Polyphonic Audio Matching for Score Following and Intelligent Audio Editors

Roger B. Dannenberg and Ning Hu

School of Computer Science, Carnegie Mellon University
email: dannenberg@cs.cmu.edu, ninghu@cs.cmu.edu, *web:* www.cs.cmu.edu/~music

Abstract

Getting computers to understand and process audio recordings in terms of their musical content is a difficult challenge. We describe a method in which general, polyphonic audio recordings of music can be aligned to symbolic score information in standard MIDI files. Because of the difficulties of polyphonic transcription, we perform matching directly on acoustic features that we extract from MIDI and audio. Polyphonic audio matching can be used for polyphonic score following, building intelligent editors that understand the content of recorded audio, and the analysis of expressive performance.

1 Introduction

Many interesting music processing tasks rely upon symbolic representations of music. Scores, MIDI, and note lists can be manipulated in many interesting ways. In contrast, audio data is relatively opaque and unstructured, limiting how we can use it. For example, the task "play from measure 3" is simple with a score representation but extremely problematic with just audio data.

In many applications, one has access to both symbolic and audio representations. For example, in a score-following application, the task is to match audio to symbolic data. Similarly, in studio recordings, performers often read music notation from a music notation program, where the music exists in symbolic form.

If we could align polyphonic audio with symbolic notation, we could enable many interesting applications. Of particular interest are *computer accompaniment*, in which a computer synchronizes computer-generated audio with live performers, and *intelligent audio editors* where audio from recordings can be automatically aligned with notation. Another application is the *analysis of expressive performance*, for example comparing expressive tempo changes in different recordings of Beethoven symphonies.

There are many interesting applications of music transcription. Our work recognizes that in many cases, full transcription is not necessary, because a transcription *of the score* already exists. By aligning the performance with the score, we obtain the equivalent of a transcription of the performance. Unlike automated polyphonic music transcription, which has been largely unsuccessful to date, our techniques are effective and fairly easy to implement.

The problem is simply stated: given an audio recording and a corresponding standard MIDI file, find the best correspondence between them. We assume that the timing of the MIDI data does not correspond exactly to the timing of the audio recording. Otherwise, the problem would be trivial. For example the MIDI data may be a "flat" performance using exact tempo markings from a score, while the audio may be an expressive performance by musicians.

A "standard" approach to this problem might be to perform some sort of polyphonic transcription on the music and then use a symbolic score-matching algorithm (Bloch & Dannenberg, 1985). Unfortunately, accurate polyphonic transcription is yet to be achieved, and the error rates of the best systems are sufficiently high as to make matching difficult in many cases.

We present an alternative in which matching is performed on acoustic features rather than symbolic ones. In the simplest approach, we convert MIDI data to audio, extract audio features, and use a dynamic time warping algorithm (Sankoff & Kruskal, 1983) to align the resulting sequences. The result tells us the correspondence between the audio and MIDI data. We can optimize this approach by extracting acoustic features directly from MIDI.

Our work is closely related to that of Orio and Schwarz (2001), who also use dynamic time warping to align polyphonic music to scores. They obtain accurate alignment using small (5.8ms) analysis windows. Orio and Schwarz use a measure called Peak Structure Distance, which is derived from the spectrum of audio and from synthetic spectra computed from score data, whereas we use the chromagram, described below. Another novel aspect of our work is that we have demonstrated success with popular vocal music, in spite of obvious discrepancies between MIDI data and vocal performance.

In the next section, we describe the matching process in more detail. In Section 3, we discuss an optimization that bypasses the synthesis of audio from MIDI. Since our approach never fails to generate a maximum likelihood match, it is important to evaluate whether the matches are really correct in a musical sense. Section 4, presents several evaluation strategies and the results we have obtained. Section 5 describes some possible applications and future work. Finally, we present a summary and conclusions in Section 6.

