Transactions of the International Society for
Music Information Retrieval

Author1, F., Author2, S., and Author3, T. (2020). Automatic Generation of Piano Score Following Videos, *Transactions of the International Society for Music Information Retrieval*, V(N), pp. xx–xx, DOI: https://doi.org/xx.xxxx/xxxx.xx

RESEARCH ARTICLE

Automatic Generation of Piano Score Following Videos

First Author1*, Second Author2†, and Third Author3*

Abstract

This article studies the problem of generating a piano score following video from an audio recording in a fully automated manner. This problem contains two components: identifying the piece and aligning the audio with raw sheet music images. Unlike previous work, we focus primarily on working with raw, unprocessed sheet music from IMSLP, which may contain filler pages, other unrelated pieces or movements, or repeats and jumps whose locations are unknown a priori. To solve this problem, we combine state-of-the-art methods with a novel alignment algorithm called Hierarchical DTW to handle discontinuities, which are the bottleneck on system performance. Hierarchical DTW handles repeats and jumps by considering all line breaks as possible jump locations, and it applies DTW at both a feature level and a segment level. We evaluate our algorithm with 200 PDFs from IMSLP and real audio recordings from Youtube. Our experiments show that Hierarchical DTW consistently outperforms a previously proposed Jump DTW algorithm in handling various types of discontinuities. We present extensive experimental results and analysis of the proposed algorithm.

Keywords: score following, alignment, sheet image, audio.

1. Introduction

This article investigates the problem of automatically generating a score following video given only an audio recording of a piano performance. In this task, the single input to our system is an audio recording, and the output is a video showing the corresponding line of sheet music at each instant in the audio recording. To accomplish this task, the system must solve two problems: (i) identifying the piece and downloading the corresponding sheet music from IMSLP, and (ii) aligning the raw, unprocessed sheet music images with the audio to generate the score following video. A system like this could be used to add visual information to audio-only Youtube recordings in a way that enhances a viewer's enjoyment.

Audio-sheet image alignment and retrieval have been actively studied in recent years. There are three general approaches to the problem. The first approach is to use optical music recognition (OMR) to convert the sheet music into a MIDI format, compute chroma-like features from the MIDI, and then compare the resulting features with chroma features extracted from audio. This approach has been applied to audio-sheet image alignment (Kurth et al., 2007; Damm et al., 2008; Thomas et al., 2012; Fremerey et al., 2010; Grachten et al., 2013) as well as various audio-sheet image retrieval scenarios (Fremerey et al., 2008, 2009). The second approach is to directly extract mid-level features from both sheet music and audio. This mid-level representation avoids complex dependencies like key signature and accidentals, and instead simply encodes simplified information like the locations of noteheads relative to staff lines (İzmirli and Sharma, 2012; Yang et al., 2019). The third approach is to learn a common feature embedding for snippets of audio and sheet images in an end-to-end manner using deep convolutional neural networks. These features encode the semantic similarity between a chunk of audio and a portion of sheet music. This has been done using convolutional neural networks combined with canonical correlation analysis (Dorfer et al., 2016b, 2018c), pairwise ranking loss (Dorfer et al., 2017, 2018a), or other loss metric. This approach has been applied to several audio-sheet image alignment scenarios (Dorfer et al., 2017, 2016a) as well as audio-sheet retrieval problems (Dorfer et al., 2017, 2018a,c). Recent works (Dorfer et al., 2018b; Henkel et al., 2019) have also explored formulating the score following problem as a reinforcement learning game and shown promising results. See Müller et al. (2019) for a summary of work in this field.

Our approach differs from previous work in four ways. First, our alignment system does not assume that the sheet music is provided beforehand. In all pre-

*University One, Campus Rd. 3, 12345 Mirland
†Research Institute Two, Campus Rd. 4, 12345 Flatland

vious studies of sheet-image synchronization, the sheet music is assumed to be known. Second, we study the audio-sheet retrieval problem at a much larger scale than previous studies. Whereas previous works on audio-sheet retrieval have used databases containing dozens or hundreds of pieces, we study the retrieval task using all piano sheet music scores in the IMSLP database. Third, we work with completely raw, unprocessed PDFs from IMSLP. Many previous works either focus on training and testing with synthetic sheet music or assume that the sheet music is pre-processed in various ways (e.g. removing filler pages or pages containing other pieces or movements). Fourth, our system is explicitly designed to handle repeats and jumps in the sheet music when jump locations are unknown a priori. While there are several existing offline alignment algorithms for handling discontinuities, they are not well-suited to our task for one of three reasons: (a) they assume that repeat symbols have been identified so that potential jump locations are known a priori (Fremerey et al., 2010; Joder et al., 2011), (b) they simply ignore or skip repeated sections, rather than estimating the alignment at all times (Grachten et al., 2013; Müller and Appelt, 2008), or (c) they allow completely unconstrained jumps, such as in practice sessions (Jiang et al., 2019).[1] Hierarchical DTW fills the gap of offline alignment of performances when jump locations are unknown.

This work has three main contributions. First, we propose a novel alignment algorithm called Hierarchical DTW that handles repeats and jumps when repeat locations are unknown a priori. Hierarchical DTW considers every line break as a potential jump location and incorporates musical domain knowledge to only consider reasonable transitions. The algorithm performs a feature-level alignment between each sheet music line and the whole audio recording, and then it performs a second alignment at the segment level. Second, we present extensive empirical evaluation and analysis of Hierarchical DTW. We perform experiments on various types of audio data (synthetic audio and real audio), various types of jumps (regular repeats and D.S. al fine), and different assumptions on jump locations (only at line breaks and uniformly distributed). We show that Hierarchical DTW consistently outperforms a previously proposed Jump DTW algorithm (Fremerey et al., 2010) in all scenarios. Third, we integrate our alignment approach with a large-scale audio-sheet retrieval system to solve the video generation task using only an audio recording as input. By performing the retrieval and alignment tasks together, our system can generate a score following video in a completely automated manner. This system thus has a very clear practical use: it can be used to augment audio-only recordings with visual information.

