Tune Retrieval in the Multimedia Library

Rodger J. McNab[1], Lloyd A. Smith[1],
Ian H. Witten[1] and Clare L. Henderson[2]
[1]*Department of Computer Science*
[2]*School of Education*
University of Waikato, Hamilton, New Zealand.
{rjmcnab, las, ihw, clhend}@waikato.ac.nz

Abstract

Musical scores are traditionally retrieved by title, composer or subject classification. Just as multimedia computer systems increase the range of opportunities available for presenting musical information, so they also offer new ways of posing musically-oriented queries. This paper shows how scores can be retrieved from a database on the basis of a few notes sung or hummed into a microphone. The design of such a facility raises several interesting issues pertaining to music retrieval. We first describe an interface that transcribes acoustic input into standard music notation. We then analyze string matching requirements for ranked retrieval of music and present the results of an experiment which tests how accurately people sing well known melodies. The performance of several string matching criteria are analyzed using two folk song databases. Finally, we describe a prototype system which has been developed for retrieval of tunes from acoustic input and evaluate its performance.

Keywords: Music retrieval, melody recall, acoustic interfaces, relevance ranking.

1. Introduction

Music librarians are often asked to find a piece of music based on a few hummed or whistled notes. The magnitude of this task may be judged by the fact that the Library of Congress holds over six million pieces of sheet music—not including tens of thousands of operatic scores and other major works (Goodrum and Dalrymple, 1982). As digital libraries develop, these collections will be placed on-line through the use of optical music recognition technology (Carter, 1989; Selfridge-Field, 1994; Bainbridge and Bell, 1996). Furthermore, with the increasing use of music notation software for composition, it is likely that many compositions will be acquired in computer readable form—particularly by national libraries, such as the Library of Congress, that acquire material through copyright administration.

The possibility of electronically searching corpora of music raises the interesting prospect of retrieval based on direct acoustic input. A user could literally sing a few bars and have all melodies containing that sequence of notes retrieved and displayed—a facility that would be attractive to casual and professional users alike. With advances in digital signal processing, music representation techniques, and computer hardware technology, it is becoming feasible to transcribe melodies automatically from microphone input. For example, a person can sing a tune and have it printed in ordinary musical notation. Searching large music databases and retrieving items in which a given theme or sequence of notes occurs is not a trivial undertaking, particularly given the inaccuracies that occur when people sing known melodies, but it is certainly within the scope of current technology.

This capability 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 musical passages—perhaps imperfectly remembered. Moreover, the underlying technology for transcribing from acoustic melodies will facilitate the transcription of monophonic material such as field recordings of folk songs.

Like most multimedia projects, developing a system for retrieving music from acoustic input is a multidisciplinary undertaking and, while there has been some work done on the individual components, there has been little effort to integrate them into a single system. Computer-based pitch trackers, which identify the frequency of acoustic input, have been around for thirty years or more, and hundreds of different algorithms have been developed (Hess, 1983). Askenfelt (1978) describes a system that automatically transcribes folk songs into music notation; however, the system requires extensive human intervention to correct wrongly transcribed pitches and rhythms. Hawley (1990) developed a system to search a tune database, given a note sequence entered via MIDI keyboard, but his system is inflexible in its retrieval strategy, returning only tunes whose beginnings exactly match the input. A great deal of work has been done in approximate string matching, particularly in identifying substrings in text (Wu and Manber, 1992) or genetic sequences (Bishop and Thompson, 1986).

Our goal is to provide detailed design information, and a prototype system, encompassing all these aspects of a music retrieval facility. The system will transcribe acoustic input, typically sung or hummed by the user, and retrieve music, ranked by how closely it matches the input. It must operate on a substantial database of realistic size, and retrieve information in at most a few seconds. It is necessary to allow for inaccurate singing or imperfect memory on the part of the user, for variation in the way music is performed, and for differences between music as it is notated and performed. In order to take into account human inaccuracies of recall and of performance, we must model the kinds of errors that people make in remembering melodies, and in singing them; and devise flexible retrieval mechanisms that are tailored to the errors actually encountered in practice.

Ghias *et al.* (1995) describe a system developed along the same lines, but which is rather more limited than our endeavor. The user hums a tune and the system tracks it using an autocorrelation method, converts it to a melodic contour, and matches this against a database of 183 songs. Ranked retrieval is performed, based on an approximate string matching algorithm that allows for replacement, dropout and duplication—although the system does not search for themes within songs. The biggest problems are the time taken to perform pitch tracking, the restriction to humming, the lack of attention to human performance in melody recall, and the very small database size. The present paper is a more comprehensive account of a somewhat more ambitious system.

