Melody Matching Directly From Audio*

Dominic Mazzoni
Carnegie Mellon University
5000 Forbes Ave
Pittsburgh, PA 15213
(412) 268-3069
dmazzoni@cs.cmu.edu

Roger B. Dannenberg
Carnegie Mellon University
5000 Forbes Ave
Pittsburgh, PA 15213
(412) 268-3827
rbd@cs.cmu.edu

ABSTRACT

In this paper we explore a technique for content-based music retrieval using a continuous pitch contour derived from a recording of the audio query instead of a quantization of the query into discrete notes. Our system determines the pitch for each unit of time in the query and then uses a time-warping algorithm to match this string of pitches against songs in a database of MIDI files. This technique, while much slower at matching, is usually far more accurate than techniques based on discrete notes. It would be an ideal technique to use to provide the final ranking of candidate results produced by a faster but lest robust matching algorithm.

1. INTRODUCTION

Today, musicians who wish to locate a particular song can visit any one of a number of websites that let them search a database of music by inputting a few notes from the song (in any key), or even just the melodic contour [1-6]. Realistically, most people are not musically literate and are not capable of transcribing a melody they are hearing in their heads into normal music notation. Even identifying whether the next note in a sequence goes up, down, or stays the same, is beyond the capabilities of many potential users. That is the motivation behind creating an interface where the user only needs to hum the melody he or she would like to search for.

It is not sufficient to rely on a melody transcription algorithm to convert a digital recording of the hummed query into a sequence of notes to search for in a song. Common problems include regions where the pitch tracker cannot lock onto any frequency, octave errors, and segmentation errors (two consecutive notes mistranscribed as a long note or vice versa).

Instead we propose searching for a melody based on the best estimate of the continuous pitch contour derived directly from the audio recording. Speech recognition researchers have discovered time and time again that in the many steps necessary to go from a recording of speaking to the textual transcription, making hard decisions at any step can be disastrous. Guided by this experience, we try to eliminate the transcription steps that quantize pitches and segment them into discrete notes, as this process is certain to introduce errors.

This work is guided by the model of query-by-humming systems. In such a system, the user hums, sings, or whistles (we will refer to any of these simply as "humming"), and the system finds matching entries in a music database. An entry matches if it contains a close match to the hummed query. Since songs are generally considered to be equivalent when performed at a speed or in a different key, the system should be invariant with respect to transposition and tempo.

2. METHODOLOGY

Our idea for searching based on the pitch contour is very straightforward. First use a pitch transcription system to compute the continuous pitch contour of the hummed query. (We distinguish between pitch transcription systems, which simply attempt to determine the pitch being hummed at each point in time, and full melody transcription systems, which attempt to extract a discrete series of notes, each with its own pitch, onset time, and duration.) Overlay the pitch contour on top of every possible place in the song, for every possible pitch offset, and for a range of reasonable time scaling factors. For each position, offset, and time scale, approximate the integral of the difference between the instantaneous pitch at each point in time and the pitch of the song at that point, giving a simple distance measure between the two. The song that contains the minimum distance measure is the one that best matches the query. Because hummed queries are not likely to have a perfectly consistent tempo, we use a dynamic time warping algorithm to allow for small rhythmic differences.

This method is very computationally intensive, and even with heavy optimization it is not likely to be fast enough to be a complete melody-matching solution. However, note that pitch and rhythm are taken into account without relying on pitch quantization, beat induction, or note segmentation. We believe this contributes to the improved quality of the resulting searches.

Here are the details of our implementation. We segment the query and candidate melody into frames of 100 ms. (100 ms was chosen as a compromise between efficiency and accuracy.) Then we run the pitch transcription algorithm on each frame of the audio recording of the humming. The pitch transcription algorithm that we use is based on the *enhanced autocorrelation* algorithm described by Tolonen and Karjalainen [7]. We investigated many other pitch transcription programs, including spectral-based approaches, other autocorrelation methods, and commercial products, but found that choosing the peak of the enhanced autocorrelation signal worked as well if not better than anything else when the goal was simply to come up with one target pitch for each frame. We represented pitches as MIDI note numbers, allowing fractions, so for example 60.13 stands for a pitch 13 cents above middle C. Other details, such as pitch ranges for different singers and silence thresholds, can be obtained directly from our source code, which is freely available on the Internet [8].

*Originally published as: Mazzoni and Dannenberg, "Melody Matching Directly from Audio," in *ISMIR 2001 2nd Annual International Symposium on Music Information Retrieval*, Bloomington: Indiana University, (2001), pp. 73-82.

The songs in our database are all MIDI files, so they also require some preprocessing before we perform our melody-matching algorithm. To compute a string of 100 ms pitch frames from a MIDI file, we consider each MIDI channel separately, and find the note that is most contained in each time frame. If multiple notes are found, we choose the one with the highest pitch. Also, because note releases seem to be much less important perceptual cues than note onsets, and because note releases are performed inconsistently, we extend all notes to the beginning of the next note, thereby eliminating rests in the melody. This mirrors the technique of defining a note's duration as the inter-onset time used in almost all note-based melody matching algorithms. Because the tempo of the query may not have exactly matched the tempo stored in the MIDI file, we repeat this process with different time scaling factors from 0.5 to 2.0, allowing for an opportunity to match a hummed melody from half the speed up to twice the speed.

