MAKE YOUR OWN ACCOMPANIMENT: ADAPTING FULL-MIX RECORDINGS TO MATCH SOLO-ONLY USER RECORDINGS

TJ Tsai
Harvey Mudd College

Steve Tjoa
Violin.io

Meinard Müller
International Audio Laboratories Erlangen

ABSTRACT

We explore the task of generating an accompaniment track for a musician playing the solo part of a known piece. Unlike previous work in real-time accompaniment, we focus on generating the accompaniment track in an offline fashion by adapting a full-mix recording (e.g. a professional CD recording or Youtube video) to match the user's tempo preferences. The input to the system is a set of recorded passages of a solo part played by the user (e.g. solo part in a violin concerto). These recordings are contiguous segments of music where the soloist part is active. Based on this input, the system identifies the corresponding passages within a full-mix recording of the same piece (i.e. contains both solo and accompaniment parts), and these passages are temporally warped to run synchronously to the solo-only recordings. The warped passages can serve as accompaniment tracks for the user to play along with at a tempo that matches his or her ability or desired interpretation. As the main technical contribution, we introduce a segmental dynamic time warping algorithm that simultaneously solves both the passage identification and alignment problems. We demonstrate the effectiveness of the proposed system on a pilot data set for classical violin.

1. INTRODUCTION

Ima Amateur loves her recording of Itzhak Perlman performing the Tchaikovsky violin concerto with the London Symphony Orchestra. She has been learning how to play the first movement herself, and she would love to play along with the recording. Unfortunately, there are parts of the recording that are simply too fast for her to play along with. She finds an app that can slow down the parts of the Perlman recording that are difficult. All she has to do is upload several solo recordings of herself performing sections of the concerto, along with the original full-mix recording that she would like to play along with. The app analyzes her playing and generates a modified version of the Perlman recording that runs in sync with her solo recordings.

This paper explores the technical feasibility of such an application. In technical terms, the problem is this: given a full-mix recording and an ordered set of solo-only recordings that each contain a contiguous segment of music where the soloist is active, design a system that can time-scale modify the full-mix recording to run synchronously with the solo recordings. [1]

There are three main technical challenges underlying this scenario. The first challenge is to identify the passages in the full-mix recording that correspond to the solo-only recordings. The second challenge is to temporally align the corresponding passages in the full-mix and solo recordings. The third challenge is to time-scale modify the full-mix recording to follow the calculated alignment without changing the pitch of the original recording. This paper focuses primarily on the first two challenges, and it assesses the technical feasibility of solving these problems on a pilot data set. The main technical contribution of this work is to propose a segmental dynamic time warping (DTW) algorithm that simultaneously solves the passage identification and temporal alignment problems. We will simply adopt an out-of-the-box approach to solve the third challenge.

The idea of generating accompaniment for amateur musicians has been explored in two different directions. On one end of the spectrum, companies have explored fixed accompaniment tracks. Some examples include the popular Aebersold Play-A-Long recordings for jazz improvisation and Music Minus One for classical music. The benefit of fixed accompaniment tracks is their simplicity – all you need is a device that can play audio. The drawback of fixed accompaniment tracks is their lack of adaptivity – they do not respond or adapt to the user's playing in any way. On the other end of the spectrum, academics have explored real-time accompaniment (e.g. see work by Raphael [23] [24] and Cont [3]). These are complex systems that can track a musician's (or group's) playing and generate accompaniment in real-time. The benefit of real-time accompaniment is the adaptivity of the system. The drawbacks of real-time accompaniment systems are that they are not easy to use for the general population (e.g. require software packages on a laptop) and may not be very expressive (e.g. sound like MIDI). Also, for the purposes of academic study, another drawback is the difficulty of evaluating such a system in an objective way. Because the user and the accompaniment system influence each other in real-time, it is difficult to decouple the effect of one from the other. When there are errors, for example, it is difficult to say whether the error is because the accompaniment

[1] Without changing the pitch, of course!

system failed, the user failed to respond appropriately, or some combination of both.