The paper is organized as follows. First we review the state of the art in signal processing for melody transcription, since this represents the main retrieval interface for tunes. The next question is how to match user input against the database. The requirements for the matching operation of a usable and useful library retrieval system turn out to be rather subtle, and in order to identify them we work from three directions. First we study the question of melody matching for music retrieval. Then we describe a pilot experiment on how people perform when asked to sing well-known songs. Next we analyze the characteristics of a test database of ten thousand folk songs in order to quantify how much information is required to identify tunes based on partial information of different types. These studies provide design information for a melody matching system that will satisfy users' requirements. Finally, we present an initial system for identifying and retrieving tunes from acoustic input. This is a "proof of concept" prototype that accepts acoustic input, typically sung by the user, displays it in standard music notation, and retrieves appropriate matching tunes from a database.

2. Automatic Transcription of Melodies

Accepting acoustic input for musical retrieval is essentially a problem in music transcription. 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. This section briefly describes how these problems are solved by a system for melody transcription called MT (McNab *et al.,* 1996), which forms the signal processing front end for our acoustic music retrieval system.

2.1 The musical scale

A musical scale is a logarithmic organization of pitch based on the *octave*, which is the perceived distance between two pitches when one is twice the frequency of the other. For example, middle C (C4) has frequency 261.6 Hz; the octave above (C5) is 523.2 Hz, and the octave below (C3) is 130.8 Hz.

Although the octave seems to be a perceptual unit in humans (Deutsch, 1972), pitch organization within the octave takes different forms across cultures. In Western music, the primary organization since the time of Bach has been the equal-tempered scale, which divides the octave into twelve equally spaced *semitones*. The semitone is the smallest unit of pitch in Western music, but smaller units can easily be perceived and are used in the music of some cultures (Backus, 1969). The *cent* is defined as one hundredth of a semitone in the equal tempered scale. An octave, then, is 1200 cents. The smallest pitch difference between two consecutive tones that can be perceived by humans is about 3 Hz; this yields a pitch discrimination of about five cents at 1000 Hz. Above 1000 Hz discrimination stabilizes at about 4 cents.

While pitch may be represented categorically in terms of octaves, semitones and cents, frequency is continuous. Assigning a musical pitch to a given frequency involves *quantization*. Semitone resolution is sufficient to quantize pitches based on a particular tuning standard (A-440, for example). To accommodate different tuning systems, however—including adapting to users, who inevitably sing slightly sharp or flat—higher resolution is essential. MT is designed around a pitch resolution of five cents (0.29%).

2.2 The MIDI note representation

Since musical units—octaves, cents and so forth—are relative measures, a distance in cents could be calculated for each individual interval sung by the user. A fixed reference point, however, allows easier integration with applications. MIDI (Musical Instruments Digital Interface) is a standard for controlling and communicating with electronic musical instruments. It has many facets, the one most germane to our melody transcription system being its standard representation of the Western musical scale. MIDI assigns an integer to each note of the scale. Middle C (C4) is assigned 60, the note just above (C#4) is 61, and that below (B3) is 59. Although it makes little sense to assign pitch labels to frequencies below about 15 Hz, MIDI note 0 is 8.176 Hz, an octave below C0. The highest defined note, 127, is 13344 Hz, again not likely to be perceived as a musical note. The standard piano keyboard ranges from notes 21 to 108.

In our melody transcription system, all pitches are related internally to MIDI notes, each being expressed as a distance in cents from 8.176 Hz. Notes on the equal tempered scale relative to A-440 occur at multiples of one hundred cents: C4, for example, is 6000 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 adapted to identify notes in the music of other cultures.

2.3 Sampling and filtering

For music transcription, we are interested only in the fundamental frequency of the input. Harmonics, which occur at integral multiples of frequency, often confuse pitch trackers and make it more difficult to determine the fundamental. 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. While ledger lines are used to extend the staff in either direction, these represent extreme pitches for singers and are unnecessary for music retrieval in the databases we are currently considering.

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. Input is low-pass filtered with cutoff frequency of 1000 Hz, stopband attenuation –14 dB, and passband ripple of 2 dB. These are not stringent design requirements, and can be met by a digital finite impulse response (FIR) filter having nine coefficients (Steiglitz, Parks and Kaiser, 1992). The filtered signal is passed to the pitch tracker, which identifies its fundamental frequency.

