Ground-Truth Transcriptions of Real Music from Force-Aligned MIDI Syntheses

Robert J. Turetsky and Daniel P.W. Ellis
LabROSA, Dept. of Electrical Engineering,
Columbia University, New York NY 10027 USA
{rob,dpwe}@ee.columbia.edu

Abstract

Many modern polyphonic music transcription algorithms are presented in a statistical pattern recognition framework. But without a large corpus of real-world music transcribed at the note level, these algorithms are unable to take advantage of supervised learning methods and also have difficulty reporting a quantitative metric of their performance, such as a Note Error Rate.

We attempt to remedy this situation by taking advantage of publicly-available MIDI transcriptions. By force-aligning synthesized audio generated from a MIDI transcription with the raw audio of the song it represents we can correlate note events within the MIDI data with the precise time in the raw audio where that note is likely to be expressed. Having these alignments will support the creation of a polyphonic transcription system based on labeled segments of produced music. But because the MIDI transcriptions we find are of variable quality, an integral step in the process is automatically evaluating the integrity of the alignment before using the transcription as part of any training set of labeled examples.

Comparing a library of 40 published songs to freely available MIDI files, we were able to align 31 (78%). We are building a collection of over 500 MIDI transcriptions matching songs in our commercial music collection, for a potential total of 35 hours of note-level transcriptions, or some 1.5 million note events.

Keywords: polyphonic transcription, MIDI alignment, dataset/corpus construction

1 Introduction

Amateur listeners have difficulty transcribing real-world polyphonic music, replete with complex layers of instrumentation, vocals, drums and special effects. Indeed it takes an expert musician with a trained ear, a deep knowledge of music theory and exposure to a large amount of music to be able to accomplish such a task. Given its uncertain nature, it is appropriate to pose transcription as a statistical pattern recognition problem (Walmsley et al., 1999; Goto, 2000).

A fundamental obstacle in both the training and evaluation of these algorithms is the lack of labeled ground-truth transcription data from produced (i.e. "real") music. The kind of information we need is the kind of description present in a MIDI file: a sequence of note pitches and the precise time at which they occur. However, exact MIDI transcriptions of specific recordings are not available[1].

One approach is to take existing MIDI files, which are often constructed as musical performances in their own right, and generate the corresponding audio by passing them to MIDI synthesizer software. For this synthesizer output, the MIDI file provides a very accurate transcription, so this data would be useful for system evaluation and/or classifier training (Klapuri et al., 2001).

Ultimately this approach suffers because of the lack of expressive capability available in MIDI files, and the corresponding paucity of audio quality in MIDI-generated sound. The 128 General MIDI instruments pale in comparison to the limitless sonic power of the modern recording studio, and MIDI is intrinsically unable to represent the richness of voice or special effects. As a result, transcription performance measured on MIDI syntheses is likely to be an over-optimistic indicator of performance on real acoustic or studio recordings.

Our proposal is an attempt at getting the best of both worlds. Vast libraries of MIDI transcriptions exist on the Internet, transcribed by professionals for Karaoke machines, amateur musicians paying deference to beloved artists, and music students honing their craft. These transcriptions serve as a map of the music, and when synthesized can create a reasonable approximation of the song, although the intent is usually to create a standalone experience, and not an accurate duplicate of the original. In our experience, such transcriptions respect the general rhythm of the original, but not precise timing or duration. However, we believe that we can achieve a more precise correlation between the two disparate audio renditions - original and MIDI

[1] Although many contemporary recordings include many MIDI-driven synthesizer parts, which could in theory be captured, this data is unlikely to be released, and will only ever cover a subset of the instruments in of the music. It is worth, however, noting the work of Goto et al. (2002) to collect and release a corpus of unencumbered recordings along with MIDI transcriptions.