At this point we have a string of *n* pitches for the query, so for every possible sequence of about *2n* frames from every channel of our MIDI file, we match the query against the database clip using a dynamic programming-based time-warping algorithm, exactly the same as would be found in a limited-vocabulary speech recognition system. To limit the amount of rhythmic variation between the query and the song from the database, we use a *beam width* of *n*/10, ensuring that only paths that do not stray too far from the straight diagonal are allowed. Finally, we run this time warping algorithm 24 times, once for each possible quartertone offset.

3. EXPERIMENTAL RESULTS

In order to compare our approach against other techniques for melody retrieval, we collected a database of MIDI files in different genres and recordings of various people humming melodies from these MIDI files. We use the algorithm discussed in the previous section to compare the query to each song in our database and arrive at a distance between the query and each song. We then ranked the songs according to distance, smallest first, and looked at the rank of the intended song.

Our preliminary results were based on two small databases of MIDI files, one containing 77 big band swing songs, and one containing 18 Beatles songs. (For more recent results, see our website.) Our results were quite promising. Out of Beatles song queries, 9/11 times the correct song had a rank of one, and all 11 times the correct song appeared in the top three. Out of queries of big band songs, 13/20 times the correct song had a rank of one, and 16/20 times the correct song was in the top three. We also implemented a number of more traditional matching algorithms based on strings of discrete notes, and none of these performed as well, mostly because they returned a large number of false positives. The best note-based algorithm we implemented (which incorporated both pitch and rhythmic information) only got the correct song first 5/11 times for Beatles songs, and only got it in the top three 8/11 times. For big band songs, the note-based algorithm got the correct match first 5/20 times, and got it in the top three 6/20 times. This does not mean that it would not be possible for a better note-based algorithm to do much better, and in fact we are making our queries and our database available to any researchers who would like to try, but we feel that no approach of this form is likely to outperform our frame-based method unless there is a major breakthrough in melody transcription software.

4. CONCLUSIONS AND FUTURE WORK

Our frame-based approach shows a lot of promise. It works better than any note-based approach we were able to implement, and more importantly, there are compelling reasons why one would expect this approach to be more accurate.

In spite of these advantages, our approach is not perfect. One potential problem is that singers may change pitch in the middle of a query, and our approach does not currently deal with this as well as an interval-based algorithm. Perhaps the biggest criticism of our work is that it is clearly a brute-force approach and it is very slow. Rather than move from dynamic programming toward sub-linear retrieval algorithms suitable for large databases, we are advocating strings that are much longer than the number of notes. Our searches run orders of magnitude slower than typical note-based searches, and as a result, this algorithm could not be used by itself to drive a content-based music retrieval system.

Still, our approach could also be used behind the scenes to improve faster algorithms: when our frame-based algorithm fails, it is often because the query itself was not particularly good. Thus a researcher could use our more robust algorithm to distinguish between cases where the query was simply no good and cases where a prototype algorithm failed for a different reason.

In the future we would like to improve the speed by using two or more levels of refinement. We would begin with a fast but imprecise algorithm to narrow the search to a small subset of the database, then use successively more precise but more expensive algorithms to arrive at the final result. In addition, we would like to experiment with searching audio data instead of MIDI.

The authors would like to thank Kjell Lemstrom for answering questions about SEMEX and providing some valuable insight. This material is based upon work supported by NSF Award #0085945, an IBM Faculty Partnership Award, and an NSF Graduate Research Fellowship.

5. REFERENCES

[1] R.J. McNab, L. A. Smith, D. Bainbridge and I.H. Witten. The New Zealand Digital Library MELody inDEX (MELDEX). *D-Lib Magazine*, May 1997.

[2] A. Kornstädt. “Themefinder: A web-based melodic search tool.” *Computing in Musicology,* v11, pp. 231-36, 1998.

[3] D. Huron et. Al. *Themefinder* (website). http://www.themefinder.org/

[4] R. Typke. *Tuneserver* (website). http://wwwipd.ira.uka.de/tuneserver/

[5] K. Lemström. “String Matching Techniques for Music Retrieval.” Ph.D. thesis, University of Helsinki, Finland, Nov., 1999.

[6] K. Lemström and S. Perttu. “SEMEX – An Efficient Music Retrieval Prototype.” *Proceedings of the ISMIR 2000.*

[7] T. Tolonen, M. Karjalainen. “A computationally efficient multi-pitch analysis model.” *IEEE Transactions on Speech and Audio Processing*, Vol. 8, No. 6, Nov. 2000.

[8] CMU Music group website: http://www.cs.cmu.edu/~music/