Music Information Retrieval Using Audio Input

Lloyd A. Smith, Rodger J. McNab and Ian H. Witten

Department of Computer Science
University of Waikato
Private Bag 3105
Hamilton, New Zealand
{las, rjmcnab, ihw}@cs.waikato.ac.nz

Abstract

This paper describes a system designed to retrieve melodies from a database on the basis of a few notes sung into a microphone. The system first accepts acoustic input from the user, transcribes it into common music notation, then searches a database of 9400 folk tunes for those containing the sung pattern, or patterns similar to the sung pattern; retrieval is ranked according to the closeness of the match. The paper presents an analysis of the performance of the system using different search criteria involving melodic contour, musical intervals and rhythm; tests were carried out using both exact and approximate string matching. Approximate matching used a dynamic programming algorithm designed for comparing musical sequences. Current work focuses on developing a faster algorithm.

Introduction

Retrieval of musical information presents challenging problems. Although scores are traditionally indexed by title, composer or subject classification, music librarians are often asked to identify a piece of music on the basis of a few sung or hummed notes. The magnitude of this task is indicated by the sizes of national library music collections—the Library of Congress holds over six million pieces of sheet music (not including tens of thousands of operatic scores and other major works), and the National Library of France has 300,000 hours of sound recordings, 90% of it music. Fortunately, sequential pattern matching techniques offer previously unexplored possibilities for searching large corpora of music information.

This paper describes a system that successfully retrieves music on the basis of a few notes that are sung, or otherwise entered. Such a system will form an important component of the digital music library of the future. With it, researchers will analyze the music of given composers to find recurring themes or duplicated musical phrases, and both musicians and casual users will retrieve compositions based on remembered (perhaps imperfectly remembered) musical passages. We assume a database stored in the form of music notation. One current limiting factor is that few score collections are available in machine-readable form. As digital libraries develop, however, scores will be placed on-line through the use of optical music recognition technology. Furthermore, with the increasing use of music notation software, many compositions will be acquired in electronic form—particularly by national libraries, which acquire material through copyright administration. Our test database is a collection of 9400 folk tunes.

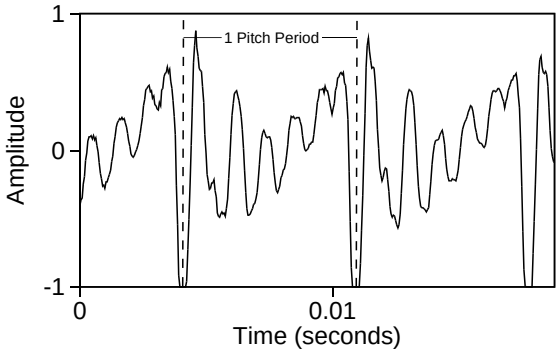
With our system, a user can literally sing a few bars and have all melodies containing that sequence of notes retrieved and displayed—a facility that is attractive to casual and professional users alike. Capitalizing on advances in digital signal processing, music representation techniques, and computer hardware technology, melodies are transcribed automatically from microphone input. Searching large music databases and retrieving items that contain a given theme or sequence of notes is not a trivial undertaking, particularly given the inaccuracies that ensue when people sing known melodies—but, as our scheme demonstrates, it is certainly within the scope of current AI technology. Our system searches a database for tunes containing melodic sequences similar to the sung pattern, and retrieval is ranked according to the closeness of the match. We have experimented with different search criteria involving melodic contour, musical intervals and rhythm; and carried out tests using both exact and approximate string matching. For approximate matching we have used a dynamic programming algorithm designed for comparing musical sequences, and we are currently experimenting with a faster state matching algorithm. Our principal conclusion is that users should be offered a choice of several matching procedures, and be able to explore the results interactively in their search for a particular melody.

Melody Transcription

The front end of our music retrieval system performs the signal processing necessary to automatically transcribe a melody from audio input. The analog acoustic signal is sampled for digital processing, notes are segmented from the acoustic stream, the frequency of each note is identified and each note is labeled with a musical pitch name and a rhythmic value.