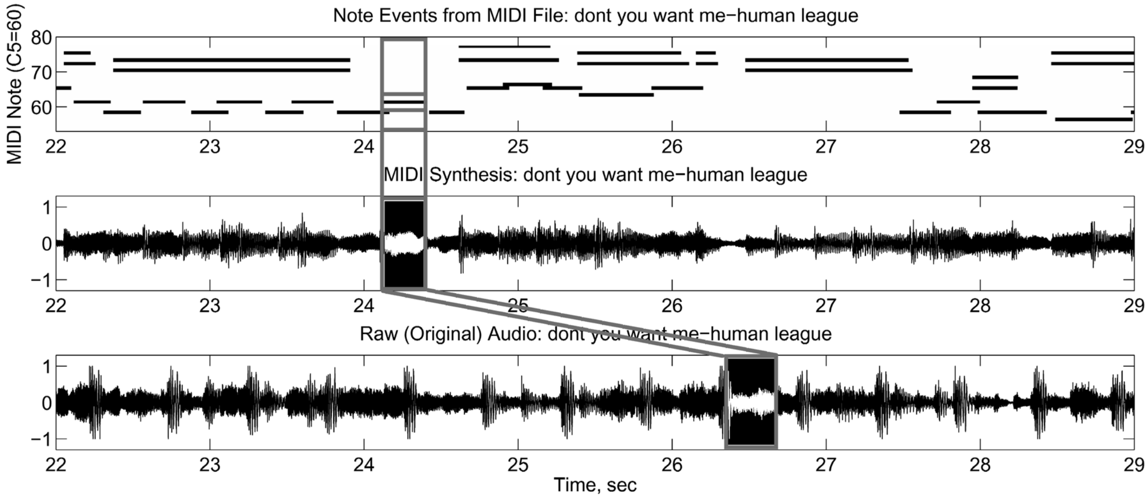

Figure 1: Goal of MIDI Alignment. The upper graph is the transcription of the first verse of “Don’t You Want Me”, by the Human League, extracted from a MIDI file placed on the web by an enthusiast. The MIDI file can be synthesized to create an approximation of the song with a precise transcription, which can analyzed alongside the original song in order to create a mapping between note events in the MIDI score and the original song, to approximate a transcription.

synthesis - through forced alignment. This goal is illustrated in figure 1.

In order to align a song with its transcription, we create a similarity matrix, where each point gives the cosine distance between short-time spectral analyses of particular frames from each version. The features of the analysis are chosen to highlight pitch and beat/note onset times. Next, we use dynamic programming to find the lowest-cost path between the starts and ends of the sequences through the similarity matrix. Finally, this path is used as a mapping to warp the timing of the MIDI transcription to match that of the actual CD recording.

Because the downloaded MIDI files vary greatly in quality, it is imperative that we judge the fitness of any alignments that we plan to use as ground truth transcriptions. Given a good evaluation metric, we could automatically search through thousands of available MIDI files to find the useful ones, and hence generate enough alignments to create a substantial corpus of real, well-known pop music recordings with almost complete note-level transcriptions aligned to an accuracy of milliseconds. This could be used for many purposes, such as training a robust non-parametric classifier to perform note transcriptions from real audio.

The remainder of this paper is organizes as follows: In section 2 we present the methodology used to create MIDI/raw forced alignments. Following that is a discussion on evaluation of the features used and an estimate of the fitness of the alignments. In the final section, we detail the next steps of this work.

2 Methodology

In this section we describe the technique used for aligning songs with their MIDI syntheses. A system overview is shown in figure 2. First, the MIDI file is synthesized to create an audio file (syn). Second, the short-time spectral features (e.g. spectrogram) are computed for both the syn and the original music audio file (raw). A similarity matrix is then created, providing the cosine distance between each frame i in raw and frame j in syn. We then employ dynamic programming to search for the “best” path through the similarity matrix. To improve our search, we designed a two-stage alignment process, which effectively searches for high-order structure and then for fine-grain details. The alignment is then interpolated, and each note event in the MIDI file is warped to obtain corresponding onset/offset pairs in the raw audio. Each stage of the process is described below.

2.1 Audio File Preparation

To begin we start with two files, the raw audio of the song we wish to align, and a MIDI file corresponding to the song. Raw audio files are taken from compact disc, and MIDI files have been downloaded from of user pages found through the MIDI search engine `http://www.musicrobot.com`. The MIDI file is synthesized using WAVmaker III by Polyhedric Software. Rather than simply recording the output of a hardware MIDI synthesizer on a sound card, WAVmaker takes great care to ensure that there is a strict correspondence between MIDI ticks (the base unit of MIDI timekeeping) and samples in the resynthesis. We can move from ticks to samples using the following conversion:

$$N_{cur} = N_{base} + F_s \cdot 10^6 \cdot (T_{cur} - T_{base})(\Delta/PPQ) \quad (1)$$

where N_{cur} and N_{base} are the sample indices giving current position and the time of the most recent tempo change, and T_{cur} and T_{base} are the corresponding times in MIDI ticks. Δ is the current tempo in quarters per microsecond, and PPQ is the division parameter from the MIDI file, i.e. the number of ticks per quarter note. F_s is the sampling rate.