2.4 Pitch tracking and note segmentation

Sounds that are perceived as having pitch are made up of a number of recurring *pitch periods.* Algorithms for identifying the pitch of an acoustic signal may be classified by whether they work in the time domain, by examining the structure of the sampled waveform, the frequency domain, by examining the spectrum generated by a Fourier transform, or the cepstral domain, by performing a second Fourier transform on the log amplitude spectrum and examining the resulting cepstrum (Hess, 1983). It was not our purpose to perform research into pitch tracking—our focus is on the integrated multimedia application. For that reason, we chose to use the Gold-Rabiner pitch tracking algorithm (Gold and Rabiner, 1969), a time domain method which is well understood and documented, and which has become something of a standard against which other algorithms are compared; if another algorithm seems more appropriate at some point, we can replace our current pitch tracker without affecting the modules that use its output.

The Gold-Rabiner algorithm assigns pitch by finding the repeating pitch periods comprising the waveform. Figure 1 shows 20 ms of a typical waveform for the vowel *ah,* as in *father.* Our implementation of the algorithm breaks the input sound into 20 ms *frames* and returns a pitch estimate for each frame.

Once pitches have been identified, it is necessary to determine where notes begin and end. We have developed two ways of doing this, one based on amplitude and the other on pitch. Amplitude segmentation is simpler, but depends on the user's separating each note by singing *da* or *ta*—the consonant causes a drop in amplitude of 60 ms duration or more at each note boundary. Adaptive thresholds are then used to determine note onsets and offsets; in order to keep a marginal signal from oscillating on and off, the onset threshold is higher than the offset threshold. Figure 2 illustrates the use of amplitude to segment a series of notes.

The alternative to amplitude segmentation is to segment notes directly from the pitch track by grouping and averaging 20 ms frames. An adjacent frame whose frequency is within 50 cents of a growing note segment is included in that segment. Any segment longer than 100 ms is considered a note. Pitch based segmentation has the advantage of relaxing constraints on the user, but may not be suitable for all applications—repeated notes at the same pitch may not be segmented, while a slide, or *glissando,* is segmented into a sequence of ascending or descending notes.

After note onsets and offsets are determined, rhythmic values are assigned by quantizing each note to the nearest sixteenth according to the tempo set by the user.

2.5 Adapting to the user's tuning

MT labels a note by its MIDI number according to its frequency and the current reference frequency. In some applications it is desirable to tie note identification to a particular standard of tuning. In others it is more desirable to adapt to the user's own tuning and tie note identification to musical intervals rather than to any standard. MT is able to do either.

In adaptive tuning mode, the system assumes 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 (MIDI 60) 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.

While constantly changing the reference frequency may seem computationally expensive, it is efficiently implemented as an offset in MIDI note calculation. If tuning is tied to a particular standard, the offset is fixed. To use a fixed A-440 tuning, for example, the offset is fixed at 0.

3. String Matching for Music Retrieval

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. Tunes that *begin* with a certain sequence of notes can be found by the standard search techniques of binary search or hashing, while tunes that *contain* a certain sequence of notes can be found by standard string-matching methods such as the Knuth-Morris-Pratt or Boyer-Moore algorithms, or Rabin-Karp signature matching (Sedgewick, 1988). These algorithms find strings that match the input exactly (or, in the case of binary searching, find the match which is closest in lexicographic order). This is not suitable for matching music based on acoustic input.

There are several problems with seeking an exact match between input string and database. The first is the variability in the way that music is performed. Folk songs, for example, appear in many variants (Sundberg and Lindblom, 1976). This applies not only to songs that have been handed down orally from generation to generation, but also to composed songs that have recently entered the folk tradition (Cohen and Cohen, 1973). Popular songs and well-known standards are often performed differently from how they appear in the score (Bauer, 1988). Performances of classical music generally have a more stable relationship to the score. However, there are other sources of error. Problems may be caused by deficiencies in the user's singing efforts—or his or her memory of the tune may be imperfect. Sloboda (1982) reports that people often distort and recombine melodic fragments in complex ways, changing melodic contours, intervals and tonalities; our own studies confirm this.

It is necessary, then, to perform 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 in the case of binary search.

3.1 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 what key they are played or sung in—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.