This work explores the realm in between these two extremes. Like fixed accompaniment tracks, the proposed system has the benefit of simplicity – the user does not need any specialized software or hardware, but simply receives an audio track that can be played on any audio device. Like real-time accompaniment, the proposed system has the benefit of (partial) adaptivity – the system tailors the accompaniment track to the user's playing in an offline manner. This middle realm has several additional benefits. Because the user and the accompaniment are no longer coupled in real-time, we can measure how well the accompaniment system "follows" the user's playing with objective metrics. Another benefit is that the offline nature of this system makes it suitable for a client-server model, which is ideal for the envisioned app. Lastly, by approaching this problem through adapting an existing recording, we can also potentially get the benefit of a very musical and expressive accompaniment track (assuming we don't introduce too many artifacts from time-scale modification).

The two challenges we will focus on – passage identification and temporal alignment – are closely related to previous work in audio matching and music synchronization. The passage identification problem has strong similarities to audio matching, where the goal is to identify a given passage in other performances of the same (usually classical) work. Previous work has introduced robust features for this task [20] and efficient ways to handle global tempo variations such as using multiple versions of a query that have been tempo-adjusted [19]. Subsequent work has explored the use of indexing techniques to scale the system to large data sets [14] [2]. The temporal alignment problem has strong similarities to music synchronization, where the goal is to temporally align two performances of the same piece. The bread-and-butter approach is to apply DTW with suitably designed features [12] [4] [10]. One problem with this approach is that the memory and computation requirements increase quadratically as the feature sequences increase in length. Many variants have been proposed to mitigate this issue, including limiting the search space to a band [25] or parallelogram [13] around the cost matrix diagonal, doing the time-warping in an online fashion [5] [15], or adopting a multiscale approach that estimates the alignment at different granularities [26] [21] [9]. Other variants tackle issues like handling repeats [11], identifying partial alignments between recordings [17] [18], dealing with memory constraints [22], and taking advantage of multiple recordings [27] [1].

Though similar, the proposed scenario differs from most previous work in three important ways. First, we are matching solo-only recordings to full-mix recordings (i.e. solo and accompaniment). Most work in audio matching and music synchronization assumes that the recordings of interest are different performances of the same piece, and therefore have the same audio sources. One could think of the current scenario as audio matching with very high levels of additive noise (i.e. the accompaniment). Second, the task is offline but there are still stringent runtime constraints. In music synchronization, the best approach is the one with the highest alignment precision, and we are willing to accept significant runtimes since the task is offline. In the current scenario, however, the runtime is a very important factor because the application is user-facing. A user will not be willing to wait 30 seconds for the accompaniment track to be generated. For this reason, in this paper we will not consider any approaches to these two challenges that require more than 5-6 seconds of runtime. Third, the current scenario deals with consumer-produced recordings. Much previous work focuses on album tracks from professional CDs and professional musicians. In contrast to this, amateur musicians will play wrong notes, count incorrectly, rush, and play out of tune. These issues will be important factors affecting system performance.

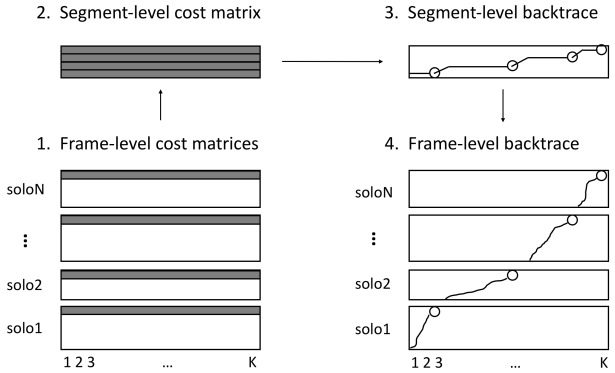

Figure 1. A graphical overview of the segmental DTW algorithm for aligning an ordered set of solo recordings against a full-mix recording. Rows correspond to solo recording frames and columns correspond to full-mix recording frames. Time increases from bottom to top and left to right. In this example, $N = 4$.