Both the raw and syn are downsampled to 22.05kHz and normalized, with both stereo channels scaled and combined into a monophonic signal.

2.2 Feature Calculation

The next step is to compute short-time features for each window of syn and raw. Since the pitch transcription is the strongest cue

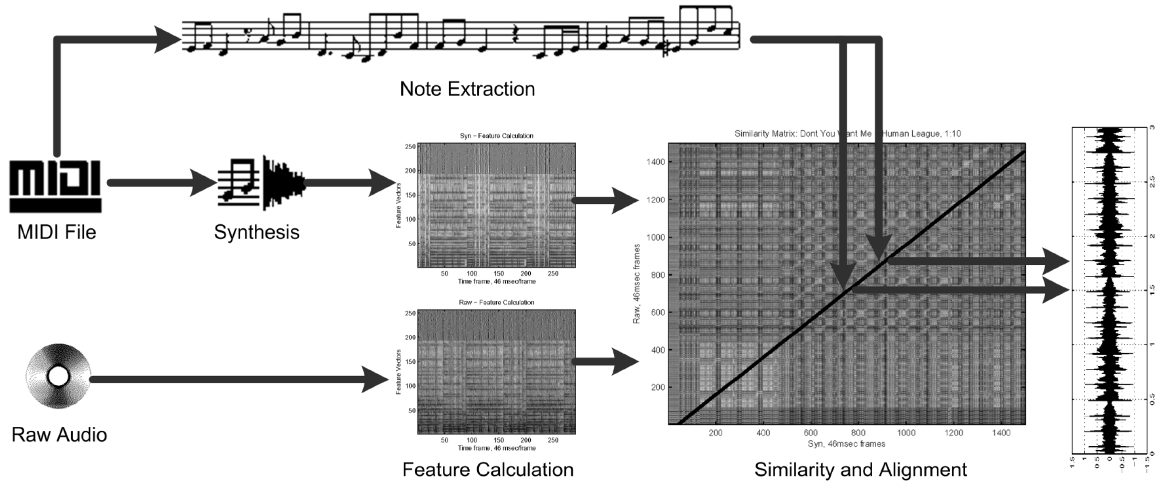

Figure 2: The Alignment Process. Features computed on original audio and synthesized MIDI are compared in the similarity matrix (SM). The best path through SM is a warping from note events in the MIDI file to their expected occurrence in the original.

for listeners to identify the approximate version of the song, it is natural that we choose a representation that highlights pitch. This rules out the popular Mel-Frequency Cepstral Coefficients (MFCCs), which are specifically constructed to eliminate fundamental periodicity information, and preserve only broad spectral structure such as formants. Although authors of MIDI replicas may seek to mimic the timbral character of different instruments, such approximations are suggestive at best, and particularly weak for certain lines such as voice etc. We use features based on a narrowband spectrogram, which encapsulates information about the harmonic peaks within a frame. We apply 2048 point (93 ms) hann-windowed FFTs to each frame, with each frame staggered by 1024 samples (46 ms), and then discard all bins above 2.8 kHz, since they relate mainly to timbre and contribute little to pitch. As a safeguard against frames consisting of digital zero, which have an infinite vector similarity to nonzero frames, 1% white noise is added across the spectrum.

In our experiments we have augmented or altered the windowed FFT in one or more of the following ways (values that we tried are shown in braces):

1. *spec_power* {'dB', 0.3, 0.5, 0.7, 1, 2} - We had initially used a logarithmic (dB) intensity axis for our spectrogram, but had problems when small spectral magnitudes become large negative values when converted to dB. In order to approximate the behavior of the dB scale without this problem, we raised the FFT magnitudes to a power of spec_power. In our experiments we were able to perform the best alignments by taking the square root (0.5) or raising the spectrum to the 0.7 power.

2. *diff_points* {0, [1 16], [1 64], [224 256]} - First order differences in time of some channels are appended as additional features. The goal here is to highlight onset times for pitched notes using the low bins and for percussion using the high bins. In our experiments, we found that adding the high-bin time differences had negligible effect because of the strong presence of the snares and hi-hats across the entire spectrum. Adding low-order bins can improve alignment, but may also cause detrimental misalignment in cases where the MIDI file was transcribed off-key.