2 Matching Audio to MIDI

Our task is to align MIDI data with audio data. This is accomplished in three steps. First, we convert MIDI data to audio using a MIDI synthesizer. For these experiments, we use Timidity (Toivonen & Izumo, 1999), which generates audio files from standard MIDI files.

2.1 The Chroma Representation

The second step is to convert audio data into *discrete chromagrams*: sequences of chroma vectors. The chroma vector representation is a 12-element vector, where each element represents the spectral energy corresponding to one pitch class (i.e. C, C#, D, D#, etc.). To compute a chroma vector from a magnitude spectrum, we assign each bin of the FFT to the pitch class of the nearest step in the chromatic equal-tempered scale. Then, given a pitch class, we average the magnitude of the corresponding bins. The 12 values that result are called the *chroma vector*. Each chroma vector in this work represents 0.25 seconds of audio data (non-overlapping).

The exact details of the chroma computation concerning how to deal with low-frequency bins that span more than one half-step, whether to average magnitude or sum power, etc., are not critical. Our work differs from the original chroma vector work (Bartsch & Wakefield, 2001) in that we use linear rather than logarithmic amplitudes.

The reason chroma *might* be good for this application is that the chroma vector depends on the pitch classes of strong partials in the signal. By design, chroma vectors are sensitive to prominent pitches and chords, but since all spectral energy is collapsed into one octave, chroma vectors are not very sensitive to spectral shape. Since we are comparing MIDI data to acoustic data, it is good to focus on pitch classes and more-or-less ignore details of timbre and spectral shape. Further experiments (Hu, Dannenberg & Tzanetakis, 2003) proved that chromagrams are the best choice for this task among several common acoustic features including MFCC (Logan, 2000) and Pitch Histograms (Tzanetakis, Ermolinskyi & Cook, 2002).

2.2 Comparing and Aligning Chroma

After computing chroma for each audio signal (one of which is derived from MIDI data), we obtain two sequences of chroma vectors. We want to find a correspondence between the two sequences such that corresponding chroma vectors are similar. One way to think about this problem is that we will modify the tempo of the MIDI data in order to obtain the best agreement between the resulting sequences of chroma vectors.

We must first define what “agreement between chroma vectors” means. We first normalize the vectors to have a mean of zero and a variance of one. The normalization reduces differences due to absolute magnitude (loudness), which seems to be a good idea because loudness in MIDI files is rarely calibrated to control absolute levels. We then calculate the Euclidean distance between the vectors. The distance is zero if there is perfect agreement. Figure 1 shows a *similarity matrix* where the vertical axis is a time index into the acoustic recording, and the horizontal axis is a time index into the audio rendered from MIDI. The intensity of each point is the distance between the corresponding chroma vectors, where black represents a distance of zero.

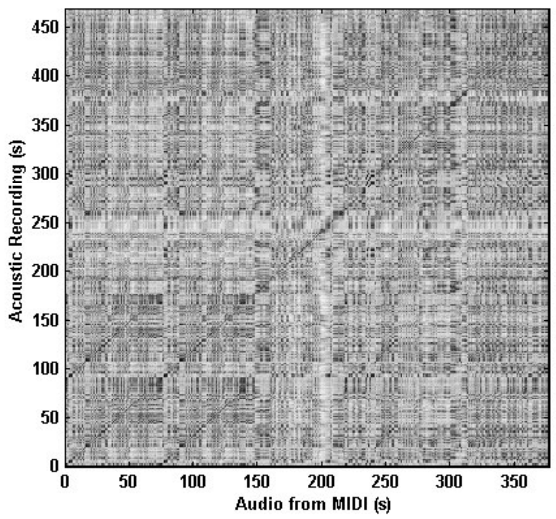
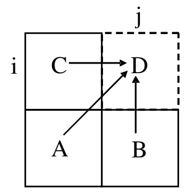


Figure 1. Similarity Matrix for Beethoven’s 5th Symphony, first movement.