Pitch tracking and note segmentation

Our retrieval system runs on an Apple Macintosh PowerPC 8500, which has built-in sound I/O. The acoustic waveform is sampled at 44.1 kHz and quantized to an 8-bit linear representation. For music transcription, we are interested only in the fundamental frequency of the input. Therefore the input is filtered to remove as many harmonics as possible, while preserving the fundamental frequency. Reasonable limits for the singing voice are defined by the musical staff, which ranges from F2 (87.31 Hz) just below the bass staff, to G5 (784 Hz) just above the treble staff. Input is low-pass filtered, with a cutoff frequency of 1000 Hz, then passed to the pitch tracker, which identifies pitch by finding the repeating pitch periods which make up the waveform (Gold & Rabiner 1969). Figure 1 shows 20 ms of a typical waveform for the vowel *ah*, as in *father*. The pitch tracker returns a pitch estimate for each 20 ms segment of input; this gives the system a pitch resolution of approximately five cents, or about 0.29%, which equals human pitch resolution.



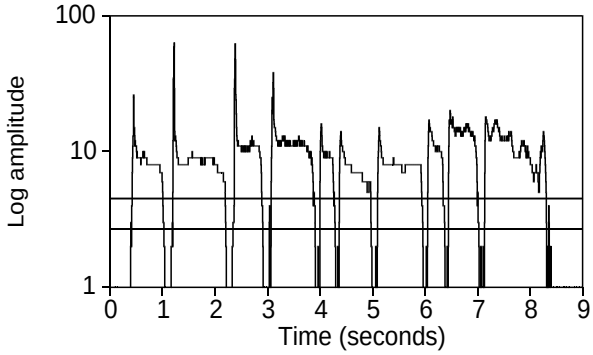


Figure 2. Amplitude segmentation

Pitch representation

Transcription of Western music requires that each note be identified to the nearest semitone. For several reasons, however, it is desirable to represent pitches internally with finer resolution. Our system expresses each note, internally, by its distance in cents above 8.176 Hz, or MIDI note 0. Notes on the equal tempered scale, relative to A-440, occur at multiples of 100 cents; C4, or middle C, is 6000 cents, while A4, or concert A, is 4400 cents. This scheme easily incorporates alternative (non-equitempered) tunings of Western music, such as the "just" or Pythagorean system, simply by changing the relationship between cents and note name. It can also be easily adapted to identify notes in the music of other cultures.

Adapting to the user's tuning

Our system labels each note according to its frequency and a reference frequency. In some applications it is desirable to tie note identification to a particular standard of tuning, such as A-440. For a melody retrieval application, however, it is more desirable to adapt to the user's own tuning and tie note identification to musical intervals rather than to any standard.

The system begins a transcription with the assumption that the user will sing to A-440, but then adjusts by referencing each note to its predecessor. For example, if a user sings three notes, 5990 cents, 5770 cents and 5540 cents above MIDI note 0, the first is labeled C4 and the reference is moved down 10 cents. The second note is labeled Bb3, which is now referenced to 5790 (rather than 5800) cents, and the reference is lowered a further 20 cents. The third note is labeled Ab3, referenced now to 5570 cents—even though, by the A-440 standard, it is closer to G3. Thus the beginning of *Three Blind Mice* is transcribed.

Searching Musical Databases

Retrieving music from a collection of musical scores is essentially a matter of matching input strings against a database. This is a familiar problem in information retrieval, and efficient algorithms for finding substrings in a body of text are well known. Unfortunately, there are

several problems with seeking an exact match between the transcribed melody and the database. The first is the variability in the way that music is performed. Folk songs, for example, appear in many variants, while popular songs and well-known standards are often performed differently from the way they are notated. Performances of classical music generally have a more stable relationship to the score, but there are other sources of error. Problems may be caused by deficiencies in the user's singing—or his or her memory of the tune may be imperfect. We have performed an experiment to determine the accuracy of people in singing well known songs (McNab et al. 1996), and our findings indicate that people make a number of mistakes in singing intervals and rhythms of songs, even when those songs are well known to them.

It is necessary, then, to allow approximate string matching on the score database in order to retrieve music. Approximate matching algorithms are, in general, far less efficient than those which match strings exactly, and invariably take time which grows linearly with database size rather than logarithmically, as does binary search.