The rest of the article is organized as follows. Section 2 describes our proposed system and algorithms. Section 3 explains the experimental setup. Section 4 presents the empirical results. Section 5 provides a detailed analysis of system behavior. Section 6 concludes the work.

2. System Description

Figure 1 shows an overview of the system. There are four main components: feature extraction, retrieval, alignment, and video generation. These will be discussed in the next four subsections.

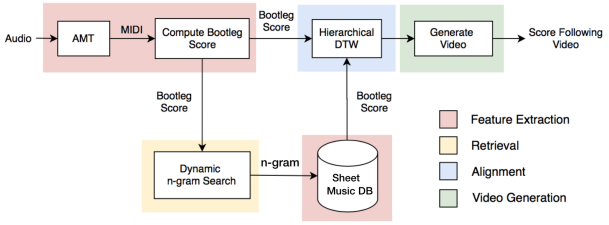

Figure 1: Architecture of the proposed approach. The audio query is converted into a bootleg score and used to find a match in a precomputed database of sheet music bootleg scores (retrieval). The matching sheet music and the audio query are then aligned, and the predicted alignment is used to generate a score following video.

2.1 Feature Extraction

The first step is to convert both audio and sheet music images into a common bootleg score representation. This feature extraction includes the two upper left blocks in Figure 1, as well as the offline construction of the sheet music database. The bootleg score is a recently proposed feature representation for aligning piano sheet music images and MIDI (Yang et al., 2019; Tsai et al., 2020). It is a manually designed, low-dimensional feature representation that encodes the location of noteheads in sheet music relative to the staff lines. The bootleg score itself is a $62 \times N$ binary matrix, where 62 indicates the total number of possible staff line positions in both the left and right hands, and where N indicates the total number of simultaneous note events.

There are slightly different forms of the bootleg score representation for alignment tasks and for retrieval tasks. Figure 2 shows an overview of how to compute a bootleg score from audio (top) and from sheet music (bottom) for an alignment task, as described in the original formulation (Yang et al., 2019). These will be described in the next two paragraphs, followed by a third paragraph describing a variant that is used for the retrieval task.

The audio bootleg score features for audio-sheet alignment are computed in two steps (top of Figure 2). The first step is automatic music transcription. We use the pre-trained Onsets & Frames piano transcription system (Hawthorne et al., 2018) to convert the input audio into MIDI. The second step is to compute

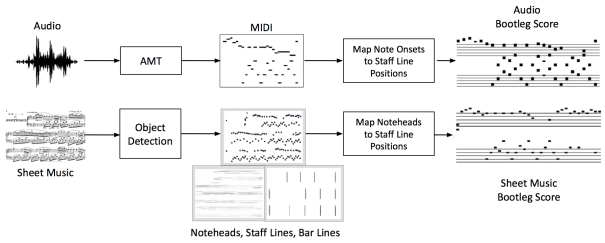

Figure 2: Computing a bootleg score from audio (top) and sheet music (bottom).

a bootleg score from the estimated MIDI. As described in Yang et al. (2019), this is done by first extracting a list of all individual note onsets and grouping them into note events, each containing one or more simultaneous note onsets. This list of note events is then projected into the bootleg feature space using the rules of Western musical notation. When there are ambiguities due to enharmonic representations or left/right hand attribution, multiple noteheads are placed in the bootleg score at all potential locations.

The sheet music bootleg score features for audio-sheet alignment are computed in three steps (bottom of Figure 2). First, we convert the IMSLP sheet music PDF into a sequence of 300 dpi png images. Second, we detect filled noteheads, bar lines, and staff lines in each image. Since these objects are all simple geometrical shapes (filled circles and straight lines), they can be detected efficiently and robustly using simple classical computer vision techniques. Third, we estimate the location of filled noteheads relative to the staff lines and encode the resulting information into a bootleg score. If there are multiple pages in the PDF, we also concatenate the bootleg scores from all pages to construct a single long bootleg score for the entire PDF. The reader is referred to Yang et al. (2019) for more details.

The bootleg score features for audio-sheet *retrieval* are a slight variant of the features described above (Tsai, 2020). There are two differences between the audio and sheet music (alignment) bootleg score representations that prevent their use in a cross-modal hashing framework. The first difference is that black notes on the piano have enharmonic representations that create ambiguity about a notehead's location on the sheet music. For example, MIDI note number 61 can be represented as a C-sharp or a D-flat, which correspond to two different locations in the sheet music. To handle this first difference, we can generate two completely separate versions of the bootleg score: one in which all black notes are interpreted as sharps, and one in which all black notes are interpreted as flats. A search can be performed with both bootleg score versions, and the one with the higher match score is kept. The second difference is that notes in the middle register may appear in the left or right hand staves. To handle this second difference, we place all notes in the middle register in both the right and left hand staves. By modifying the bootleg score features in these two ways, the retrieval bootleg score representation can be used for large-scale retrieval, as recently demonstrated in Tsai (2020).

2.2 Retrieval

The second step is to identify the sheet music in IMSLP that matches the audio query. Our approach combines ideas from three recent works: (1) automatic piano transcription using the Onsets and Frames system (Hawthorne et al., 2018), (2) bootleg score features modified for large-scale MIDI-sheet image retrieval (Tsai, 2020), and (3) dynamic n-gram fingerprinting (Yang and Tsai, 2020), which has recently demonstrated sub-second image-image retrieval of IMSLP piano sheet music. While this part of our system simply combines existing techniques, we believe this is the first empirical study of audio-sheet image retrieval at the scale of IMSLP. We will describe our retrieval system in two parts: database construction and search.

