Music Melody Matching Machine

Thomas Gersic
Northwestern University
http://www.gersic.com

ABSTRACT

I describe a simple computer-based melody matching system, based on comparing short monophonic audio recordings of pieces of tonal music. The process starts with the user singing a piece that they would like to identify into a microphone. The system then attempts to transcribe the recording from its audio representation into a string of quantized MIDI notes. For simplicity, the matching algorithm has been limited to just pitch data, but a more complete system would employ rhythmic and dynamic data as well to more accurately match pieces of music.

1. INTRODUCTION

The sale of music on the Internet has blossomed from relative obscurity to a multi-million dollar industry in the two short years since Apple launched its iTunes service. A number of other services, most notably Napster, have attempted to compete with Apple, but few have had success. One potential reason for this is that none of the competing services offer anything substantially different from what Apple has to offer, so current iTunes users have little reason to switch services. Prices are largely fixed by the record industry, so lowering prices is not an option, but one way to compete is with a significant improvement to the search technology. One issue discussed by Birmingham et al. [1] is that current music search engines use meta-information such as song title or performer, not on the content of the music, which is the information with which consumers are frequently more familiar. The system described in this paper attempts to match musical queries, sung into the system, with stored pieces of music by comparing musical content.

The system described here is an approach to comparing and grouping audio files by melodic content, based on a simple pitch tracker and an edit-distance based string matcher. I test various edit-distance settings for match reward and skip cost, as well as a comparison between global and local alignment methods. I also attempt a system of diatonic key finding based on the Krumhansl-Kessler algorithm [2], supplied by MIDI Toolbox version 1.0 [3].

Several approaches to music matching systems have been tried in the past. The MusArt Music-Retrieval System, proposed by Birmingham et al. [1] in February 2002, is a system that attempts to match an automatically generated transcription of a sung or hummed query to a database of stored MIDI files. The system uses a "stochastic representation of music", by employing the use of hidden Markov models for both the query and the stored themes.

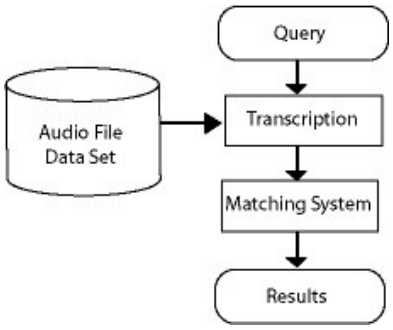
Another system, proposed by Uitdenbogerd and Zobel [4] compares polyphonic MIDI files by attempting to discern the melody line, and then comparing the files based on similarity between these melodies. They attempt a number of different methods for determining the melody line, but find that taking the highest note starting at any time yielded the best results. Uitdenbogerd and Zobel also compare two different approaches to matching strings in a similarly designed music retrieval system: n-gram count commons and edit distance local alignment. They find the best results with local alignment, and so I have chosen to use local alignment over n-grams or hidden Markov models.

The system I describe here is a matching system based on the best methods that have been tried in past literature. Through these methods, I have developed a working system whose accuracy is very promising.

2. SYSTEM DESCRIPTION

The melody matching system I describe here is in three main parts: the data set, the transcription system, and the matching system. The data set consists of stored monophonic recordings of main themes for popular songs. This data set is queried by a user, who sings or hums a melody into a microphone. After the user has recorded their query, it is transcribed into a series of frequencies by the Praat-PD algorithm [5], and these frequencies are then quantized into a series of MIDI pitches. A matching algorithm is performed on the pitches deemed to be most significant, and a number of different measures of significance are attempted and compared.

Figure 1. Music Matching System Diagram



2.1 Pitch Tracking and Quantization

Pitch tracking was accomplished with the praat-pd algorithm, coded by Mark Bartsch, modified by Bryan Pardo, and based on praat by Paul Boersma [5]. The algorithm works through windowing a signal in the time domain in order to determine the period and harmonicity (harmonics to noise ratio) of a sound. Boersma shows this method to have an accuracy and reliability greater than that of frequency-domain methods, and to return good results for sounds up to 80% of the Nyquist frequency.

The praat-pd algorithm allows for a number of input settings that determine how the algorithm functions, which affects the output results. For this system, the distance between window centers was set to 0.01 seconds, which is significantly faster than an average human is likely to be able to sing a sequence of notes. This is necessary because a longer time between windows would potentially miss notes that occurred at the edges of the window. The minimum and maximum allowable frequencies were set to 65 Hz and 1000 Hz respectively because most of the energy of the human voice is concentrated below 1000 Hz, especially the energy of vowels, which contain most of the pitch information that we're extracting. Voice threshold and silence threshold, which set the autocorrelation strength that the algorithm uses to determine harmonic sound and silence were set to 0.5 and 0.02 respectively. Defaults were used for the rest of the settings.