The dark diagonal represents a path where the chroma vectors are near one another. This path is the alignment we are after. Notice that the tempo of the MIDI performance (Viens, 2000) is substantially faster than the audio (North German Radio Symphony Orchestra, 1992), so the acoustic recording is much longer than the audio from MIDI. Also notice that the repetition at the beginning of the movement yields additional off-diagonal paths where the first time of the acoustic data matches the second time of the MIDI data and vice versa.

Although the path is visually clear in the figure, we need an automated method to locate the path. Alignment is computed using a dynamic time warping (DTW) algorithm. DTW computes a path in a matrix where the rows correspond to one chroma vector sequence (chromagram) and columns correspond to the other. The path is a sequence of adjacent cells, where each cell indicates a correspondence between the two sequences, and DTW finds the path with the smallest sum of distances.

For DTW, each matrix cell (*i,j*) represents the sum of distances along the best path from (0,0) to (*i,j*). We use the calculation pattern shown in Figure 2 for each cell. The best path up to location (*i,j*) in the matrix (labeled "D" in the figure) depends only on the adjacent cells (A, B, and C) and the distance between the chroma vectors corresponding to row *i* and column *j*. The DTW algorithm requires a single pass through the matrix to compute the cost of the best path. Then, a backtracking step is used to identify the actual path. Other formulations of DTW are possible (Hu & Dannenberg, 2002), but we have not explored them in this application.



$$D = M_{i,j} = \min(A,B,C) + \text{dist}(i,j)$$

Figure 2. The calculation pattern for cell (i,j) in the matrix.

Using dynamic time warping, we can use the similarity matrix in Figure 1 to identify the path as shown by the white line in Figure 3.

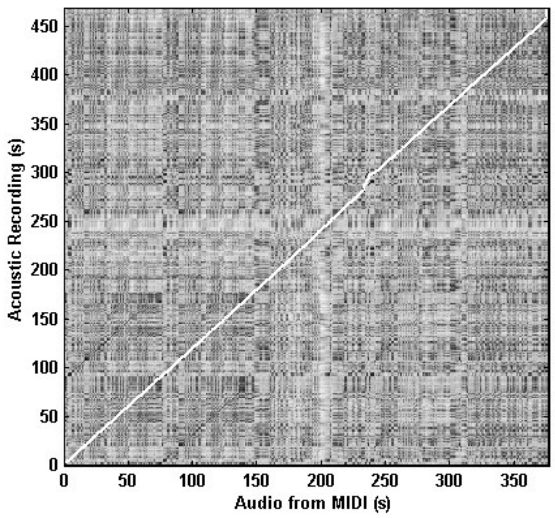


Figure 3. The optimal alignment path is shown in white over the similarity matrix of Figure 1.

3 From MIDI to Chroma

One optimization of this work is to avoid synthesizing MIDI to obtain audio. Why compute and process so many audio samples when the ultimate goal is to obtain the very reduced chromagram representation? It is possible to associate a chroma vector with each pitch class. Then, where there is polyphony in the MIDI data, the chroma vectors can simply be added and normalized.

We have found that the chromagram is relatively insensitive to these details. For example, we can substitute a piano sound for all instruments in an MIDI file and still obtain good matching to audio. Figure 4 was computed using the same recording and same MIDI data as Figure 3, except all MIDI instruments were changed to use a generic piano sound. As the figure illustrates, this change had little impact on the results.

Taking this one step further, rather than synthesizing piano tones, we can simply map each pitch to a chroma vector. Polyphony is handled by summing vectors and then normalizing. Of course, this ignores many details that would be present in synthesized sound, including envelopes, instrument (MIDI program change), and vibrato. Nevertheless, we have obtained good results with this approach. Details on this work are forthcoming (Hu, Dannenberg & Tzanetakis, 2003).