The database construction consists of two steps. First, we compute the retrieval bootleg score for every piano sheet music score in IMSLP. Second, we construct several n-gram reverse index databases. We represent each column in the bootleg score matrix as a 64-bit fingerprint, consider every set of N consecutive fingerprints as an n-gram, and store the location information for each n-gram in a reverse index. We construct N-gram databases for $N = 1, 2, 3, 4$.

The search process consists of three steps. The first step is to compute the retrieval bootleg score for the input audio file as described in section 2.1. The second step is to construct a sequence of dynamic n-grams based on the audio bootleg score. For each column in the audio bootleg score, a single n-gram is constructed with a dynamically selected value of n, which is the smallest value for which the number of n-gram matches in the database is below a threshold. This avoids performing lookups on extremely common n-grams (e.g. a single notehead present) and ensures that every query n-gram is sufficiently discriminative to warrant a table lookup. The third step is to rank the database items using the histogram of offsets method (Wang, 2003). For every IMSLP PDF in the database, we construct a histogram of relative offsets for matching n-grams and then use the maximum bin count in each histogram as a match score. Because there might be several PDFs for a single piece, we compute the piece score as the maximum score among its constituent PDFs. Finally, we sort the pieces by their scores to generate a ranked list. The piece with the highest score is selected, and its (precomputed) sheet music bootleg score is passed along to the alignment subsystem.

2.3 Alignment

The third main step is to align the audio with the selected sheet music. We frame this problem as aligning the audio bootleg score with a sequence of sheet music bootleg score fragments, each representing a single line of music. Because we are working with completely unprocessed data, this alignment must be able to handle filler pages, unknown starting and ending locations, and structural discontinuities. We propose a novel alignment algorithm called Hierarchical DTW that allows for possible jumps at every line break. This algorithm has three main stages, as shown in Figure 3.

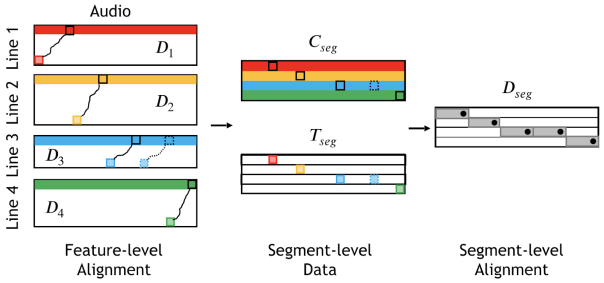

Figure 3: Example of Hierarchical DTW on a given piece. There are four lines in the sheet music. In the audio, the four lines are performed in the following order: Line 1, Line 2, Line 3, Line 3, Line 4. On the left, we use subsequence DTW to perform feature-level alignment. The optimal paths are shown as black curves. Note that for Line 3, two matching paths are shown, where the dotted path is the suboptimal path corresponding to a repeat. In the middle, the two segment-level data matrices are shown. C_{seg} records the last rows of D_1, D_2, D_3 and D_4. T_{seg} records the starting location of every possible subsequence path. On the right, D_{seg} shows the optimal path through C_{seg}, reflecting the predicted order of Line 1, Line 2, Line 3, Line 3, Line 4.

The first stage is feature-level alignment. We align every sheet music bootleg score fragment (corresponding to different lines) against the entire audio bootleg score using subsequence DTW. Subsequence DTW is a variant of DTW that finds the optimal matching path between a short query sequence and any subsequence in a reference sequence (see Müller (2015) for more details). We use a normalized negative inner product as a distance metric and possible transitions $\{(1,1),(1,2),(2,1)\}$ with weights $\{1,1,2\}$. At the end of this stage, we have a cumulative cost matrix D_i and backtrace matrix B_i for every line of sheet music. The left part of Figure 3 shows the cumulative cost matrices D_i for four lines of sheet music.

The second stage is constructing segment-level data matrices C_{seg} and T_{seg}. These two matrices will be used to perform alignment at the segment-level in the third stage. The C_{seg} matrix plays the role of the pairwise cost matrix at the segment level. It is constructed by stacking the last rows of D_i (from stage 1), so that $C_{seg}[i,j]$ indicates the optimal subsequence path score of line i ending at position j in the audio bootleg score. The T_{seg} matrix records the starting locations of all optimal subsequence paths. It is formed by backtracking from every possible end location in every D_i matrix, so that $T_{seg}[i,j]$ indicates the starting location of the optimal subsequence path for line i ending at position j in the audio bootleg score. In Figure 3, five representative elements in T_{seg} are shown, along with their corresponding subsequence paths in stage 1.

The third stage is segment-level alignment. Our goal is to find the optimal path through C_{seg} using the information in T_{seg} to inform allowable transitions. We use dynamic programming to construct a segment-level cumulative cost matrix D_{seg} and corresponding backtrace matrix B_{seg} column by column (from left to right, as shown in Figure 3). The first column of D_{seg} is initialized to all zeros to allow starting from any line of sheet music without penalty. Note that, unlike regular DTW, the set of allowable transitions is unique for every position in D_{seg}. When computing $D_{seg}[i,j]$, there are two possible types of allowable transitions. The first type is skipping, which is a $(0,1)$ step (i.e. directly to the right in Figure 3) without accumulating any score. The second type is matching a line of music. If we let $k \triangleq T_{seg}[i,j]$ denote the starting position of the current matching line, then the transition from the previous line of music must come from $(l,k-1)$, where l is the index of the sheet music line we jump from. Thus, a candidate value for $D_{seg}[i,j]$ is $\tilde{D}_{seg}[i,j] = D_{seg}[l,k-1] + C_{seg}[i,j] * \gamma(l,i) + \alpha(l,i)$, where $\gamma(l,i)$ is a multiplicative weight and $\alpha(l,i)$ is an additive penalty for jumps. To calculate $D_{seg}[i,j]$, we calculate the minimum cost among all possible transitions $\{(i,j-1),(0,k-1),(1,k-1),...,(L-1,k-1)\}$, where L indicates the number of sheet music lines in the PDF. Once D_{seg} has been filled out, we identify the minimum element in the last column and backtrace to obtain the optimal path. Figure 3 indicates the optimal segment-level path as a sequence of black dots in D_{seg}, along with the induced segmentation of audio as gray rectangles.