Once the praat-pd algorithm determined a series of frequencies by analyzing the raw PCM data, it was necessary to limit the total amount of data by quantizing it to musical pitches and time segments. For convenience, MIDI pitch numbers, with middle C equaling the number 60, were used to represent the musical pitch of each frequency. A simple formula for converting from frequency to MIDI pitch was used (the reference point of A440 = 69 was used because that pitch is commonly used as the basis for equal temperament):

$$\text{round}\left(12 * \left(\frac{\log\left(\frac{\text{frequency}}{440}\right)}{\log(2)}\right) + 69\right)$$

Once pitch was determined, an array was created that contained the distance of each note, in half steps, from the first note in the sequence. This relational method of representation allowed for the comparison of strings between keys without the key being a factor in the comparison.

In addition to quantizing to musical pitch, properly quantizing the sequence of frequencies in time proved to be very important for getting good results from the matching system, so a number of different methods were attempted. Most significantly, the discrete derivatives of the frequency path and the autocorrelation path were calculated in order to determine starting points for new notes. Based on graphical representations of praat-pd transcriptions of the human voice, the assessment was made that intentional changes in pitch are frequently associated with short-term, large jumps in both frequency and harmonicity. This effect is the result of transients in the recording caused by vocal consonants. Because of this, the effect is more pronounced in recordings where the user has sung words instead of humming.

Once the derivatives were calculated, I determined the average difference between successive values in order to compare each value to it. Doing this allowed me to determine where the significant peaks were in the derivative array without having to make an assumption that peaks would be over a certain static level. Peaks were then determined to be any non-zero frequency value with a frequency or harmonicity value over the average frequency or harmonicity values. Once starting points were determined for each note, the average of the frequencies between that starting point and the next starting point was used to determine the frequency of the note. In this system, for simplicity, time (rhythmic) information was not used, but a future system should make use of this information in order to better match melodic themes.

In addition to this, a number of different methods were tested to quantize the sequence of notes further. Firstly, the assumption was made that pitch changes occurring very quickly are most likely to be the result of a singer "scooping" up to, or down to an intended pitch. Because of this, these quickly changing pitches were determined to be unimportant, and were removed from the transcription array, with only the last pitch remaining from any group of quickly changing notes.

Secondly, two different experimental methods were introduced to the system. An FIR lowpass filter was introduced between the linearly encoded PCM vector, and the praat-pd algorithm. The reasoning behind trying this was with the hope that the algorithm would find harmonic values better if high-frequency values were gently attenuated. This method, however, had the effect of removing too many transients, and significantly reduced the effectiveness of the derivative calculation method discussed earlier. Because of this, no further testing was done. The second experimental method was the introduction of a diatonic key finding algorithm, based on the Krumhansl-Kessler algorithm [2], supplied by MIDI Toolbox version 1.0 [3]. A comparison between using the key finding algorithm and not using it is given in the results section.

2.2 Edit-distance Alignment

The Edit-distance alignment algorithm is a method taken from Durbin et al [6]. This method, shown by Uitdenbogerd and Zobel [4] to be effective and efficient for melodic theme matching, compares two strings in an attempt to determine how similar they are to each other by calculating the number of substitutions necessary to change one string into the other one. The Edit-Distance algorithm generates a matrix of data that can be used to determine either a global-alignment value or a local-alignment value. Both values are tested here. With the algorithm used, higher numbers indicate better matches, and values are normalized to a range between 0.0 and 1.0 (1.0 is a perfect match).

2.3 Krumhansl-Kessler Algorithm

The Krumhansl-Kessler algorithm was used to determine the key of each transcription. This algorithm works by analyzing the relative popularity of each pitch, and comparing it to a table of weighting values for each step in the diatonic scale. For instance, a recording with a most popular pitch of C (the tonic), and second most popular pitch of G (the dominant), is likely to be in the key of C (major or minor). The key determined by this algorithm was then used to assess the likelihood that pitches in the transcription were correct or incorrect. Pitches determined to be out of the key

were then changed either to be equal to the previous pitch, or to the closest diatonic pitch, depending on how close in frequency the pitch in question was to the previous pitch.

3. EXPERIMENTAL SECTION

The melodic matching system was tested by evaluating how well a recording could be matched to another recording of the same piece, and how well it would discriminate between correct matches and incorrect matches.

3.1 Corpus Construction

Forty different recordings were used for the experimental corpus, and are available at http://www.cs.northwestern.edu/~teg196/. These recordings were of four different pieces, each recorded five times with words and five times without words. The four pieces were: *99 Red Balloons*; the *Gilligan's Island* theme song; *Allouette*; and *Jingle Bell Rock*. The first two songs were recorded by a 26 year old male, with a Labtec AM-240 unidirectional electret microphone with a frequency response from 100 to 16000 Hz. These songs were recorded with Matlab 7.0 (R14), and linearly encoded as a PCM waveform with a sampling rate of 11025 Hz, and a bit depth of 16 bit. The second two songs were recorded by a male in his mid 30's. These songs were also linearly encoded as a PCM waveform, with a sampling rate of 8000 HZ, and a bit depth of 16 bit. All files were recorded as monaural files. The files had an average length of 4.81 seconds.