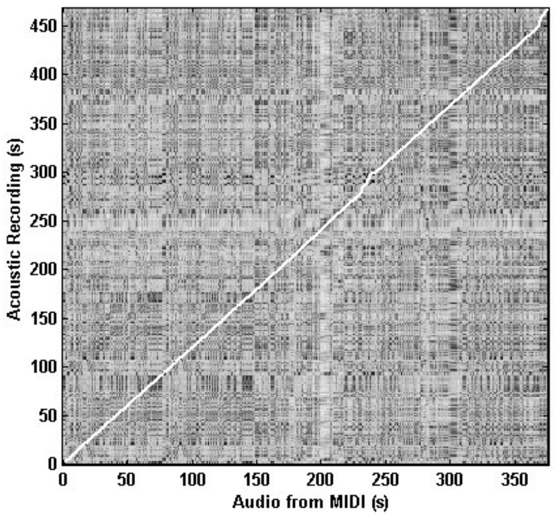


Figure 4. The optimal alignment path computed using a piano synthesizer to create audio from MIDI rather than using the original symphonic instrumentation.

4 Why not HMMs?

Readers familiar with hidden Markov models (HMMs) may wonder why we did not chose this formalism. For example, Christopher Raphael has used HMMs for computer accompaniment (Raphael, 1999) and demonstrated a polyphonic score-following system at ISMIR 2002 (Raphael, 2003). Others have also used HMMs for score following. (Cano, Loscos, & Bonada, 1999; Orio & Dechelle, 2001) We should point out that dynamic time warping is a particular form of HMM, where cells in the matrix correspond to states and the chroma vector distance serves as the output probability for a given state. (Durbin, Eddy, Krogh, & Mitchison, 1998)

The advantages of an HMM *might* be the possibility of more general state transitions and probabilities and the ability to train output probability distributions. On the other hand, an HMM would typically output discrete symbols rather than continuous chroma vectors. This requires some sort of vector quantization, and there is a tradeoff between the number of parameters to learn and the coarseness of quantization. Successful HMMs typically “tie” states to simplify training, but this introduces still more design decisions.

We plan to investigate HMMs and believe that a careful design could lead to improved performance over our DTW approach. However, we have achieved good results with a simple model and *no* training, and we believe the simplicity makes this approach very attractive for a variety of applications.

5 Evaluation and Results

Figures 3 and 4 illustrate successful matching between audio and MIDI data. The roughly diagonal lines show the correspondence. Of course, we could simply draw a diagonal line and the result would be approximately correct. How can we evaluate whether our matching is really working?

One method of evaluation is to try matching against a random MIDI file. If the best alignment is not a smooth diagonal, it indicates that the dynamic time warping path is at least being guided by the data. Figure 5 shows the same audio as in Figure 1, but matched against a MIDI file containing the second movement of Beethoven’s Fifth Symphony.

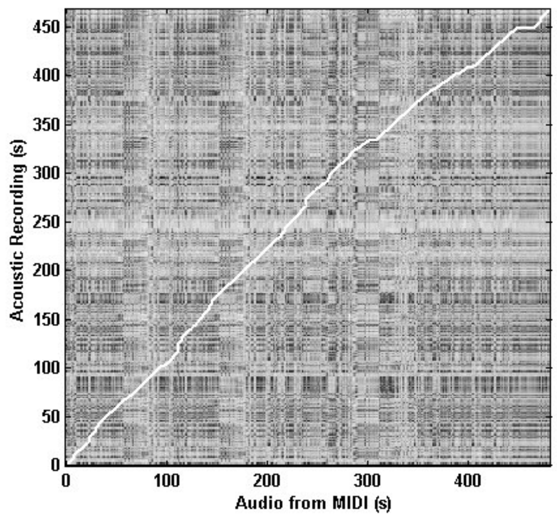


Figure 5. Matching the first movement (acoustic audio) to second movement (audio from MIDI).

As shown in the figure, the path is fairly erratic because no “true” match was found between the two audio signals. Since the DTW algorithm searches for the optimal path, the path wanders quite a bit, avoiding highly dissimilar pairs of chroma vectors, and seeking out locally similar sequences. Of course, the overall shape is still roughly diagonal, and without knowing the music, one could imagine that even this erratic path is a plausible match between two performances.