Search criteria

What attributes should be used when searching a musical score database? The first point to note is that melodies are recognizable regardless of the key in which they are played or sung—so it is important to allow users to enter notes in any key. This is accomplished simply by conducting the search on the basis of pitch ratios, or musical intervals. Second, a number of experiments have shown that interval *direction*, independent of interval size, is an important factor in melody recognition (Dowling 1978)—indeed, Parsons (1975) has produced an index of melodies based entirely on the sequence of interval directions, which is called the "melodic contour" or "pitch profile." Using the notation of Parsons, where * represents the first note, D a descending interval, U an ascending interval, and R a repetition, the beginning of *Three Blind Mice* is notated: *DDUDDUDRDUDRD

One cardinal advantage of searching on contour, at least for casual singers, is that it releases them from having to sing accurate intervals.

Approximate string matching for music

Approximate string matching is a standard application of dynamic programming. In general, two strings of discrete symbols are given and the problem is to find an economical sequence of operations that transforms one into the other. The basic operations are *deletion* of a single symbol, *insertion* of a single symbol, and *substitution* of one symbol by another. These three operations have associated numeric "costs," or "weights," which may be fixed or may depend on the symbols involved. The cost of a sequence of operations is the sum of the costs of the individual operations, and the aim is to find the lowest-cost sequence that accomplishes the desired transformation. The cost of this sequence is a measure of the distance between the strings. Using dynamic programming, the optimal solution can be found in a time which is proportional to the product of the lengths of the sequences.

Mongeau and Sankoff (1990) adapt dynamic programming to music comparison by adding two music-related operations: *consolidation*, which combines a sequence of notes into one whose duration is their sum and whose pitch is their average (computed with respect to the distance metric), and *fragmentation*, which does the reverse. While Mongeau and Sankoff designed their algorithm to compare complete melodies, it is necessary for a general music retrieval system to allow a search for embedded patterns, or musical themes. This capability is incorporated into the dynamic programming algorithm by modifying the start condition so that deletions preceding the match of the pattern receive a score of 0.

Retrieving Tunes from Folk Song Databases

The weaker the matching criteria, the larger the musical fragment that is needed in order to identify a particular song uniquely from a given corpus. To get a feeling for the tradeoffs involved, we performed an extensive simulation based on two corpora of folk songs. The first is a collection of 1700 tunes, most of North American origin, from the Digital Tradition folksong database (Greenhaus 1994). The other is comprised of 7700 tunes from the Essen database of European and Chinese melodies (Schaffrath 1992). Combining the two sources gave us a database of 9400 melodies. There are just over half a million notes in the database, with the average length of a melody being 56.8 notes.

Retrieval experiments

We are interested in the number of notes required to identify a melody uniquely under various matching regimes. The dimensions of matching include whether interval or contour is used as the basic pitch metric, whether or not account is taken of rhythm, and whether matching is exact or approximate.

Based on these dimensions, we have examined exact matching of:

- interval and rhythm
- interval regardless of rhythm
- contour and rhythm
- contour regardless of rhythm

and approximate matching of:

- interval and rhythm
- contour and rhythm

For each matching regime we imagine a user singing the beginning of a melody, comprising a certain number of notes, and asking for it to be identified in the database. If it is in the database, how many other melodies that begin this way might be expected? We examined this question by randomly selecting 1000 songs from the database, then matching patterns ranging from 5 to 20 notes against the

entire database. This experiment was carried out both for matching the beginnings of songs and for matching sequences of notes embedded within songs. For each sequence of notes, we counted the average number of "collisions"—that is, other melodies that match. Fragmentation and consolidation are relevant only when rhythm is used in the match; in these experiments, fragmentation and consolidation were allowed for approximate matching but not for exact matches.

Figure 3 shows the expected number of collisions, for each of the matching regimes, when queries are matched at the beginnings of songs. The number of notes required to reduce the collisions to any given level increases monotonically as the matching criteria weaken. All exact-matching regimes require fewer notes for a given level of identification than all approximate-matching regimes. Within each group the number of notes decreases as more information is used: if rhythm is included, and if interval is used instead of contour. For example, for exact matching with rhythm included, if contour is used instead of interval two more notes are needed to reduce the average number of items retrieved to one. The contribution of rhythm is also illustrated at the top of Figure 3, which shows that, if rhythm is included, the first note disqualifies a large number of songs. It is interesting that melodic contour with rhythm is a more powerful discriminator than interval without rhythm; removing rhythmic information increases the number of notes needed for unique identification by about three if interval is used and about six if contour is used. A similar picture emerges for approximate matching except that the note sequences required are considerably longer.