3.2 Evaluation method

Two different evaluation methods were considered. Both were based off of a data set created by comparing each file with every other file in the corpus to determine edit distance scores. The first evaluation method was to compare the average score generated for "same" matches versus the average score for "different" matches. Since the scores were quantized to a range of 0 to 1, scores for different recordings of the same song should be closer to 1 than scores for those that are different songs. The second evaluation method was to determine whether a query to the system would yield a "correct" answer. A correct answer, in this case, would be the system returning a filename for a song that was the same as the query song. For instance, if recording #1 of 99 Red Balloons was the query song, a correct answer would be recording #4 of 99 Red Balloons, but an incorrect answer would be recording #2 of Jingle Bell Rock.

3.3 Results

The experimental results were very good. A number of different variables were tested in order to produce the best possible results with this corpus of data. Variables tested were the value settings for edit distance match reward and skip cost, local versus global alignment, and whether or not the key-finding algorithm was used. With a total of 16 different test runs, I determined that a match reward of 15, a skip cost of 2, using local alignment, and not attempting to find the key yielded the best results. These values gave an average score between "same" files of approximately 0.373 and an average score between "different" files of 0.220. The ratio between these scores is 1.694 : 1. These settings returned 34 out of 40 correct matches, for an accuracy of 85.00%, which is significantly better than chance (25%). Substituting global alignment for local alignment raised the ratio significantly to 4.173 : 1, but the percent of correct matches was reduced to 80.00%. The key finding algorithm raised the average scores for both "same" and "different" matches, but reduced the system's accuracy in most cases.

Table 1. Partial Experimental Results

Full Results: http://www.cs.northwestern.edu/~teg196/teg196_testResults.xls

Match Reward	Skip Cost	Local/Global	Key NoKey	Score Ratio	Percent Matches
15	2	Local	NoKey	1.694	85.0%
15	2	Local	Key	1.579	67.5%
15	1	Local	NoKey	1.663	85.0%
2	8	Local	NoKey	1.464	57.5%
15	2	Global	NoKey	4.173	80.0%

4. CONCLUSIONS AND FUTURE WORK

The results of this experiment are promising. Using a small data set, and minimal post-processing of transcription data, relatively good results were returned from the matching algorithm. The biggest issue found was correctly quantizing the frequencies into pitches that were representative of the pitches that the user intended. The average person does not sing intervals perfectly, nor do they sing with a perfectly steady pitch, so significant work must be done to weed out values which are not intended to be considered as part of the melody. The diatonic key finding algorithm was an attempt to deal with these issues, but for the most part did not yield good results. The problem may have been that with such short segments, the key returned was not the correct key for the piece, resulting in corrupted data.

Future work in this area should concern the research of better ways to quantize the transcription data into strings that are more similar for songs that are supposed to be the same, and more different for songs that are supposed to be different. The diatonic key finding algorithm yielded some promising results, but for the most part it just made matters worse. Further testing is needed to determine exactly how best to use it. Rhythmic information was also left completely out of this system, and could prove to be pivotal in correctly matching two pieces to each other. Another area of improvement would be to have a system that would know and expect common errors, and while matching, would punish those less than other errors. A piece could be matched to another piece even if the two transcriptions have a significant number of errors, presuming that those errors fall into the set of expected errors, such as minor pitch fluctuations.

5. ACKNOWLEDGMENTS

Thanks to Bryan Pardo.

6. REFERENCES

[1] Birmingham, William et al. The MusArt Music-Retrieval System. D-Lib Magazine, (February 2002).

[2] Krumhansl, C. L. Cognitive Foundations of Musical Pitch. New York: Oxford University Press, (1990).

[3] Eerola, T. & Toiviainen, P. (2004). MIDI Toolbox: MATLAB Tools for Music Research. University of Jyväskylä: Kopijyvä, Jyväskylä, Finland. Available at http://www.jyu.fi/musica/miditoolbox/.

[4] Uitdenbogerd, Alexandra and Zobel, Justin. Matching Techniques for Large Music Databases. Available at http://www.cs.northwestern.edu/~pardo/machine_perception_of_music/uitdenbogerd-acm-multimed-99.pdf

[5] Boersma, Paul. Accurate Short-Term Analysis of the Fundamental Frequency and the Harmonics-to-Noise Ratio of a Sampled Sound. Institute of Phonetic Sciences (1993).

[6] Durbin, Richard et al. Biological Sequence Analysis. Probabalistic Models of Proteins and Nucleic Acids. Cambridge University Press (1999). p. 12-27.