This paper is structured around our main goal: to assess the technical feasibility of solving the passage identification and temporal alignment problems in a robust and efficient manner. Section 2 describes our system, including an explanation of the proposed segmental DTW algorithm. Section 3 discusses the experimental setup. Section 4 presents empirical results of our experiments on the pilot data set. Section 5 investigates several questions of interest to gain more intuition into system performance. Section 6 concludes the work.

2. SYSTEM DESCRIPTION

We describe the proposed system in three parts: the segmental DTW algorithm, the features, and the time-scale modification.

2.1 Segmental DTW Algorithm

There are four main steps in the segmental DTW algorithm, each explained below.

Step 1: Frame-level cost matrices. The first step is

to compute a subsequence DTW cumulative cost matrix for each solo segment. Subsequence DTW is a variant of the regular DTW algorithm in which one of the recordings (the query) is assumed to only match a *section* of the other recording (the reference), rather than matching the entire recording from beginning to end. This can be accomplished by allowing the query to *begin* matching anywhere in the reference without penalty, and allowing the query to *end* matching anywhere in the reference without penalty. We allow the following $(query, reference)$ steps in the dynamic programming stage: $(1, 1)$, $(1, 2)$, and $(2, 1)$. These steps have weights of 1, 1, and 2, respectively. [2] This set of steps assumes that the instantaneous tempo in the query and reference will differ at most by a factor of 2. For more details about subsequence DTW, see chapter 7 in [16]. In the case of our proposed algorithm, we compute the subsequence DTW cumulative cost matrix but refrain from backtracing until step 4. Rather than backtracing from the local optimum in each cumulative cost matrix, we will instead backtrace from the element on the globally optimum path. This globally optimum path will be determined in steps 2 and 3.

Step 2: Segment-level cost matrix. The second step is to compute a cumulative cost matrix of global path scores across all solo segments. This can be done in two substeps. The first sub-step is to create a matrix that contains the last row of each subsequence cumulative cost matrix from step 1. [3] This matrix will have N rows and K columns, where N is the number of solo segments and K is the number of frames in the reference (i.e. full-mix) recording. Note that this matrix is analogous to a pairwise cost matrix, where instead of pairwise frame-level costs we have segment-level subsequence path costs. The second sub-step is to compute a (segment-level) cumulative cost matrix on this (segment-level) pairwise cost matrix by doing dynamic programming. This dynamic programming step differs from regular DTW dynamic programming in one important way. Unlike most scenarios where the set of possible transitions is fixed regardless of position in the cost matrix, here the possible transition steps changes from row to row. Specifically, for an element in row n, the two possible transitions are $(0, 1)$ and $(1, \frac{L_{n+1}}{2})$, where L_{n+1} is the length (in frames) of the $(n + 1)^{\text{th}}$ solo segment. The weights on these two transitions are 0 and 1, respectively. In words, we are looking for the N elements in the segment-level pairwise cost matrix (one per row) that have the minimum total path score under two constraints: (1) they are consistent with the given ordering (i.e. segment n comes before segment $n + 1$), and (2) elements in adjacent rows must be separated by a minimum distance, which is determined by the length of the solo segment and the maximum tempo difference in the subsequence DTW step (in this case, a factor of 2).

Step 3: Segment-level backtrace. The third step is to backtrace through the segment-level cumulative cost matrix. We start at the last element of the matrix (i.e. the upper right hand corner) and backtrace until we reach the first element of the matrix (i.e. the lower left hand corner). Note that the $(0, 1)$ steps with 0 weight allow for skipping portions of the full-mix recording without penalty. The $(1, \frac{L_{n+1}}{2})$ transitions in the backtraced path indicate the element in each row that contributes to the globally optimal path.

Step 4: Frame-level backtrace. The final step is to backtrace through each subsequence DTW cumulative cost matrix from step 1, where we begin the backtracing at the elements selected in step 3. These elements have been selected to optimize a global path score across all solo segments, rather than a local path score across a single solo segment. After performing this frame-level backtrace step, we have achieved our desired goal: identifying both segment-level and frame-level alignments for each solo segment.