3.2 Approximate string matching for musical sequences

The problem of approximate string matching was formulated in the early 1970s as a standard application of dynamic programming (Wagner and Fischer, 1974). 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" which may be fixed or may depend on the symbols involved: in the case of deletion and insertion the cost might depend on the symbol, while for substitution it might depend on some measure of "distance" between the two symbols. 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. The problem can be augmented by adding new operators such as *transposition* of adjacent symbols, and the basic dynamic programming solution can be extended quite easily to handle this (Lowrance and Wagner, 1975).

The dynamic programming algorithm for matching sequence *a* against sequence *b* is given by equation 1 (Sankoff and Kruskal, 1983).

$$d_{ij} = \min[d_{i-1,j} + w(a_i, \emptyset), d_{i-1,j-1} + w(a_i, b_j), d_{i,j-1} + w(\emptyset, b_j)] \quad (1)$$

where $1 \leq i \leq$ length of sequence a
$1 \leq j \leq$ length of sequence b
$w(a_i, b_j)$ is the cost (or weight) of substituting element a_i with b_j
$w(a_i, \emptyset)$ is the cost of inserting a_i
$w(\emptyset, b_j)$ is the cost of deleting b_j
d_{ij} is the accumulated distance of the best alignment ending with a_i and b_j

Initial conditions are:

$$d_{00} = 0 \quad (2)$$
$$d_{i0} = d_{i-1,0} + w(a_i, \emptyset), i \geq 1 \quad (3)$$
$$d_{0j} = d_{0,j-1} + w(\emptyset, b_j), j \geq 1 \quad (4)$$

The algorithm is usually designed to be symmetric, meaning that matching sequence *a* with sequence *b* returns the same result as matching *b* with *a.* In order for the algorithm to be symmetric, the cost of an insertion must be equal to the cost of a deletion, and $w(a_i, b_j) = w(b_j, a_i)$. There is no penalty for substituting an element with itself, i.e.

$$w(a_i, b_j) = 0, a_i = b_j \quad (5)$$

This methodology can be applied to (monophonic) music by regarding a melody as a sequence of notes, each with an associated pitch and rhythm. Rests are dummy notes with only the duration specified. A distance metric between notes can be constructed by defining the distance between two pitches, and between two rhythms, and coming up with a suitable way of combining these components. Deletion and insertion can be handled by transforming a note to a notional zero-length note, and vice versa. Further operations are desirable: the *consolidation* operator, 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 the *fragmentation* operator which does the reverse. Of course, the same effect can be achieved by successive insertions or deletions, along with an appropriate substitution to adjust the rhythm, but in certain circumstances—for example, when the notes all have the same pitch—the consolidation and fragmentation operations involve much smaller costs. This reflects the fact that in these situations, a fragmentation or consolidation makes a less musically significant change to the melody than an equivalent sequence of basic operations.

Mongeau and Sankoff (1990) have performed an extensive study of the comparison of musical sequences using this approach. They measure pitch differences in a way that gives consonant intervals like octaves and fifths a smaller distance than dissonant intervals such as seconds and sevenths. Scale degrees (*do* = 1, *re* = 2, and so forth) are used to allow melodies in minor keys to be mapped to ones in major keys without excessive penalty. Rhythmic differences are measured by subtracting the lengths of the notes. Mongeau and Sankoff define the distance between notes as a linear combination of the distances between pitches and rhythms; the relative weight of pitch vs. rhythm is determined heuristically by analyzing particular tunes and standard variants.

So far, we have discussed how to match one complete melody against another. To locate a fragment in a melody database in prefix-match mode, a sequence of contiguous insertions leading right up to the end of a melody receives zero penalty.

3.3 Searching music databases

One consideration in designing a music retrieval system is whether to search only at the beginnings of musical scores, or to search for an occurrence of the input pattern of notes anywhere in the score. Obviously, searching for embedded substrings dramatically increases the complexity of the search. In searching musical databases, the type of music to be retrieved will determine the necessity of searching for embedded patterns. A folk song database, for example, contains mostly strophic songs which people are likely to sing from the beginning. Users will search databases of symphonies and other instrumental compositions, however, by singing a theme from some point in the composition. Searching from the beginning may be acceptable if the user is looking for Beethoven's Fifth Symphony, but will not produce the desired result if the user sings the *Ode to Joy* theme from his Ninth Symphony, or the theme from Grieg's Piano Concerto in A Minor.

Problems also arise with other song databases. Show songs are generally composed with a verse and a chorus, where the chorus is likely to be the familiar part of the song—indeed, few people will even be aware of the verse. Similarly, operatic arias are often preceded by a *recitative* that is unfamiliar to most people.