3. *freq_diff* {0, 1} - First-order difference in frequency attempts to normalize away the smooth spectral deviations between the two signals (due to differences in instrument timbres) and focus instead on the local harmonic structure that characterizes the pitch.

4. *noise_supp* {0, 1} - Noise suppression. Klapuri has proposed a method for simultaneously removing additive and convolutive noise through RASTA-style spectral processing (Klapuri et al., 2001). This method was not as successful as we had hoped.

2.3 The Similarity Matrix

Alignment is carried out on a similarity matrix (SM), based on the self-similarity matrix used commonly in music analysis (Foote, 1999), but comparing all pairs of frames between two songs, instead of comparing one song against itself. We calculate the distance between each time step i $(1 \ldots N)$ in the raw feature vector series $spec_{raw}$ with each point j $(1 \ldots M)$ in the syn series $spec_{syn}$. Each value in the $N \times M$ matrix is computed as follows:

$$SM(i,j) = \frac{spec_{raw}(i)^T spec_{syn}(j)}{|spec_{raw}(i)||spec_{syn}(j)|}, \quad \text{for } 0 \leq i < N, \quad 0 \leq j < M \qquad (2)$$

This metric is proportional to the similarity of the frames, e.g. similar frames have a value close to 1 and dissimilar frames have a value that can approach -1. Several similarity matrices for different songs are shown in figure 4, in which we can see the structures that are also present in self-similarity matrices, such as the checkerboard pattern corresponding to segment boundaries, and off-diagonal lines signifying repeated phrases Bartsch and Wakefield (2001); Dannenberg and Hu (2002). The presence of the strong diagonal line indicates a good alignment between the raw and syn files, and the next stage attempts to find this alignment.

2.4 Alignment

The most crucial stage in the process of mapping from MIDI note event to raw audio sample is searching for the "best path" through the similarity matrix. This is accomplished with dynamic programming or DP (Gold and Morgan, 1999), tracing a path from the origin to $(N-1, M-1)$. DP is used in many applications, ranging from the Dynamic Time Warp used in speech recognition, through to aligning closely-related sequences of DNA or amino acids in bioinformatics.

DP consists of two stages, forward and traceback. In the forward step, we calculate the lowest-cost path to all of the point's neighbors plus the cost to get from the neighbor to the point - in this case, $S_{max} - SM(i,j)$, where S_{max} is the largest value in the similarity matrix. This makes the cost for the most similar frame pairs be zero, and all other frame pairs have larger, positive costs. The object of this stage is to compute the cost of the best path to point $(N-1, M-1)$ recursively by searching across all allowable predecessors to each point and accumulating cost from there. In the traceback stage, we find the actual path itself by recursively looking up the point providing the best antecedent for each point on the path.

We have experimented with of two complementary flavors of DP, which we have dubbed "unconstrained" and "greedy". Figure 3 demonstrates this on a small scale, using the alignment of a three note sequence. Panels (a) and (d) show the freedom of movement allowed in each configuration; in our DP, all allowed steps have equal weight. Panels (b) and (e) show the DP when the transcription of the sequence is correct, and the third column shows the behavior of the DP when the MIDI transcription has the second and third notes swapped.

Unconstrained DP (top row) allows the best path to include horizontal and vertical segments, thus allowing it to skip entire regions of original or resynthesis. When searching for the lowest-cost path on a SM that penalizes dissimilar frames, unconstrained DP is capable of bypassing extraneous segments such as repeated verses or mis-transcribed notes. However, in terms of fine-grained structure, its performance is weak, since it may include short horizontal or vertical regions within individual notes when certain frame pairs align particularly well. Greedy DP (bottom row), on the other hand, enforces forward motion by at least one frame on both axes at every step. This behavior results in smooth structure when the notes are correctly transcribed; however, in the presence of errors, the greedy method will still find a convincing-looking roughly-diagonal path even when no good one exists (panel (f) of figure 3).