Figure 1 shows a graphical summary of the segmental DTW algorithm. In this figure, rows correspond to different solo segment frames and columns correspond to different full-mix frames. Time increases from bottom to top and from left to right. The four rectangles in the lower left are the frame-level cumulative cost matrices for each solo recording. The segment-level cost matrix (top left) is constructed by aggregating the last row from each frame-level cumulative cost matrix (highlighted in dark gray). We then backtrace at the segment level, and use the predicted segment ending points to backtrace at the frame level. The final predicted alignments are shown in the lower right. Note that the proposed system only indicates how the full-mix recording should be warped during the segments of the piece when the soloist is playing. One could interpolate the tempo for the other segments.

2.2 Features

The segmental DTW algorithm is compatible with any frame-based feature and cost metric. For the experiments in this paper, we computed L2-normalized chroma features every 22 ms and used a cosine distance metric. This combination was selected for two practical reasons. First, we wanted to demonstrate the segmental DTW algorithm with a standard feature, so as not to conflate the performance benefits of both a new matching algorithm and a novel (or less widely used) feature. Second, this combination allows the subsequence DTW cost matrices to be computed very efficiently with simple matrix multiplication. Given the constraints on runtime of this consumer-facing application, efficiency is an important consideration. We selected the feature rate to ensure that the average time required to align a single query (i.e. multiple solo recordings against a full-mix recording) was under 6 seconds. This threshold could be set arbitrarily depending on how long we are willing to make the user wait. In the discussion section, we will compare our main results with a system that uses more state-of-the-art features, which were developed in an offline context where runtime is not a significant consideration. These latter features can provide a lower bound on

[2] Note that the $(2, 1)$ step should be weighted double to prevent degenerate matchings to very short sections.

[3] Here, we assume that rows correspond to different query frames, and columns correspond to different reference frames.

Composition	full	solo	avgLen	segs
Seitz concerto no2, mv3	5	5	187s	5
Bach double concerto, mv1	5	5	250s	5
Vivaldi concerto in a, mv1	5	5	229s	5
Veracini sonata in d, mv4	3	4	223s	4

Table 1. Summary of the pilot data set. Each row indicates the number of full-mix and solo recordings, the average length, and the number of segments in the composition.

error rate when we ignore runtime constraints.

2.3 Time-scale modification

The goal of the time-scale modification (TSM) step is to stretch or compress the duration of a given audio signal while preserving properties like pitch and timbre. Typically, TSM approaches stretch or compress an audio signal in a linear fashion by a constant stretching factor. In our scenario, we need to stretch the full-mix recording according to the solo-mix alignment, which leads to a non-linear time-stretch function. To deal with non-linear stretches, we apply the strategy described in [8], where the positions of the TSM analysis frames are specified according to the time-stretch function instead of a constant analysis hop-size.

To attenuate artifacts and to improve the quality of the time-scale modified signal, we use a recent TSM approach [7] that involves harmonic-percussive separation and combines the advantages of a phase-vocoder TSM approach (preserving the perceptual quality of harmonic signal components) and a time-domain TSM approach (preserving transient-like percussive signal components). An overview of different TSM procedures can be found in [6, 8].

3. EXPERIMENTAL SETUP

The experimental setup will be described in three parts: the data collection, the data preparation, and the evaluation metric.

3.1 Data Collection

Our data collection process was dictated by practicality. In order to evaluate the proposed system, we need two different types of audio data: full-mix recordings and solo recordings. Clearly, the full-mix recordings are in abundant supply and can be selected from any professional CD recording or Youtube video. The solo recordings, however, are much more difficult to find, as musicians typically do not record performances that are missing the accompaniment part. Our solution to this problem was to focus data collection efforts on a small subset of pieces from the highly popular Suzuki violin method. The Suzuki method prescribes a specific sequence of violin works in order to develop a violinist's mastery of the instrument. Because of the popularity of the Suzuki method, we were able to find Youtube videos of violinists performing the solo parts (in isolation) from several works. Some of these recordings are violin teachers demonstrating how to perform a piece. Some recordings are young adults wishing to document their progress on the violin. Other recordings are doting parents trying to show off their talented children.