For these reasons, we believe that a general purpose music retrieval system must have the capability to return songs based on matching embedded strings. Mongeau and Sankoff (1990) suggest "inverting" the dissimilarity score into a "quality" function which is to be maximized. This, however, is a cumbersome method, requiring some interpretation of the quality value. A simpler method is to modify the dynamic programming starting condition so that deletions preceding the match of the pattern are given a score of 0. The only change necessary is to equation 4 (Galil and Park, 1990):

$$d_{0j} = 0, j \geq 1 \qquad (6)$$

4. Human Performance in Melody Recall

Experiments described in the literature focus on people's recognition of well known melodies; these experiments indicate the importance of melodic contour and interval in melody recall (Dowling, 1978). For a music retrieval system, however, we must know not only how people *recognize* melodies, but also the ways in which people *generate* them. In order to get some idea of the kind of input we can expect to our music retrieval system, we performed an experiment to find out how people sing well known tunes.

4.1 Method

Ten subjects were each asked to sing ten songs from memory. The songs, listed in Table 1, are all well known in the popular culture, and include folk songs, standards, show songs and popular songs (*Pokare kare ana* is a New Zealand Maori folk song). Subjects represented a wide range of musical background; three had degrees in music and three others had ten or more years of musical training on an instrument. One subject had two years training on piano; the remaining three subjects had little formal musical training.

Subjects were invited to practice singing each song in order to refresh their memories and to decide what key to sing in, then the investigator taped the following performance for later analysis. Subjects were not expected to sing songs in their entirety—rather, they were instructed to sing as much as they knew. The focus was on the tunes; subjects were encouraged to sing words or any comfortable syllable, such as *la* or *da*. Composed songs were compared against publisher's sheet music to determine accuracy of melodic contour and interval sequences; folk songs (*Yankee Doodle* and *Pokare kare ana*) were compared against a well known version chosen as the norm for this experiment. If a subject could not sing a particular song on the list, he or she was asked to substitute a song from an alternate list of five songs. Because few people used the list of alternate songs, the alternates were not analyzed closely.

4.2 Results

Table 1 lists the number of people attempting each song. All subjects were able to sing at least one phrase of each song attempted with fewer than three errors in contour. While contour was generally correct, however, repetitions of the same note were very sensitive to subjects' perceptions of the words. In some cases, for example, singers added extra syllables at the same pitch, or left one or more syllables out. Omitted notes were infrequent, but did occur occasionally; in fact, about half the subjects omitted the descending interval at the end of the second and fourth bars of *Yankee Doodle*. Subjects sometimes added a passing note between notes a third apart—this is a common phenomenon reported by Sloboda (1982).

Subjects started at the beginning of all songs except *Bridge Over Troubled Water*, which seven of nine started at the chorus.

Singers were considered to end *in key* if they finished singing a song within 25 cents (sharp or flat) of the key they started in. Table 1 shows that, in half the songs, subjects were generally able to sing a phrase or more while staying in key. These songs tended to have a narrow range, with predominantly stepwise melodic movement and repetitive melodic patterns. Several songs were particularly difficult. In *Yesterday*, measures two and three suggest a change of tonality from a major key to the melodic scale of its relative minor three semitones below. Subjects were generally unable to negotiate this change of tonality, with nine of the ten missing the accidentals (added sharps or flats) in measure two. Seven of the nine then continued in a new key, while two returned to the original key. *Moon River* presents a challenge with the first interval—an ascending fifth (seven semitones) between the first two notes. Four singers missed this interval, with the inaccuracy putting them into a new key. In *King of the Road,* the problem interval is an augmented fourth (six semitones) between the second and third bars. *Summertime* exhibits a number of major and minor thirds (four and three semitones, respectively), both ascending and descending, in the first two phrases, with one descending fourth (five semitones) as well. Five of the nine subjects attempting this song were unable to accurately negotiate these intervals, with four of the five landing in a new key following these phrases.

In general, intervals greater than two semitones caused problems for singers, and the songs performed most accurately were those with largely stepwise movement. Wide leaps were often "compressed," with subjects singing the top note of an ascending interval flat, or the bottom note of a descending interval sharp; this was particularly noticeable on fifths and sixths (seven to nine semitones). On the other hand, subjects tended to "expand" sequences of smaller intervals—flatting, for example, on a descending sequence of tones and semitones and sharping on ascending stepwise or arpeggiated (three or more consecutive thirds and fourths) sequences.