2.4.1 Two-stage Alignment

We developed a two stage alignment process exploiting the complementary DP flavors. In the first stage, we use unconstrained DP to discover the regions of the song which appear likely to align well, and to skip regions that contain no good diagonal paths. We then apply a 31-point (1.4 s) median filter across the slope of the best path found by unconstrained DP. The median filter searches for contiguous regions in which the majority of steps are diagonal, as these segments are most likely able to be aligned rather than skipping over misalignments. Straight segments of any significant length become boundaries, and greedy DP is applied in between them to re-align these 'good' segments more accurately.

2.5 Interpolation

Once alignment is complete, we can treat the best path as a warping function between syn samples and raw samples. In order to achieve sub-frame resolution, the best path is linearly interpolated and smoothed across multiple alignment frame pairs. To demonstrate the warping by the best path, we created two sets of stereo resyntheses with raw in one ear against syn in the other. The first simply plays each frame of raw/syn along the best path, without taking too much care to smooth transitions between discontiguous frames. The second plays the unmodified raw in one channel, and MIDI synthesis based on the time-warped MIDI notes in the other. These examples are available at the MIDIAlign project webpage, `http://www.ee.columbia.edu/~rob/midialign/`.

3 Featureset and Alignment Evaluation

The primary objective of these alignments is to create a large set of transcribed real-world polyphonic music. Given the murky origins of many of the MIDI files we encounter, there are alignments that, simply put, do not work; figure 4 gives four example alignments, illustrating some of the problems we encountered. While the alignment algorithm is tolerant of missing sections and a certain amount of ornamentation or interpretation, other errors will defeat it completely. We have encountered transcriptions that are transposed, with segments transcribed out of order, of a different remix/version of the song, or which are simply abysmal. Our web crawling has uncovered MIDI transcriptions for thousands of songs, with a handful of different versions for each, and hence the importance of an automatic metric to discriminate the successful alignments cannot be overstated.

3.1 Evaluation of featuresets

In order to evaluate different candidate metrics, we first computed the alignment for 7 different featuresets on each of the 40 songs in our corpus. The songs were chosen to include some variety in terms of different types of popular music, and not because of the quality of their transcriptions. We then manually evaluated 560 alignments, two for each feature for each song corresponding to the best path found by the first and second stages. Manual evaluation was our only choice, initially, since we had no ground truth available of time-point correspondences between original recordings and their MIDI renditions. Listening to the aligned resyntheses was considered less arduous than annotating the set of music to create this ground truth. We aurally evaluated the rendered comparisons on the following scale:

Note that this score encapsulates both the performance of the algorithm and the quality of the transcription. The Count column is the number of songs that reached this level of evaluation as the highest score across all of the features. Thus, for 31 of the 40 MIDI/raw pairs, our automatic alignment gave us at least one perfect or near-perfect alignment. These could be extracted and used as a manually-verified ground truth for further alignment experiments, although we have not used them this way as yet. They are of course by definition the examples on which automatic analysis is most likely to succeed.

We can use these annotations to determine which featureset is most effective. Table 2 charts each featureset that we've used vs. the number of occurrences of each subjective quality label for alignments based on that featureset. From the table, it can