Table 1 shows a summary of the audio recordings. The data set contains four violin pieces or movements selected from Suzuki books five and six. For each piece, we collected multiple full-mix recordings and solo recordings from Youtube. By focusing on annotating multiple recordings of the same piece, we can make the most of the limited amount of (annotated) data by considering different combinations of full-mix and solo recordings. At the same time, we wanted several pieces of music from different composers and periods, so as to avoid a composer-specific bias. The recordings range in length from 161 to 325 seconds, and they range in quality from cell phone videos to professionally recorded performances. All audio tracks were converted to mono wav format with 22050 Hz sampling rate. In total, there is approximately 2 hours and 20 minutes of annotated audio data.

3.2 Data Preparation

Once the audio data was collected, there were two additional steps needed to prepare the data for use in our experiments.

The first preparation step was to generate beat-level annotations. The annotations were done in SonicVisualizer [4] by three different individuals with extensive training in classical piano. We kept only those beats that had two or more independent annotations, and we use the mean annotated time as the ground truth.

The second data preparation step was to divide the solo recordings into segments. Recall that the input to the system is a set of contiguous segments of music where the soloist is active. Each segment is specified by a pair of unique identifiers (e.g. start at measure 5 beat 1 and end at measure 37 beat 4), and the segments are non-overlapping. For each composition, we manually selected segments by identifying natural breakpoints where a violinist would likely end a segment, such as section boundaries or the start/end of a long rest.

We can summarize the prepared data set as follows. Each query in the benchmark is a pairing of a full-mix recording and a solo recording (i.e. the 4-5 segments from a solo recording). There are thus a total of 87 queries in the benchmark. This is clearly not a large data set. It is meant to serve as a pilot data set to assess the feasibility of the proposed system.

3.3 Evaluation Metric

In this paper, we will focus only on the aspects of the system that can be evaluated objectively: the segment boundaries and frame-level alignments. To evaluate segment boundary predictions, we compare the predicted and ground truth boundary points for each solo segment, and

[4] http://www.sonicvisualiser.org/

tolerance	global	subseq	segmental
1s	40.2%	8.4%	2.2%
2s	20.2%	6.1%	0.0%
5s	14.9%	6.1%	0.0%
10s	9.3%	6.1%	0.0%

Table 2. Boundary prediction error rates for global, subsequence, and segmental DTW algorithms. Each entry indicates the percentage of predicted boundary points that are incorrect at a specified allowable error tolerance.

then determine what fraction of predicted boundary points are correct (or incorrect) for a given allowable error tolerance. To evaluate frame-level alignments, we compare predicted and ground truth timestamps in the full-mix recording that correspond to the annotated beat locations in the solo segments. [5] We then determine what fraction of alignments are correct (or incorrect) for a given allowable error tolerance. By considering a range of different error tolerances, we can determine an error tolerance curve. Note that the error tolerances for the segment boundary metric are much larger than the error tolerances for frame alignment, since the former is measuring retrieval at the segment level.

4. RESULTS

To assess the effectiveness of the proposed segmental DTW algorithm, we compared its performance against two other baseline systems. The first baseline system is to simply concatenate all of the solo audio segments and perform a single global DTW against the full-mix recording. For this baseline system, we use transition steps $(0, 1)$, $(1, 0)$, and $(1, 1)$ in order to handle the discontinuities between solo segments. All steps are given equal weight. The second baseline system is to perform subsequence DTW on each solo segment independently, where the best locally optimal path in each cost matrix is taken as the predicted segment-level and frame-level alignment. In order to make the comparison between systems fair, all three systems use the same chroma features. Any differences in performance should thus reflect the effectiveness of the matching algorithm.

Table 2 compares the performance of the three systems on passage identification. The rows in the table show the percentage of predicted boundary points that are incorrect at four different error tolerances. The three rightmost columns compare the performance of the global DTW baseline ('global'), the subsequence DTW baseline ('subseq'), and the proposed segmental DTW algorithm ('segmental').

There are three things to notice about Table 2. First, the error rates clearly decrease from left to right. Thus, the relative performance of the three algorithms is clear: global DTW performs worst, subsequence DTW performs better, and segmental DTW performs best. Second, subsequence

[5] Since the annotated beat locations generally fall between frames, we use simple linear interpolation between the nearest predicted alignments.