Hierarchical DTW can be adapted to a range of scenarios by setting $\gamma(l,i)$ and $\alpha(l,i)$ appropriately. For example, setting $\alpha(l,i)$ to infinity whenever $l \neq i+1$ will disallow all jumps in the system. In the experiments reported in this paper, we set these hyperparameters in the following way. We allow the system to move to the next line, stay on the same line (slowing down), or skip a line (speeding up) with $\gamma = 1$, 0.5, and 0.5, respectively. This allows the system to handle time warping at both the segment level (stage 3) as well as the feature level (stage 1). We only allow jumps to lines of music that have been seen before, or immediately after a line that has been seen before (i.e. moving to the next line of music). We can enforce this constraint by

keeping track of the range of lines that have been seen at every position $D_{seg}[i, j]$, and updating this information in the same way that we update the backtracking information. For valid jumps, we impose an additive penalty of $\alpha = -\beta * p_{avg}$, where beta is a hyperparameter and p_{avg} is the average subsequence path score of a single line. We empirically found that the performance of the system is not very sensitive to the value of beta, and we use $\beta = 0.7$ for all experiments in this paper.

2.4 Video Generation

The fourth main step is to generate a score following video based on the estimated alignment. By modifying the sheet music bootleg score feature extraction to keep track of the pixel locations of individual notes and staves, we can map each time instant in the audio recording to a predicted pixel region in the sheet music images. Thus, we can create the score following video by (i) playing the audio in the background, (ii) showing the estimated line of sheet music at each time instant, and (iii) optionally overlaying a cursor on the sheet music line to show the exact predicted position in the score. Figure 4 shows an example frame from a video.

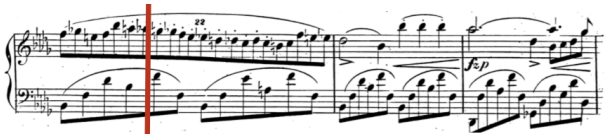

Figure 4: A single frame of the video generated from Chopin Nocturne Op. 9 No. 1. The video shows the estimated line of music and uses a red cursor to indicate the predicted location.

3. Experimental Setup

In this section, we provide an overview of the dataset, benchmarks, and metrics used in the experiments.

3.1 Data

The data we used is derived from the Sheet Midi Retrieval (SMR) dataset (Tsai et al., 2020). The SMR dataset contains scanned sheet music from IMSLP and corresponding MIDI files for 200 solo piano pieces across 25 composers. For the sheet music, there are ground truth annotations of how many measures are on each line and how many lines are on each page. For the MIDI files, there are measure-level timestamps. This dataset was originally collected to study a MIDI-sheet image alignment task.

We derived our data set in the following way. We synthesize the MIDI files to audio using the FluidSynth library. We also found one real audio recording of each of the 200 pieces on Youtube and manually annotated the beginning and ending timestamps for each line of sheet music. For each sheet music PDF in the SMR dataset, we went back to the IMSLP webpage and downloaded the raw PDF without cropping any filler pages or other unrelated pieces or movements. We combine the annotations for sheet music and MIDI to generate timestamps for each line of sheet music. Finally, we convert all sheet music PDFs into 300 dpi PNG files and manually annotate the vertical pixel location of each line of sheet music. Note that this data set does not contain any repeats or jumps, so we refer to it as the 'No Repeat' data.

We use the No Repeat data to generate several benchmarks of interest. To systematically study how well our system performs with various structural discontinuities, we simulated four different types of discontinuities: Repeat x1, Repeat x2, Repeat x3, and D.S. al fine. Figure 5 shows the steps to generate these benchmarks. First, we determine the timestamps in the audio recording corresponding to line breaks in the sheet music. Then, we randomly select a certain number of line breaks based on the type of repeat structure: 2 for Repeat x1, 3 for Repeat x2, 4 for Repeat x3, and 3 for D.S. al fine. Then we cut and concatenate the audio recording based on the selected line break locations, and also modify the ground truth time annotations accordingly. Figure 5 shows an example of this process for each of the four repeat structures. In the analysis section, we also consider a variant of the above schema in which jump locations can occur anywhere within a line, not just at line breaks (bottommost example in Figure 5).

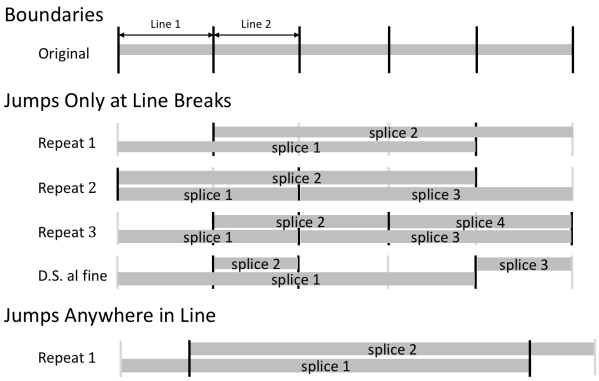

Figure 5: Generating audio with various types of repeats. To generate data with repeats at line breaks, we segment the original audio recording at sheet music line breaks, sample boundary points without replacement, and then splice and concatenate audio segments as shown above. To generate data with repeats that can occur mid-line, we first sample lines and then randomly choose time points in those lines.

3.2 Metrics

In order to maximize intuition and understanding, we evaluate the retrieval task and the alignment task separately.