To test whether bends in the path are random or meaningful, we can introduce artificial tempo changes into MIDI data and compute the new path. Figure 6 shows a match using the same data (Beethoven’s Fifth Symphony, first movement), but where part of the MIDI file has a slower tempo. The artificially changed tempo is clearly visible.

A final method of evaluation is to compare the average distance along the path for matching vs. non-matching MIDI data. When matching and alignment are possible, we would expect to see a low average distance along the alignment path. On the other hand, if we use MIDI data that is unrelated to the audio, then even the best path should exhibit a large average distance. Table 1 shows data for matched and mismatched audio/MIDI pairs. The average distance is much higher in the mismatched case (3.63 > 2.10). These averages are computed over path lengths of 1882 and 2148, respectively. This is further evidence that the chroma vector alignment is real. Moreover, the average distance might be useful to predict when matching is successful.

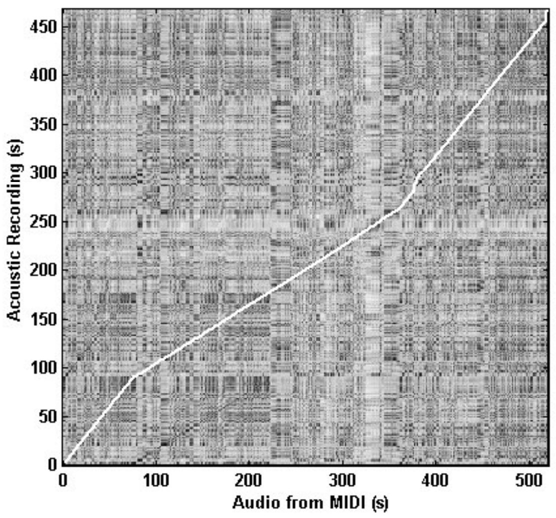


Figure 6. Matching against MIDI file with artificially varied tempo.

Acoustic	MIDI	Avg. Dist.	Ratio
Beeth. 1st Mvt.	same	2.10	0.175
Beeth. 1st Mvt.	same (piano)	2.25	0.187
Beeth. 1st Mvt.	2nd Mvt.	3.63	0.302
Beeth. 1st Mvt.	same, with tempo change	2.00	0.167
“Let It Be”	same	2.41	0.673

Table 1. Average distance along path. “Ratio” is the ratio of average distance along path to the average distance value in the entire similarity matrix.

The last column of the table shows the ratio between the average distance along the path and the average distance value in the matrix. The average distance along the path is lower than the average matrix value as indicated by ratio values less than one.

We have also evaluated alignment accuracy by comparing automatic alignment data to manual alignment. Because manual alignment is very time-consuming, we chose 5 points in each of three pieces and computed the average error and standard deviation, as shown in Table 2. The error seems to be entirely due to quantization effects of the analysis frames, so we might get better results with shorter frames.

Test Name	Avg. Error	Std. Deviation
Beeth.	0.052s	0.111s
Beeth.-vary tempo	0.034	0.056
“Let It Be”	0.076	0.112

Table 2. Alignment error averaged over 5 points in each test.

As indicated in Tables 1 and 2, we have applied our matching technique to a popular song, the Beatles' “Let It Be.” This is an interesting test case because the song features vocals prominently, whereas the corresponding MIDI must use MIDI notes to approximate the vocals. We expected to have severe problems with vocal music, but our matching technique handles the data quite well. The alignment path is shown in Figure 7.

6 Applications and Future Work

One obvious application of this work is polyphonic score following. Using techniques first introduced by Dannneberg (Dannenberg, 1985), the DTW algorithm can be implemented as an on-line algorithm, enabling real-time score following. This would enable a computer to follow an orchestra or smaller ensemble, synchronizing sound, digital audio effects processing, animation, or other time-based processes.