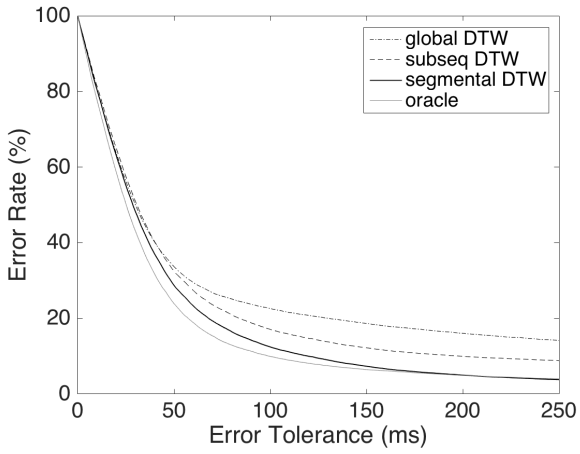

Figure 2. Error tolerance curves for the global, subsequence, and segmental DTW algorithms. Each point on a curve indicates the percentage of predicted beat alignments that are incorrect for a given error tolerance. An additional curve is shown for an oracle system, which provides a lower bound on performance.

DTW reaches an asymptotic error rate of 6.1%. These errors are passages that the subsequence DTW algorithm is matching incorrectly because it fails to take into account the temporal ordering of the solo segments. For example, it incorrectly matches the main theme to the recapitulation or matches repeated segments to the wrong repetition. Better features are unlikely to fix these errors. Third, the segmental DTW algorithm has perfect performance for error tolerances of 2 seconds and above. This suggests that the 2.2% of errors at a 1 second error tolerance are an indication of poor alignments but correctly identified passages. We will investigate these errors in the discussion section.

Figure 2 compares the performance of the three systems on temporal alignments. The figure shows the error tolerance curves for error tolerances ranging from 0 to 250 ms. Each point on a curve indicates the percentage of predicted beat timestamps that are incorrect at a given error tolerance. There is also a curve for an oracle system, which will be explained in section 5.2.

There are three things to notice about Figure 2. First, the curves are identical for error tolerances < 25 ms. This indicates that when an algorithm is "locked onto" a signal, the limit to its precision is the same for all three algorithms. This is what we expect, since all three algorithms are based on the same fundamental dynamic programming approach and use the same features. This is a realm where the segmental DTW algorithm does not help, but where better features are needed to improve performance. Second, the curves begin to diverge significantly for error tolerances > 50 ms. This is a realm where the segmental DTW algorithm provides significant benefit to system performance. For example, at 100 ms error tolerance, the segmental DTW algorithm improves the error rate from 22.6% and 17.1% to 12.4%. Third, the curves do not intersect. In other words, the segmental DTW algorithm provides uni-

lateral benefit across all error tolerances.

5. DISCUSSION

In this section, we investigate three questions of interest that will give deeper insight into system performance.

5.1 Investigation of Boundary Errors

The first question of interest is: "What is causing the segment boundary errors?" We saw from Table 2 that 2.2% of predicted segment boundaries are incorrect at an error tolerance of 1 second. We investigated all of these errors to determine the root cause of the problem.

There are three main observations we can make from our investigations of segment boundary errors. First, most segment boundary errors are a result of a mistake on the part of the musician or musical group. In one instance, the violin player messes up and stops playing for 3-4 beats at the end of a phrase. In another instance, the group is very out of sync on the last note. These two specific mistakes caused more than 50% of the segment boundary errors, since a single mistake will cause errors on all of the queries that contain the recording. Second, the maximum tempo ratio of 2x imposed by the DTW step sizes causes errors when the instantaneous tempo difference is extreme. One example came from a recording in which the violinist slows down very significantly (much more than 2x) because he or she is clearly struggling with the rolled chords in the last several measures. Another example came from a recording that had a very pronounced rubato at the end of the piece. This caused problems when the recording was paired with a performance that incorporated very little rubato at the end. Third, all of the segment boundary errors were predictions of the end of a segment. The DTW algorithm (and its variants) do well in smoothing out errors in the beginning and middle of segments, but it often fails at the end of a segment because there is no signal "on the other side" to smooth out the prediction.