We evaluate the retrieval task with mean reciprocal rank (MRR). Because we group the PDFs by piece,

there is only one correct match in the database. MRR is the average value of $\frac{1}{R}_i$ across all queries, where R_i indicates the rank of the true match for the i-th query. It ranges between 0 and 1, where 1 corresponds to perfect performance.

We evaluate the alignment task with a simple accuracy metric. Because this system is a user-facing application, we select a metric that correlates directly with the quality of the user's experience in viewing the score following video. Accordingly, we simply calculate the percentage of time that the correct line of music is being shown. We consider a line to be shown correctly if the full vertical extent of the matching line is displayed and no portion of any incorrect line is shown. When calculating the accuracy, we also use a scoring collar to exclude short intervals of time $[t_i - \Delta, t_i + \Delta]$ around each ground truth line transition timestamp t_i. This is a common practice in other segmentation tasks like speech activity detection (NIST, 2016). By evaluating with different values of delta, we can characterize how much errors are clustered near the line transitions.

We also measure the runtime for both tasks. The runtime for the retrieval task only includes the time required to find a match in the database given a query bootleg score. It does not include building the database, performing automatic music transcription, or computing bootleg score features. The runtime for the alignment task only includes the alignment algorithm (Hierarchical DTW or alternative baseline), assuming that the features have already been computed. The runtime for feature extraction is reported separately, since it is shared between the alignment and retrieval tasks.

4. Results

In this section, we present the empirical results on both the retrieval and alignment tasks.

4.1 Retrieval

We only evaluate one system for the retrieval task (described in Section 2.2). While there have been many approaches proposed for the audio-sheet image retrieval task (e.g. (Fremerey et al., 2008, 2009; Dorfer et al., 2017, 2018a)), none have been studied at the scale of IMSLP. We are aware of only one work (Yang and Tsai, 2020) that studies an image-image retrieval task on the full IMSLP piano dataset and provides precomputed features. Because the IMSLP dataset is so large, this previous work required the use of a supercomputing center to pre-compute features. Due to practical computational constraints, we only evaluate our proposed system, which reuses the pre-computed IMSLP features provided in this previous work but applies it to a different cross-modal task.

Table 1 shows the results of the retrieval task for both synthetic and real audio. There are three things to notice about these results. First, the retrieval accuracy is consistent across all benchmarks for both types of audio. With synthetic audio, the MRRs for all repeat types are in the range of 0.75 to 0.78. With real audio, the MRRs for all repeat types are in the range of 0.60 to 0.63. These results indicate that the retrieval system is relatively insensitive to structural discontinuities. Second, the retrieval accuracy with real audio is consistently worse than with synthetic audio by about $0.14 - 0.15$ MRR. This gap reflects the fact that the automatic music transcription system performs better with synthetic audio than real audio. Third, the absolute MRR values are relatively high, indicating strong performance. The IMSLP piano database contains over 29,000 scores, and an MRR of $0.6 - 0.8$ indicates that the correct piece is identified most of the time.[2]

	MRR	
Benchmark	**Synthetic**	**Real**
No Repeat	0.77	0.63
Repeat 1	0.76	0.63
Repeat 2	0.75	0.61
Repeat 3	0.75	0.60
D.S. al fine	0.78	0.63

Table 1: System performance on the audio-sheet image retrieval task with all solo piano sheet music images in IMSLP. Results are reported for five different repeat benchmarks and across two types of audio.

4.2 Alignment

We compare the performance of our proposed alignment algorithm with three other baselines. The first baseline is the audio-sheet image alignment system proposed by Dorfer et al. (2017). This system uses a multimodal convolutional neural network to embed short chunks of audio and sheet music into a shared feature space that reflects semantic similarity. This system was slightly modified by replacing DTW with subsequence DTW in order to handle an unknown starting offset. The second baseline is a simple subsequence DTW between the audio bootleg score and sheet music bootleg score. Since subsequence DTW cannot handle discontinuities, we expect these first two baselines to perform poorly on benchmarks with repeats or jumps. The third baseline is a previously proposed Jump DTW algorithm (Fremerey et al., 2010) applied to the audio bootleg score and sheet music bootleg score. Jump DTW is a variant of DTW where additional long-range transitions are inserted into the cost matrix to reflect potential jumps. As with Hierarchical DTW, we consider all line breaks as potential jump locations.

Figure 6 shows the results of the alignment task for both synthetic and real audio. The histogram bars indicate the accuracy with real audio and a scoring collar of $\Delta = 0.5$ seconds. The black horizontal lines (directly above the histogram bar) indicate the accuracy with synthetic audio and a scoring collar of $\Delta = 0.5$ seconds. The grey horizontal lines indicate the accuracy on real

audio with scoring collars of 0 seconds (below the histogram bar) and 1 second (above histogram bar).

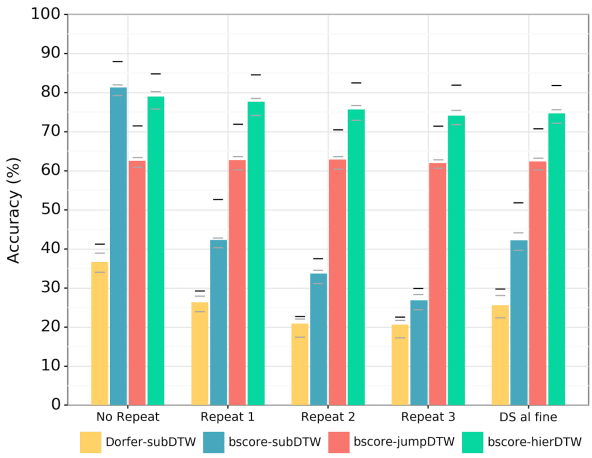

Figure 6: Comparison of system performance on the audio-sheet image alignment task with various types of jumps. The bar levels indicate accuracy with a scoring collar of 0.5 sec on real audio. The short gray lines indicate accuracy with scoring collars of 0 and 1.0 seconds on real audio. The short black line indicates accuracy with a scoring collar of 0.5 seconds on synthetic audio.

