimodal Computing and Interaction at Saarland University. We would like to thank the anonymous reviewers for their very helpful comments and suggestions.

## 6. REFERENCES

[1] D. Damm, C. Fremerey, F. Kurth, M. Müller, and M. Clausen. Multimodal presentation and browsing of music. In *Proc. ICMI*, pp. 205–208, Chania, Crete, Greece, 2008.

[2] M.E. Tekin, C. Anagnostopoulou, and Y. Tomita. Towards an intelligent score following system: Handling of mistakes and jumps encountered during piano practicing. In *Computer Music Modeling and Retrieval*, pp. 211–219, 2005.

[3] B. Pardo and W. Birmingham. Modeling form for on-line following of musical performances. In *Proc. National Conference on Artificial Intelligence*, Pittsburgh, Pennsylvania, 2005.

[4] A. Arzt, G. Widmer, and S. Dixon. Automatic page turning for musicians via real-time machine listening. In *Proc. ECAI*, Patras, Greece, 2008.

[5] R. Dannenberg and N. Hu. Polyphonic audio matching for score following and intelligent audio editors. In *Proc. ICMI*, pp. 27–34, San Francisco, USA, 2003.

[6] R. Turetsky and D. Ellis. Ground-truth transcriptions of real music from force-aligned MIDI syntheses. In *Proc. ISMIR*, pages 135–141, Baltimore, Maryland, USA, 2003.

[7] S. Salvador and P. Chan. Toward accurate dynamic time warping in linear time and space. In *Intelligent Data Analysis*, 11(5):561–580, 2007.

[8] R. Dannenberg and C. Raphael. Music score alignment and computer accompaniment. *Comm. ACM, Special Issue: Music Information Retrieval*, 49(8):38–43, 2006.

[9] C. Raphael. A hybrid graphical model for aligning polyphonic audio with musical scores. In *Proc. ISMIR*, Barcelona, Spain, 2004.

[10] J. Pickens, J. P. Bello, G. Monti, T. Crawford, M. Dovey, M. Sandler, and D. Byrd. Polyphonic score retrieval using polyphonic audio. In *Proc. ISMIR*, Paris, France, 2002.

[11] I. Suyoto, A. Uitdenbogerd, and F. Scholer. Searching musical audio using symbolic queries. *IEEE TASLP*, 16(2):372–381, 2008.

[12] C. Fremerey, M. Clausen, M. Müller, and S. Ewert. Sheet music-audio identification. In *Proc. ISMIR*, pp. 645–650, Kobe, Japan, 2009.

[13] M. Müller and D. Appelt. Path-constrained partial music synchronization. In *Proc. ICASSP*, pp. 65–68, Las Vegas, Nevada, USA, 2008.

[14] M. Goto. A chorus section detection method for musical audio signals and its application to a music listening station. *IEEE TASLP*, 14(5):1783–1794, 2006.

[15] M. Levy and M. Sandler. Structural segmentation of musical audio by constrained clustering. *IEEE TASLP*, 16(2):318–326, 2008.

[16] J. Serrà, E. Gómez, P. Herrera, and X. Serra. Chroma binary similarity and local alignment applied to cover song identification. *IEEE TASLP*, 16:1138–1151, 2008.

[17] M. Müller. *Information Retrieval for Music and Motion*. Springer, 2007.