5.2 Lower Bound on Error Rate

The second question of interest is: "What is the lower bound on error rate?" In other words, what is the best error rate that we could hope to achieve given a current state-of-the-art alignment system? In order to answer this question, we ran an experiment with two major changes. The first change is that we assume this system is an oracle and knows the ground truth segment boundaries for each solo segment. The second change is that we use an alignment system [22] that was designed to maximize alignment precision in an offline context. Note that this oracle system requires more than 45 sec on average to align each query (i.e. align multiple solo recordings against a full-mix recording), so it would not be suitable given the runtime constraints of our user-facing application. (In contrast, our proposed system required an average of 5.20 sec.) Thus, we can interpret the performance of the oracle system as a lower bound on error rate when runtime constraints are ignored.

The performance of this oracle system is shown in Figure 2 (overlaid on the same figure from the results section). There are two things to point out about this lower bound curve. First, the proposed system approximately achieves the lower bound for error tolerances > 175 ms. Second, the lower bound shows the most room for improvement in the 50 to 100 ms error tolerance range. For a 75 ms error tolerance, the proposed system and oracle system achieve error rates of 17.8% and 14.0%, respectively.

5.3 Listening to the Accompaniment Track

The third question of interest is: "How does the time-scale modified accompaniment track actually sound?" One useful way we can get a sense of how well the accompaniment is "following" the solo recordings is to create a stereo track in which one channel contains the unchanged solo recording and the other channel contains the time-stretched accompaniment track. By listening to both tracks simultaneously, we can gain an intuitive sense of how well the system is doing. We have posted several samples of these stereo recordings for interested readers. [6]

There are three qualitative observations we can make regarding these informal listening tests. First, the system performs much more erratically when the solo part is not dominant. This was particularly a problem for the Bach double violin concerto since there are two equally important violin parts. During sections when the 2^{nd} violin part is the dominant voice, the accompaniment track has significantly more time-warping artifacts. Second, the system handles rapid notes very well and prolonged notes very poorly. When the solo part is holding a single long note, the accompaniment track would sometimes have very severe temporal distortion artifacts. Third, the time-stretched accompaniment track often has a "jerky" tempo, especially in situations when the solo part has a long note. The accompaniment track is clearly tracking the solo recordings, but it often has short, sudden bursts of tempo speedups and equally short, sudden bursts of tempo slowdowns. One way to address this issue would be to do some type of temporal smoothing of the predicted alignment in order to ensure a more natural sounding accompaniment.

6. CONCLUSION

We have described a system that time-scale modifies an existing full-mix recording to run synchronously to an ordered set of solo-only user recordings of the same piece. We propose a segmental DTW algorithm that simultaneously solves the passage identification and temporal alignment problems, and we demonstrate the benefit of this algorithm over two other baseline systems on a pilot data set of classical violin music. Areas of future work include expanding the pilot data set, exploring features that are both computationally efficient and well-suited to the asymmetric nature of the scenario, and investigating pre-processing steps for solo detection and separation.

[6] `http://www.anonymizedurl.edu`

7. ACKNOWLEDGMENTS

Thanks to Zhepei Wang and Thitaree Tanprasert for helping with the data annotation. The International Audio Laboratories Erlangen are a joint institution of the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) and Fraunhofer Institut für Integrierte Schaltungen IIS.

8. REFERENCES

[1] Andreas Arzt and Gerhard Widmer. Real-time music tracking using multiple performances as a reference. In *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, pages 357–363, Málaga, Spain, 2015.

[2] Michael A. Casey, Christophe Rhodes, and Malcolm Slaney. Analysis of minimum distances in high-dimensional musical spaces. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(5):1015–1028, 2008.

[3] Arshia Cont, José Echeveste, and Jean-Louis Giavitto. The Cyber-Physical System Approach for Automatic Music Accompaniment in Antescofo. In *Acoustical Society Of America*, Providence, Rhode Island, United States, May 2014.