There are five things to notice about Figure 6. First, Subsequence DTW with bootleg score features performs the best on the No Repeat benchmark (87.9% for synthetic audio and 81.3% for real audio), but it performs terribly on all benchmarks with any jumps or repeats. Second, Jump DTW performs much worse than subsequence DTW on the No Repeat benchmark (71.5% for synthetic audio and 62.5% for real audio), but its performance is consistent across all five benchmarks (62.5%, 62.7%, 62.9%, 62.0%, 62.4% for real audio). This result matches our expectation since Jump DTW was designed to handle repeats and jumps. Third, Hierarchical DTW consistently outperforms Jump DTW by 11 – 15% across all benchmarks, and it is almost as good as Subsequence DTW on the No Repeat benchmark (84.8% for synthetic audio and 79.0% for real audio). Fourth, the overall trends with synthetic audio match the trends with real audio. This suggests that these trends are not dependent on the audio quality or how well the automatic transcription system performs, but are only dependent on the actual alignment task itself. Fifth, the results for real audio are consistently worse than those for synthetic audio by 4 – 10%. This is again due to additional errors in automatic music transcription.

5. Analysis

In this section, we perform four additional analyses to gain deeper intuition into system performance.

5.1 System Failure Modes

The first analysis focuses on characterizing the failure modes of all 4 alignment systems. We can gain a qualitative understanding of system behavior by investigating individual queries. Figure 7 shows a visualization of the error locations of all systems on two example queries. The top half shows a piece with no repeats, and the bottom half shows a piece with three repeats. Black vertical lines indicate the ground truth line break locations, red vertical lines indicate errors, and blue vertical lines indicate the jump locations. The horizontal axis is the duration of the audio recording.

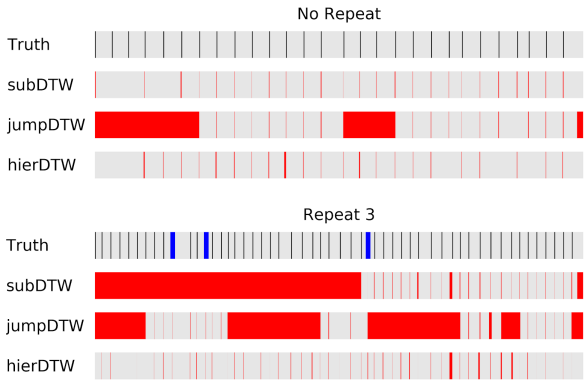

Figure 7: Visualization of system predictions for a query with no repeats (top half) and a query with three repeats (bottom half). The gray stripes represent the duration of the whole audio recording. The black vertical lines show ground truth locations of line breaks, and the red regions indicate times when an incorrect line of sheet music is being shown. The thick blue lines indicate the positions of jumps and repeats.

There are three things to notice about the examples in Figure 7. First, subsequence DTW cannot handle repeats and jumps, so there is a long interval of wrong predictions until the last repeat location (third strip from the bottom). When there are no repeats, however, errors only happen near line breaks (second strip from the top). Second, Jump DTW can handle repeats and jumps (e.g. the first two jumps in the bottom example), but it also seems to insert jumps when none are present. In both examples in Figure 7, the strip for Jump DTW shows alternating regions of correct and incorrect predictions, indicating the existence of spurious jumps. Third, the errors for Hierarchical DTW are clustered around line breaks in both examples. This shows that Hierarchical DTW is able to follow along in the sheet music in both examples, with only slight errors around the line break locations.

Using this type of visualization as a tool, we were able to identify the main failure modes for all four systems. The two baseline systems based on subsequence DTW are not designed to handle repeats and jumps, so they perform poorly on all benchmarks with

jumps. The Dorfer system also performs poorly on the No Repeat benchmark because of staff detection failures. The Jump DTW baseline often inserts spurious jumps when no jumps are present. In one example, it predicted more than 10 jumps in a piece with only two pages. The Hierarchical DTW system has two main failure modes: (a) bootleg score errors arising from half or whole notes, clef changes, and octave markings, and (b) highly repetitive music, which leads to spurious jumps. In both cases, the system is usually able to catch up to the correct alignment a few measures after the problematic sections end.

5.2 Real vs. Synthetic Audio

The second analysis is to compare the errors between real and synthetic audio queries. As shown in Table 1 and Figure 6, the performance with real audio is consistently worse than with synthetic audio due to less reliable automatic music transcription. The real audio recordings consist of three different types: live performances at a concert hall, piano tutorial videos on Youtube, and semi-professional recordings of piano performances at home. In the live concert performances, there are non-music noises like people talking, clapping their hands, or walking around. In the home recordings, there is often talking to introduce the pieces or give instructions (in tutorial videos). These types of non-musical sounds are not present in MIDI-synthesized audio, and they lead to erratic behavior in the automatic music transcription stage. When we compare the 20 pieces with the poorest performance on the retrieval task and the 20 pieces with the poorest performance on the alignment task, 16 pieces are shared in common. This indicates that the problem is with the feature extraction (which is shared by both tasks) and not the search or alignment method.