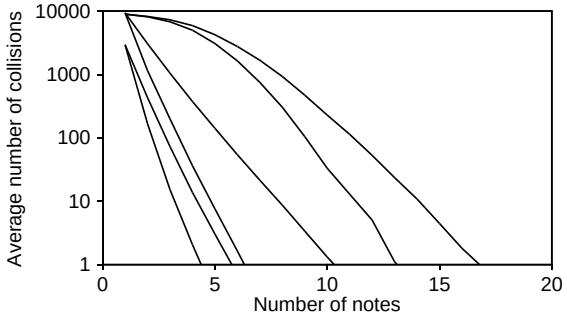


Figure 3. Number of collisions for different lengths of input sequence when matching start of song. From left to right:
- exact interval and rhythm
- exact contour and rhythm
- exact interval
- exact contour
- approximate interval and rhythm
- approximate contour and rhythm

An important consideration is how the sequence lengths required for retrieval scale with the size of the database. Figure 4 shows the results, averaged over 1000 runs, obtained by testing smaller databases extracted at random from the collection. The number of notes required for retrieval seems to scale logarithmically with database size.

Figure 5 shows the expected number of collisions for matching embedded note patterns. As expected, all matching methods require more notes than searches conducted on the beginnings of songs. In general, an additional three to five notes are needed to avoid collisions, with approximate matching on contour now requiring, on average, over 20 notes to uniquely identify a given song.

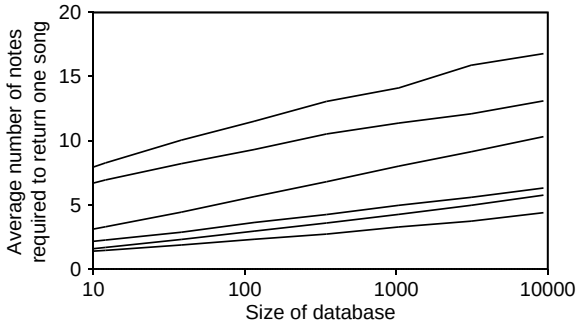


Figure 4. Number of notes for unique tune retrieval in databases of different sizes. Lines correspond, from bottom to top, to the matching regimes listed in Figure 3.

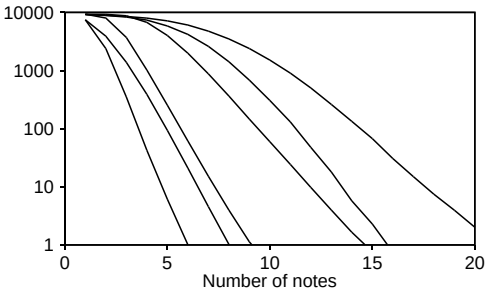


Figure 6 shows a screen display from our melody retrieval system. The transcribed input appears in the *Voice Input* window, while titles of tunes returned by the database search appear in a text window. Tunes are ranked according to how closely they match the sung pattern; scores are calculated by subtracting the dynamic programming distance from 1000. Any of the returned tunes may be selected for display, and the user can listen to the displayed tune.

The number of songs returned can be controlled by a variable score threshold. The search illustrated in Figure 6 returned tunes with a score of 950 or higher; in this instance, the search returned 22 songs. Table 1 shows the number of songs returned, using the search pattern from Figure 6, for the various matching methods. Most of the

matching schemes returned a manageable number of tunes, although, when embedded patterns were matched, approximate contour and rhythm returned over a third of the database; matching exact contour, without rhythm, did not perform markedly better.