[4] Roger B. Dannenberg and Ning Hu. Polyphonic audio matching for score following and intelligent audio editors. In *Proc. of the International Computer Music Conference (ICMC)*, pages 27–34, San Francisco, USA, 2003.

[5] Simon Dixon. Live tracking of musical performances using on-line time warping. In *Proc. of the 8th International Conference on Digital Audio Effects*, pages 92–97. Citeseer, 2005.

[6] Mark Dolson and Jean Laroche. Improved phase vocoder time-scale modification of audio. *IEEE Transactions on Speech and Audio Processing*, 7(3):323–332, 1999.

[7] Jonathan Driedger and Meinard Müller. Improving time-scale modification of music signals using harmonic-percussive separation. *IEEE Signal Processing Letters*, 21(1):105–109, 2014.

[8] Jonathan Driedger and Meinard Müller. A review on time-scale modification of music signals. *Applied Sciences*, 6(2):57–82, February 2016.

[9] Sebastian Ewert and Meinard Müller. Refinement strategies for music synchronization. In *Proceedings of the International Symposium on Computer Music Modeling and Retrieval (CMMR)*, volume 5493 of *Lecture Notes in Computer Science*, pages 147–165, Copenhagen, Denmark, May 2008.

[10] Sebastian Ewert, Meinard Müller, and Peter Grosche. High resolution audio synchronization using chroma onset features. In *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 1869–1872, Taipei, Taiwan, April 2009.

[11] Christian Fremerey, Meinard Müller, and Michael Clausen. Handling repeats and jumps in score-performance synchronization. In *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, pages 243–248, Utrecht, The Netherlands, 2010.

[12] Ning Hu, Roger B. Dannenberg, and George Tzanetakis. Polyphonic audio matching and alignment for music retrieval. In *Proc. of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, New Paltz, NY, USA, 2003.

[13] Fumitada Itakura. Minimum prediction residual principle applied to speech recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 23(1):67–72, 1975.

[14] Frank Kurth and Meinard Müller. Efficient index-based audio matching. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(2):382–395, February 2008.

[15] Robert Macrae and Simon Dixon. Accurate real-time windowed time warping. In *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, pages 423–428, 2010.

[16] Meinard Müller. *Fundamentals of Music Processing: Audio, Analysis, Algorithms, Applications*. Springer, 2015.

[17] Meinard Müller and Daniel Appelt. Path-constrained partial music synchronization. In *Proc. of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 1, pages 65–68, Las Vegas, Nevada, USA, April 2008.

[18] Meinard Müller and Sebastian Ewert. Joint structure analysis with applications to music annotation and synchronization. In *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, pages 389–394, Philadelphia, Pennsylvania, USA, September 2008.

[19] Meinard Müller, Frank Kurth, and Michael Clausen. Audio matching via chroma-based statistical features. In *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, pages 288–295, London, UK, 2005.

[20] Meinard Müller, Frank Kurth, and Michael Clausen. Chroma-based statistical audio features for audio matching. In *Proc. of the Workshop on Applications of Signal Processing (WASPAA)*, pages 275–278, New Paltz, New York, USA, October 2005.

[21] Meinard Müller, Henning Mattes, and Frank Kurth. An efficient multiscale approach to audio synchronization. In *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, pages 192–197, Victoria, Canada, October 2006.

[22] Thomas Prätzlich, Jonathan Driedger, and Meinard Müller. Memory-restricted multiscale dynamic time warping. In *Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 569–573, Shanghai, China, 2016.

[23] Christopher Raphael. Music plus one and machine learning. In *Proc. of the International Conference on Machine Learning (ICML)*, pages 21–28, 2010.

[24] Christopher Raphael and Yupeng Gu. Orchestral accompaniment for a reproducing piano. In *Proc. of the International Computer Music Conference (ICMC)*, 2009.

[25] Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(1):43–49, 1978.

[26] Stan Salvador and Philip Chan. FastDTW: Toward accurate dynamic time warping in linear time and space. In *Proc. of the KDD Workshop on Mining Temporal and Sequential Data*, 2004.

[27] Siying Wang, Sebastian Ewert, and Simon Dixon. Robust and efficient joint alignment of multiple musical performances. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(11):2132–2145, 2016.