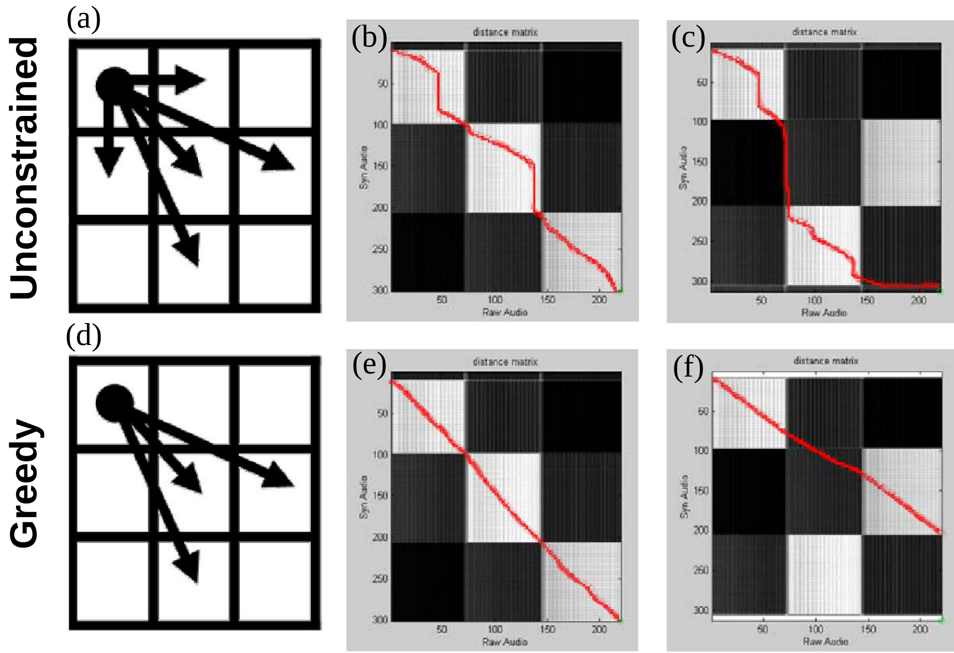

Figure 3: Flavors of DP. The top row is for unconstrained DP, the bottom row is for greedy. The first column shows the allowable directions of movement. The second column shows the alignment of raw/MIDI notes C5-C♯5-D5, for unconstrained and greedy respectively. The third column shows the alignment of raw as before, but with the last two notes of MIDI reversed.

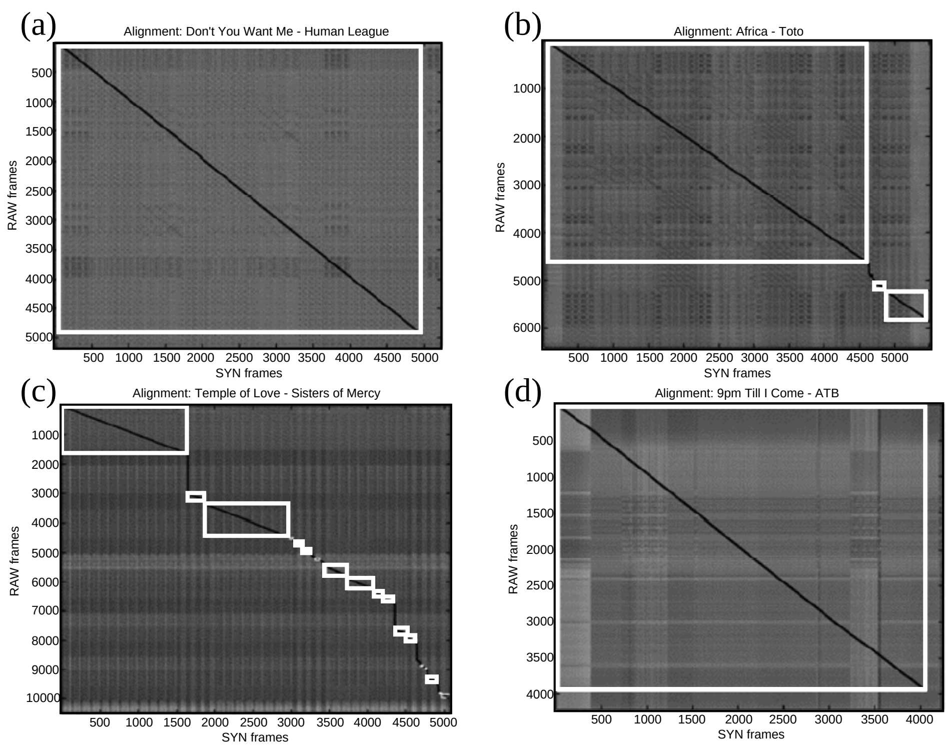

Figure 4: Example alignments for four SMs: (a) Don't You Want Me (Human League): straightforward alignment; (b) Africa (Toto): bridge missing in transcription; (c) Temple of Love (Sisters of Mercy): Transcription out of order but we recover; (d) 9pm 'Till I Come (ATB): poor alignment mistakenly appears successful. White rectangles indicate the diagonal regions found by the median filter and passed to the second-stage alignment.

Score	Interpretation	Count / 40
1	Could not align anything. Not for use in training.	4
2	Some alignment but with timing problems, transposition errors, or a large number of misaligned sections.	2
3	Average alignment. Most sections line up but with some glitching. Not extremely detrimental to training, but unlikely to help either.	3
4	Very good alignment. Near perfect, though with some errors. Adequate for training.	4
5	Perfect alignment.	27

Table 1: Subject alignment assessment. Different versions of MIDI syntheses aligned to original audio were auditioned in stereo against the original recording, and given a score from 1 to 5, indicating the combined quality of the original MIDI file and the alignment process.