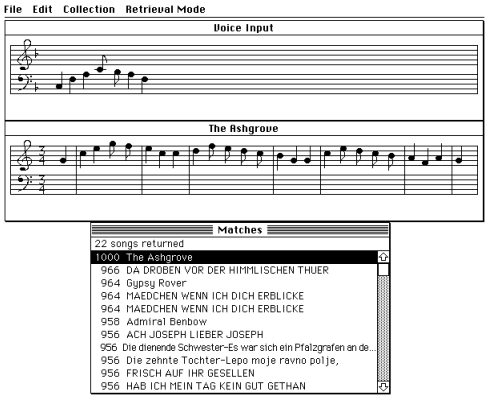


Figure 6. Display from the tune retrieval system

Search Criteria	No. Songs Returned Matching:	
	Start	Embedded Patterns
Exact interval & rhythm	1	1
Exact contour & rhythm	4	14
Exact interval	1	2
Exact contour	153	2603
Approx. interval & rhythm	22	96
Approx. contour & rhythm	349	3595

Table 1. Number of songs retrieved using Figure 6 input

On the PowerPC 8500, with a clock speed of 120 MHz, pitch tracking and display of audio input takes approximately 2.8% of recorded time; the input for Figure 6 was processed in less than 230 ms. Exact matching on the 9400 song database takes about 500 ms, but approximate matching can take much longer. Matching a 20 note search pattern, for example, requires approximately 21 seconds. While it may be reasonable to expect a user to wait that length of time for a search to complete, much larger databases—a million folk songs, for example, or a thousand symphonies—will take an unacceptably long time to search. We are currently investigating the use of a state matching algorithm (Wu and Manber 1992) for reducing the time taken for approximate searches.

One way of speeding retrieval based on embedded patterns is to automatically identify themes using an offline matching method, storing those themes in a separate collection indexed to the original database. Because themes are relatively short (in comparison to an entire composition), a theme database can be searched much more quickly; furthermore, it is unnecessary to search for embedded patterns in a database containing only themes.

Conclusion

We have presented and analyzed a novel scheme for audio information retrieval by searching large databases of musical scores. Searching such corpora requires efficient string matching algorithms. Previous experiments on melody recognition suggest that search should be carried out on the basis of melodic contour and/or musical intervals, and our own experiments suggest that there should be provision for approximate matching of the input, and that the audio transcription module should adapt to the user's musical tuning, which may vary during input.

The time taken to perform approximate matches in large databases of musical scores is a matter for some concern. We are investigating two ways of speeding these searches. One approach is to use a fast approximate search method (Wu and Manber 1992), suitably guided by knowledge of the errors people make in singing well known melodies. Another possibility is to automatically create, offline, databases of themes which allow fast indexing into the main database. It may be possible, for example, to use the Mongeau and Sankoff algorithm to find recurring themes in symphonies or popular songs; these themes can then be stored in a separate, and much smaller, database.

Our investigations have focused on retrieval of musical scores. While it may someday be feasible to directly match acoustic input against digital audio files, it is likely that the musical score will be an intermediary representation for some time to come. We envision a system where the user might whistle the theme to Grieg's Piano Concerto in A Minor; this input is then matched to a database of musical scores, and the corresponding recording is returned to the user's terminal. We believe that acoustic interfaces to online music databases will form an integral part of the digital music library of the future.

References

Dowling, W. J. 1978. Scale and Contour: Two Components of a Theory of Memory for Melodies. *Psychological Review* 85: 341–354.

Gold, B., and Rabiner, L. 1969. Parallel Processing Techniques for Estimating Pitch Periods of Speech in the Time Domain. *J. Acoust. Soc. Am.* 46: 442-448.

Greenhaus, D. 1994. About the Digital Tradition. http://www.deltablues.com/DigiTrad-blurb.html.

Schaffrath, H. 1992. The ESAC Databases and MAPPET Software. In *Computing in Musicology, Vol 8.* , ed. W. Hewlett and E. Selfridge-Field. Menlo Park, Calif.: Center for Computer Assisted Research in the Humanities.

McNab, R.J., Smith, L.A., Witten, I.H., Henderson, C.L., and Cunningham, S.J. 1996. Towards the Digital Music Library: Tune Retrieval From Acoustic Input. In Proceedings of ACM Digital Libraries '96, 11–18.

Mongeau, M., and Sankoff, D. 1990. Comparison of Musical Sequences. *Computers and the Humanities* 24: 161–175.

Parsons, D. 1975. *The Directory of Tunes and Musical Themes.* Cambridge: Spencer Brown.

Wu, S., and Manber, U. 1992. Fast Text Searching Allowing Errors. *Commun. ACM* 35(10), 83–91.