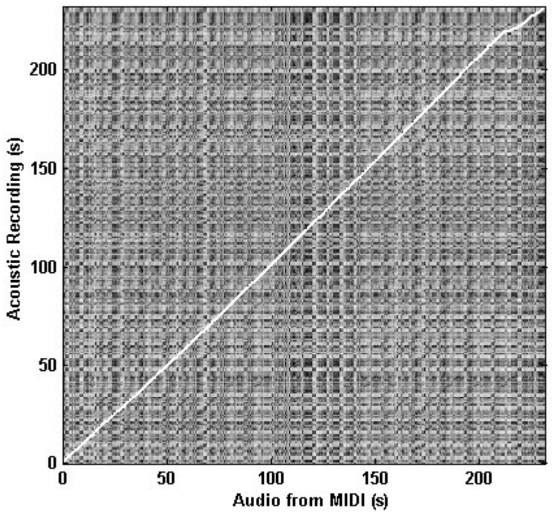
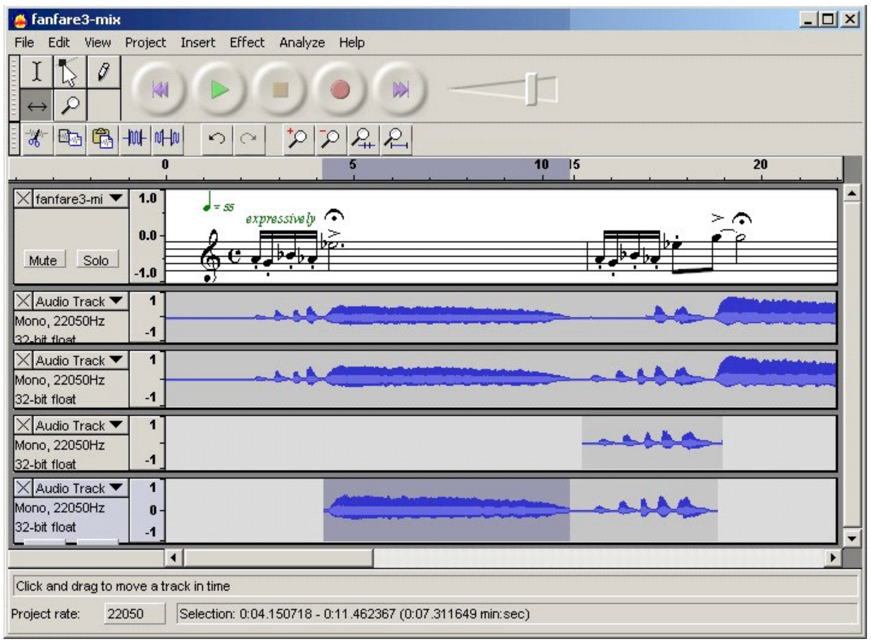


Figure 7. “Let It Be” with vocals matched with MIDI data.

Another application is to bridge between symbolic and audio representations in music editors. Imagine the following scenario: a composer creates a score and parts using a music notation editor. The parts are given to musicians who perform them in a recording studio. The parts are also transferred electronically to an “intelligent audio editor”. As music is recorded in the studio, the editor matches each “take” to MIDI data extracted from the electronic score. After everything has been recorded, often with multiple takes, the various takes are aligned with the MIDI data and displayed on multiple tracks (see Figure 8) below music notation. The composer or conductor can then easily browse through the score, auditioning various takes to select the best versions. The score can also help to find natural places to make “splices.”



Click and drag to move a track in time

Project rate: 22050 | Selection: 0:04.150718 - 0:11.462367 (0:07.311649 min:sec)

Figure 8. Mock-up of an intelligent editor. Multiple takes are automatically aligned beneath music notation.

We believe this feature would be easy to add to audio editors and would greatly simplify the management of recording projects.