	Score				
Featureset	1	2	3	4	5
spec_power = 0.5	6	2	7	6	15
spec_power = 0.7	6	3	4	10	14
spec_power = 1.0	6	4	13	5	9
spec_power = 0.7 diff_points [1 16]	5	6	4	11	13
spec_power = 0.7 diff_points [1 64]	6	7	0	4	21
spec_power = 0.7 freq_diff = 1	7	5	6	11	4
spec_power = 0.7 noise_supp = 1	28	1	0	1	0

Table 2: Feature Evaluation. Above is a representative list of featuresets which were applied to alignment on our corpus of music. Columns 1 through 5 indicate the number of songs evaluated at that score for that featureset. Alignments rated 4 or 5 are considered suitable for training.

be seen that spec_power = .5 or .7 and its variants lead to the most frequent occurrence of good alignments, with the highest incidence of alignments rated at "5" for spec_power = .7 and diff_points = [1 64].

3.2 Automatic Alignment Evaluation

Given the high costs on the attention of the subject to obtain the subjective alignment quality evaluations,our goal is to define a metric able automatically to evaluate the quality of alignments. Our subjective results bootstrap this by allowing us to compare those results with the output of candidate automatic measures. In order to judge the fitness of an alignment as part of a training set, we have experimented with a number of metrics to evaluate their ability to discriminate between good and bad alignments. They are:

1. Average Best Path Score - The average value of SM along the best path, in the diagonal regions found by the median filter. MIDI resyntheses that closely resemble the originals will have a large number of very similar frames, and the best path should pass through many of these. However, sources of dissimilarity such as timbral mismatch will be reflected in this value, without necessarily indicating a weak alignment.

2. Average Best Path Percentile - The average similarity score of pairs included in the best path, expressed as a percentile of all pair similarity scores in the SM. By expressing this value as a percentile, it is normalized relative to the actual variation (in range and offset) of similarities within the SM.

3. Off-diagonal Ratio - Based on the notion that a good alignment chooses points with low cost among a sea of mediocrity while a bad alignment will simply choose the shortest path to the end, the off-diagonal ratio takes the average value of SM along the best path and divides it by the average value of a path offset by a few frames. It is a measure of the 'sharpness' of the optimum found by DP.

4. Square Ratio - The ratio of areas of the segments used in second stage alignment (corresponding to approximately linear warping) as a proportion of the total area of the SM. First stage alignments that deleted significant portions of either version will reduce this value, indicating a problematic alignment.

5. Line Ratio - The ratio of the approximately linear segments of the alignment versus the length of the best path, i.e. approximately $\sqrt{SquareRatio}$.

Our experiments have shown that the metric 2, the average best path percentile, demonstrates the most discrimination between good and bad alignments. When automatically evaluating a large corpus of music, it is best to use a pessimistic evaluation criterion, to reduce false positives which can corrupt a training set. Figure 5 shows example scatter plots of automatic quality metric versus subjective quality rating for this metric with two different feature bases; while no perfect threshold exists, it is at least possible to exclude most poor alignments while retaining a reasonable proportion of good ones for the training set.

4 Discussion and conclusions

Although we are only at the beginning of this work, we intend in the immediate future to use this newly-labeled data to create note detectors via supervised, model-free machine learning techniques. Given the enormous amount of MIDI files available on the web, it is within the realm of possibility to create a dataset of only the MIDI files that have near-exact alignments. We can then note-detecting classifiers based on these alignments, and compare these results with other, model-based methods.

Additionally, we are investigating better features for alignment and auto-evaluation. One such feature involves explicitly searching for overtones expected to be present in the raw given an alignment. Another method for auto-evaluation is to use the transcription classifiers that we will be creating to re-transcribe the training examples, pruning those whose MIDI and automatic transcriptions differs significantly.