5.3 Runtime

The third analysis is to characterize the runtime of the alignment and retrieval subsystems. Table 2 shows the runtime of the dynamic n-gram search (retrieval) and Hierarchical DTW (alignment), excluding the feature computation.[3]

There are three things to notice about Table 2. First, the average runtimes in column 2 are quite large, and the standard deviations are enormous. For example, each query in the Repeat x3 benchmark takes an average of 20 minutes to process with a standard deviation of 103 minutes. These numbers suggest the presence of outliers. Indeed, when we investigated individual runtimes more closely, we found that a handful of extremely long pieces skewed the results. These few queries took 100 times the average time to align and 10 times the average time to search. Because the Hierarchical DTW algorithm scales in computation with the number of sheet music lines, the runtime can be extremely large when the PDF contains many other unrelated pieces (e.g. the PDF contains all Chopin Etudes). These findings motivated a followup analysis, in which we measure the runtime of Hierarchical DTW assuming that the correct range of pages has been selected. This leads to our second observation: the runtimes are much lower when only the matching pages are aligned. For example, the average and standard deviation of the runtimes for the Repeat x3 benchmark fall to 80 seconds and 69 seconds, respectively (vs. 20 minutes and 103 minutes). This huge decrease in runtime indicates that Hierarchical DTW is extremely inefficient in solving the retrieval task. The lesson here is clear: Hierarchical DTW should not be used for retrieval but only for alignment. Third, the runtimes for retrieval are acceptably low for an offline task. We can see that the average runtimes range from four to eight seconds across the five benchmarks.

Another useful analysis is to see the breakdown of runtime across the entire system, including all components. Table 3 shows this breakdown for a piece of average length (4 minutes of audio, 13 pages of sheet music). We can see that the automatic music transcription and hierarchical DTW alignment take up the majority of the total runtime. The video generation was done using ffmpeg to generate standalone video files, but this portion of the runtime could be eliminated by designing an application that simply shows a sequence of images for specified durations of time. As described in the previous paragraph, very high runtimes can be avoided by ensuring that the matching PDF does not contain many other unrelated pieces or movements. This could be done by (a) preferring shorter PDFs when selecting the sheet music from a matching piece and/or (b) estimating the matching range of pages in the retrieval task and only performing alignment on the selected pages.

5.4 Repeat Types

The fourth analysis is to study the effect of the jump locations. All previous results in this paper have assumed that jump locations occur at line breaks. In reality, the jump locations may occur anywhere within a line. To study the effect of jump location, we ran an additional set of experiments in which lines of music are randomly selected, and then the jump location is chosen within the selected line according to a random uniform distribution. The bottom of Figure 5 shows an example of a query generated in this manner.

Table 4 compares the effect of jump locations on Hierarchical DTW and Jump DTW. We can see that both systems consistently perform worse by $3-5\%$ when jumps can occur anywhere. When we investigate the qualitative behavior in these scenarios, we find that Hierarchical DTW mostly predicts the jump correctly but has an incorrect alignment for part of the line. For example, if a jump happens in the middle of the third line, Hierarchical DTW usually predicts a jump at either the beginning or end of the third line, and it has

	Alignment (All Pages)		Alignment (Only Matching Pages)		Retrieval	
Benchmark	avg (min)	std (min)	avg (sec)	std (sec)	avg (sec)	std (sec)
No Repeat	8.0	62.8	27.9	17.4	3.95	3.57
Repeat 1	11.2	69.6	32.4	20.3	5.24	3.83
Repeat 2	14.9	99.6	42.1	23.9	6.77	3.33
Repeat 3	20.4	103.4	79.8	69.0	8.34	19.22
D.S. al fine	13.7	78.2	46.3	30.7	5.30	3.49

Table 2: Runtime information for the alignment and retrieval subsystems. These times exclude the time required to extract features.

System Component	AMT	Retrieval	Alignment	Video Generation	Total
Time (sec)	30	5	30	20	85
Percentage (%)	35	6	35	24	100

Table 3: Runtime information for all components of the system on an average length piece (4 minutes of audio, 13 pages of sheet music).

alignment errors for part of the line. While Jump DTW sometimes shows similar behavior, in many cases it inserts a spurious jump during the fragment of the line before the discontinuity. This provides additional evidence that Hierarchical DTW is better than Jump DTW in handling various types and locations of jumps.

6. Conclusion

In this article, we describe a system that can automatically generate score following videos given an audio recording of a piano performance. We combine a large-scale audio-sheet image retrieval system with an alignment approach that can handle unprocessed, real-world data. We propose a novel Hierarchical DTW alignment algorithm that can handle jumps when the locations of the jumps are unknown a priori. We provide extensive empirical validation to show that Hierarchical DTW outperforms previous alignment approaches across a variety of repeat structures, jump locations, and audio conditions.

7. Reproducibility

The data and code for reproducing our results can be found at `https://github.com/anonymized/project`.

8. Competing Interests

The authors have no competing interests to declare.

Notes

[1] There are also several real-time score following algorithms that handle various types of jumps (e.g. (Nakamura et al., 2015; Arzt and Widmer, 2010; Arzt et al., 2008; Pardo and Birmingham, 2005)), though our focus in this work is on the offline context.

[2] Note that when the correct item is rank 2, the reciprocal rank is 0.5.

[3] The dynamic n-gram search is implemented in python, and the Hierarchical DTW is implemented in cython.

References

Arzt, A. and Widmer, G. (2010). Towards effective 'any-time' music tracking. In *Proc. of the Starting AI Researchers' Symposium*.

Arzt, A., Widmer, G., and Dixon, S. (2008). Automatic page turning for musicians via real-time machine listening. In *Proc. of the European Conference on Artificial Intelligence (ECAI)*, pages 241–245.

Damm, D., Fremerey, C., Kurth, F., Müller, M., and Clausen, M. (2008). Multimodal presentation and browsing of music. In *Proc. of the International Conference on Multimodal Interfaces (ICMI)*, pages 205–208.

Dorfer, M., Arzt, A., Böck, S., Durand, A., and Widmer, G. (2016a). Live score following on sheet music images. In *Late Breaking Demos at the International Conference on Music Information Retrieval (ISMIR)*.

Dorfer, M., Arzt, A., and Widmer, G. (2016b). Towards end-to-end audio-sheet-music retrieval. In *Neural Information Processing Systems (NIPS) End-to-End Learning for Speech and Audio Processing Workshop*.