Score-following has been used for the study of expressive performance. (Hoshishiba, Horiguchi, & Fujinaga, 1996; Large, 1993) Typically, researchers must either restrict their examples to keyboard performances where MIDI output is available, or manually label beats in audio. With our alignment technique, tempo variations in symphonic performances can be tracked automatically. We suspect there may be interesting differences between the expressive performance techniques used by pianists and those used by orchestras waiting to be discovered. Although one might argue that analysis is only possible after creating a MIDI representation, most standard orchestral works are already freely available in MIDI format on the Web.

The original idea for this work came from problems of music search (Birmingham et al., 2001). To search for melodies, we need transcriptions of audio, but transcription has yet to be automated well enough for this application. However, transcriptions already exist for most popular music in the form of MIDI files that can be obtained on the Web. The problem is to find MIDI files that correspond to recordings. (File names are not reliable in our experience.) By computing the average distance along the path, we can search a MIDI database for a match to audio. Once a match is found, melodic lines can be easily extracted. (Meek & Birmingham, 2001) This approach might also be used to identify various acoustic performances of works for which MIDI representations exist, for example, find all the live recordings of a song in a recording archive.

There are many possible directions for future research. We need to test this method on more music to learn what types of music are especially difficult. We know for example, that jazz is difficult because MIDI files do not typically transcribe improvised solos which strongly affect the chromagram. We have not attempted any contemporary music, but if the music can be reasonably rendered by MIDI, alignment should work well. On the other hand, music that emphasizes extended techniques that cannot be approximated by MIDI tones may not exhibit strong similarities we can use for alignment. We can also experiment with variations on the chromagram or try entirely new features. The applications discussed in this section have not been implemented, although we have begun working on an intelligent editor. (Tzanetakis, Hu, & Dannenberg, 2003) It would be interesting to make a careful comparison between the dynamic programming and hidden Markov model approaches.

7 Summary and Conclusions

Polyphonic music in audio form presents a great barrier to any number of music processing tasks. Without polyphonic transcription, we are almost always forced to treat polyphonic audio as a time-varying spectrum in which individual instruments cannot be distinguished. Gross features such as tempo, beat strength, and average spectrum can be estimated, but a more detailed analysis is frustratingly difficult.

In this work, we show how to align polyphonic audio with a symbolic (MIDI) representation. The technique uses the chromagram representation, which carries information about harmony and prominent pitches, but otherwise tends to suppress spectral details. We convert MIDI data to audio, then convert both audio representations to sequences of chroma vectors. These sequences can then be aligned using dynamic time warping. We also note that it is possible to convert directly from MIDI to chromagrams, avoiding intermediate synthesis and analysis steps.

The alignment provides a bridge from signal to symbol, almost as if we had a transcription of the polyphonic audio. While a true transcription would provide both pitch sequences and timing, we assume the correct pitch sequence is given and only deduce the timing. This is still enough information to be useful in several applications.

Polyphonic score following in real time is a promising application. Unlike most current systems that follow monophonic instruments or polyphonic MIDI keyboard performances, our technique should be able to follow an orchestra or other ensemble. Another intriguing possibility is to use our matching procedures to align audio with music notation in a music editor. This would greatly simplify many editing and browsing tasks. We also see applications in music information retrieval.

8 Acknowledgments

This work was supported by NSF Award #0085945. We would like to thank Greg Wakefield and Mark Bartsch for their chromagram code and discussions, and Christopher Rafael for his insights into the workings of HMM score following models.

References

Bartsch, M., & Wakefield, G. H. (2001). "To Catch a Chorus: Using Chroma-Based Representations For Audio Thumbnailing." *Proceedings of the Workshop on Applications of Signal Processing to Audio and Acoustics*. IEEE.

Birmingham, W. P., Dannenberg, R. B., Wakefield, G. H., Bartsch, M., Bykowski, D., Mazzoni, D., Meek, C., Mellody, M., & Rand, W. (2001). "MUSART: Music Retrieval Via Aural Queries." *International Symposium on Music Information Retrieval*. pp. 73-81.