Since MIDI transcriptions only identify the attack and sustain portion of a note, leaving the decay portion that follows the 'note off' event to be determined by the particular voice and other parameters, we will need to be careful when training note detectors not to assume that frames directly following a note off

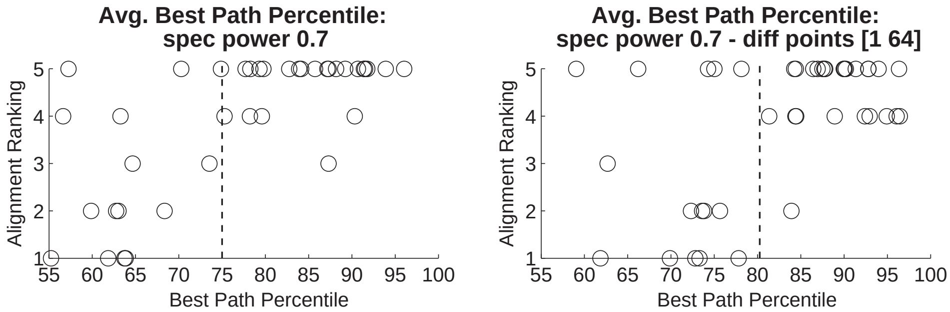

Figure 5: Comparison between subjective and automatic alignment quality indices for the average best path percentile, our most promising automatic predictor of alignment quality. Shown are equal risk cutoff points for two good-performing featuresets, with good alignments lying to the right of the threshold.

do not contain any evidence of that note. Of course, our classifiers will have to learn to handle reverberation and other effects present in real recordings.

There is also a question of balance and bias in the kind of training data we will obtain. We have noticed a distinct bias towards mid-1980s pop music in the available MIDI files: this is the time when music/computer equipment needed to generate such renditions first became accessible to a mass hobbyist audience, and we suspect many of the files we are using date from that initial flowering of enthusiasm. That said, the many thousands of MIDI files available online do at least provide some level of coverage for a pretty broad range of mainstream pop music for the past several decades.

Given that the kind of dynamic time warping we are performing is best known as the technique that was roundly defeated by hidden Markov models (HMMs) for speech recognition, it is worth considering whether HMMs might be of some use here. In fact, recognition with HMMs is very often performed using "Viterbi decoding", which is just another instance of dynamic programming. The main strength of HMMs lies in their ability to integrate (through training) multiple example instances into a single, probabilistic model of a general class such as a word or a phoneme. This is precisely what is needed in speech recognition, where you want a model for /ah/ that generalizes the many variations in your training data, but not at all the case here, where we have only a single instance of original and MIDI resynthesis that we are trying to link. Although our cosine distance could be improved with a probabilistic distance metric that weighted feature dimensions and covariations according to how significant they appear in past examples, on the whole this application seems perfectly suited to the DTW approach.

Overall, we are confident that this approach can be used to generate MIDI transcriptions of real, commercial pop music recordings, mostly accurate to within tens of milliseconds, and that enough examples are available to produce as much training data as we are likely to be able to take advantage of, at least for the time being. We anticipate other interesting applications of this corpus, such as convenient symbolic indexing into transcribed files, retrieval of isolated instrument tones from real recordings, etc. We welcome any other suggestions for further uses of this data.

Acknowledgments

Our thanks go to the anonymous reviewers for their comments which were very helpful.

References

Bartsch, M. A. and Wakefield, G. H. (2001). To catch a chorus: Using chroma-based representations for audio thumbnailing. In *Proc. IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, Mohonk, New York.

Dannenberg, R. and Hu, N. (2002). Pattern discovery techniques for music audio. In Fingerhut, M., editor, *Proc. Third International Conference on Music Information Retrieval ISMIR-02*, pages 63–70, Paris. IRCAM.

Foote, J. (1999). Methods for the automatic analysis of music and audio. Technical report, FX-PAL.

Gold, B. and Morgan, N. (1999). *Speech and Audio Signal Processing: Processing and Perception of Speech and Music*. John Wiley & Sons, Inc., New York.

Goto, M. (2000). A robust predominant-f0 estimation method for real-time detection of melody and bass lines in cd recordings. In *IEEE International Conference on Acoustics, Speech and Signal Processing ICASSP-2000*, pages II–757–760, Istanbul.

Goto, M., Hashiguchi, H., Nishimura, T., and Oka, R. (2002). Rwc music database: Popular, classical, and jazz music databases. In Fingerhut, M., editor, *Proc. Third International Conference on Music Information Retrieval ISMIR-02*, Paris. IRCAM.

Klapuri, A., Virtanen, T., Eronen, A., and Seppänen, J. (2001). Automatic transcription of musical recordings. In *Proceedings of the CRAC-2001 workshop*.

Walmsley, P. J., Godsill, S. J., and Rayner, P. J. W. (1999). Bayesian graphical models for polyphonic pitch tracking. In *Proc. Diderot Forum*.