Dorfer, M., Arzt, A., and Widmer, G. (2017). Learning audio-sheet music correspondences for score identification and offline alignment. In *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, pages 115–122.

Dorfer, M., Hajič, J., Arzt, A., Frostel, H., and Widmer, G. (2018a). Learning audio-sheet music correspondences for cross-modal retrieval and piece identification. *Trans. of the International Society for Music Information Retrieval*, 1(1):22–33.

Dorfer, M., Henkel, F., and Widmer, G. (2018b). Learning to listen, read, and follow: Score following as a reinforcement learning game. In *Proc. of the Inter-*

System	Benchmark	Line Breaks Only (%)	Random Location (%)	Difference (%)
JumpDTW	No Repeat	71.5	71.5	N/A
	Repeat 1	71.9	69.0	-2.9
	Repeat 2	70.5	67.2	-3.3
	Repeat 3	71.4	65.8	-5.6
	D.S. al fine	70.7	66.1	-4.6
HierDTW	No Repeat	84.8	84.8	N/A
	Repeat 1	84.5	81.2	-3.3
	Repeat 2	82.5	78.4	-4.1
	Repeat 3	81.9	76.4	-5.5
	D.S. al fine	81.8	77.4	-4.4

Table 4: Assessing the effect of jump locations on the audio-sheet image alignment task. Two conditions are compared: when jump locations occur only at line breaks (column 3) and when jump locations can occur anywhere in a line (column 4). Column 5 shows the performance difference between these two conditions.

national Conference on Music Information Retrieval (ISMIR), pages 784–791.

Dorfer, M., Schlüter, J., Vall, A., Korzeniowski, F., and Widmer, G. (2018c). End-to-end cross-modality retrieval with cca projections and pairwise ranking loss. *International Journal of Multimedia Information Retrieval*, 7(2):117–128.

Fremerey, C., Clausen, M., Ewert, S., and Müller, M. (2009). Sheet music-audio identification. In *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, pages 645–650.

Fremerey, C., Müller, M., and Clausen, M. (2010). Handling repeats and jumps in score-performance synchronization. In *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, pages 243–248.

Fremerey, C., Müller, M., Kurth, F., and Clausen, M. (2008). Automatic mapping of scanned sheet music to audio recordings. In *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, pages 413–418.

Grachten, M., Gasser, M., Arzt, A., and Widmer, G. (2013). Automatic alignment of music performances with structural differences. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 607–612.

Hawthorne, C., Elsen, E., Song, J., Roberts, A., Simon, I., Raffel, C., Engel, J., Oore, S., and Eck, D. (2018). Onsets and frames: Dual-objective piano transcription. In *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, pages 50–57.

Henkel, F., Balke, S., Dorfer, M., and Widmer, G. (2019). Score following as a multi-modal reinforcement learning problem. *Trans. of the International Society for Music Information Retrieval*, 2(1):67–81.

İzmirli, Ö. and Sharma, G. (2012). Bridging printed music and audio through alignment using a mid-level score representation. In *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, pages 61–66.

Jiang, Y., Ryan, F., Cartledge, D., and Raphael, C. (2019). Offline score alignment for realistic music practice. In *Sound and Music Computing Conference*.

Joder, C., Essid, S., and Richard, G. (2011). A conditional random field framework for robust and scalable audio-to-score matching. *IEEE Trans. on Audio, Speech, and Language Processing*, 19(8):2385–2397.

Kurth, F., Müller, M., Fremerey, C., Chang, Y., and Clausen, M. (2007). Automated synchronization of scanned sheet music with audio recordings. In *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, pages 261–266.

Müller, M. (2015). *Fundamentals of Music Processing: Audio, Analysis, Algorithms, Applications*. Springer.

Müller, M. and Appelt, D. (2008). Path-constrained partial music synchronization. In *Proc. of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 65–68.

Müller, M., Arzt, A., Balke, S., Dorfer, M., and Widmer, G. (2019). Cross-modal music retrieval and applications: An overview of key methodologies. *IEEE Signal Processing Magazine*, 36(1):52–62.

Nakamura, T., Nakamura, E., and Sagayama, S. (2015). Real-time audio-to-score alignment of music performances containing errors and arbitrary repeats and skips. *IEEE/ACM Trans. on Audio, Speech, and Language Processing*, 24(2):329–339.

NIST (2016). *NIST Open Speech-Activity-Detection Evaluation Plan*. `https://www.nist.gov/system/files/documents/itl/iad/mig/Open_SAD_Eval_Plan_v10.pdf`.

Pardo, B. and Birmingham, W. (2005). Modeling form for on-line following of musical performances. In *Proc. of the National Conference on Artificial Intelligence*, volume 20, pages 1018–1023.

Thomas, V., Fremerey, C., Müller, M., and Clausen, M.

(2012). Linking sheet music and audio – challenges and new approaches. In *Multimodal Music Processing*, volume 3, pages 1–22. Citeseer.

Tsai, T. (2020). Towards linking the lakh and imslp datasets. In *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 546–550.

Tsai, T., Yang, D., Shan, M., Tanprasert, T., and Jenrungrot, T. (2020). Using cell phone pictures of sheet music to retrieve MIDI passages. *IEEE Transactions on Multimedia*.

Wang, A. (2003). An industrial strength audio search algorithm. In *Proc. of the International Conference on Music Information Retrieval (ISMIR)*.

Yang, D., Tanprasert, T., Jenrungrot, T., Shan, M., and Tsai, T. (2019). Midi passage retrieval using cell phone pictures of sheet music. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 916–923.

Yang, D. and Tsai, T. (2020). Camera-based piano sheet music identification. In *Proc. of the International Conference on Music Information Retrieval (ISMIR)*.