Bloch, J., & Dannenberg, R. B. (1985). "Real-Time Accompaniment of Polyphonic Keyboard Performance." *Proceedings of the 1985 International Computer Music Conference*. International Computer Music Association, pp. 279-290.

Cano, P., Loscos, A., & Bonada, J. (1999). "Score-Performance Matching using HMMs." *Proceedings of the 1999 International Computer Music Conference*. San Francisco: International Computer Music Association, pp. 441-444.

Dannenberg, R. B. (1985). "An On-Line Algorithm for Real-Time Accompaniment." *Proceedings of the 1984 International Computer Music Conference*. San Francisco: International Computer Music Association, pp. 193-198.

Durbin, R., Eddy, S., Krogh, A., & Mitchison, G. (1998). *Biological sequence analysis*: Cambridge University Press.

Hoshishiba, T., Horiguchi, S., & Fujinaga, I. (1996). "Study of Expression and Individuality in Music Performance Using Normative Data Derived from MIDI Recordings of Piano Music." *International Conference on Music Perception and Cognition*. pp. 465-470.

Hu, N., & Dannenberg, R. B. (2002). "A Comparison of Melodic Database Retrieval Techniques Using Sung Queries." *Joint Conference on Digital Libraries*. Association for Computing Machinery.

Hu, N., Dannenberg, R. B., & Tzanetakis, G. (2003). "Polyphonic Audio Matching and Alignment for Music Retrieval." *Proceedings of the Workshop on Applications of Signal Processing to Audio and Acoustics*. IEEE. (to appear).

Large, E. W. (1993). "Dynamic programming for the analysis of serial behaviors." *Behavior Research Methods, Instruments, and Computers, 25*(2), 238-241.

Logan, B. (2000). "Mel Frequency Cepstral Coefficients for Music Modeling." *First International Symposium on Music Information Retrieval*. Plymouth, Massachusetts.

Meek, C., & Birmingham, W. P. (2001). "Thematic Extractor." *2nd Annual International Symposium on Music Information Retrieval*. Bloomington: Indiana University, pp. 119-128.

North German Radio Symphony Orchestra. (1992). *Beethoven's Fifth Symphony, First Movement*: RCA Red Seal.

Orio, N., & Dechelle, F. (2001). "Score Following Using Spectral Analysis and Hidden Markov Models." *Proceedings of the 2001 International Computer Music Conference*. San Francisco: International Computer Music Association, pp. 151-154.

Orio, N., & Schwarz, D. (2001). "Alignment of Monophonic and Polyphonic Music to a Score." *Proceedings of the 2001 International Computer Music Conference*. San Francisco: International Computer Music Association, pp. 155-158.

Raphael, C. (1999). "Automatic Segmentation of Acoustic Musical Signals Using Hidden Markov Models." *IEEE Transactions on PAMI, 21*(4), 360-370.

Raphael, C. (2003). Personal Communication.

Sankoff, D., & Kruskal, J. B. (1983). *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*. Reading, MA: Addison-Wesley.

Toivonen, T., & Izumo, M. (1999). TiMidity. http://www.onicos.com/staff/iz/timidity/index.html.

Tzanetakis, G., Ermolinskyi, A., & Cook, P. (2002). "Pitch Histograms in Audio and Symbolic Music Information Retrieval." *ISMIR 2002 Conference Proceedings.* Paris: IRCAM, pp. 31-38.

Tzanetakis, G., Hu, N., & Dannenberg, R. B. (2003). "Toward an Intelligent Editor for Jazz Music." *Digital Media Processing for Multimedia Interactive Services (Proceedings of the 4th International Workshop on Image Analysis for Multimedia Interactive Services, WIAMIS 2003).* World Scientific Press.

Viens, D. L. (2000). Beethoven's Fifth Symphony, First Movement [Standard MIDI File]. dlviens@empire.net.