Some subjects decorated their performances with anticipations and slides. This was often a stylistic effect, on *Summertime* and *Memories*, for example, but happened also on songs that singers knew particularly well—*Pokare kare ana*, *Yankee Doodle*, and *Puff, the Magic Dragon*.

Interestingly, subjects' accuracy was more dependent on singing experience rather than on musical training. Three subjects, for example, had random pitch problems throughout their performances. Two of these were instrumentalists with extensive musical backgrounds—one with a degree in music. Subjects who had sung extensively in amateur choirs were the most accurate. This suggests that subjects' performance accuracy depends more on motor training rather than on ability to hear musical differences—which further implies that an acoustic interface for musical subpopulations of users should be as forgiving as one designed for general use. Alternatively, it may be useful to provide a MIDI keyboard for input from musically trained users.

4.3 Discussion

The results of the experiment are illuminating in terms of devising a practical matching strategy, indicating the need for an approximate matching algorithm such as the one devised by Mongeau and Sankoff (1990). The tendency of subjects to add or delete syllables calls for fragmentation and consolidation procedures; these procedures will also deal with long slides which are broken into multiple notes by a pitch-based note segmentation mechanism. Missed notes, particularly as exhibited in the three "problem" songs—*Yesterday, Moon River,* and *Summertime*—indicate the need for allowing replacement. Insertion and deletion are required in order to handle omitted notes and added passing notes.

In most of the cases presented here, subjects started at the beginning of the song. However, on one of the commercial popular songs (*Bridge Over Troubled Water*), most subjects started at the chorus. The structure of modern popular music, where songs are designed with a memorable "hook" line to capture listeners, means that singers are likely to start at the hook—which often occurs at the beginning of the chorus— rather than at the beginning of a song. As mentioned above, people are also likely to sing show songs from the chorus rather than from the verse. As a preliminary to this experiment, a number of people were asked to sing the first phrase of *Maria.* Most people started at the beginning of the chorus ("Maria, Maria, I just met a girl named Maria…"), but one person, who had recently performed in an amateur production of *West Side Story*, started at the beginning of the verse ("The most beautiful sound I ever heard…"). Similarly, while only two people in the experiment sang Jerome Kern's *Old Man River* (from the alternate list), both started at the chorus ("Old man river…") rather than at the beginning of the song ("Here we all work on the Mississippi…").

Because singers tend to compress wide intervals and stretch small ones, it is useful for the system to adapt to the user's gradually changing tonality. Abrupt changes in tonality, such as those occurring in subjects' performances of *Yesterday*, are adequately handled by a replacement operation along with a general strategy of matching intervals rather than absolute notes.

Anticipations are notated by our music transcription system, and appear as a modification of rhythm. While it may be possible to use musical knowledge to deal with this in the front end transcription module, a more general approach is to accommodate these differences in the matching strategy. It is difficult to know how to handle slides. Because our system determines the pitch of a note by a weighted average of the input frames, a short slide will be incorporated into the target note, but may affect the pitch. A longer slide will be broken into two or more notes; the extra notes generated in this way should either be deleted by the matching algorithm, or grouped together (consolidated) to match one note in the database.

While the general approximate matching scheme of Mongeau and Sankoff is well supported by the experiment, there is no indication that match weights should be based on musical consonance rather than absolute distance in semitones.

5. 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 the Digital Tradition collection of mostly American folk songs. This contains 1700 songs including a small number of duplicates (14) which were removed. The other is the Essen database of around 8300 melodies, about 6000 of which are German folk songs, 2200 are Chinese, and the remainder are Irish. Nearly 400 duplicates—the same song with a different name, perhaps in a different key—are present, and were removed. Because our music transcription system does not currently display triplets, the approximately 200 songs containing triplets were also removed. Combining the two sources and eliminating the three songs common to both collections 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.

5.1 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; whether matching is exact or approximate, with the possibility of note deletion, insertion or substitution; and whether attention is paid to note fragmentation and consolidation.

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

- interval and rhythm;
- contour and rhythm;
- interval regardless of 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 c_n 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.

5.2 Results of retrieval experiments

Figure 3 shows the expected number of collisions plotted against n, 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.

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.

6. A System for Tune Retrieval

We have developed a system, based on the melody transcription program described above, for retrieving tunes from the combined Essen and Digital Tradition folk song database. The user starts singing on any note, and the input is notated in the key that yields the fewest accidentals. Transcription operates in adaptive mode, adjusting to the user’s gradually changing tuning.

The user is able to retrieve folk tunes using an exact match of pitch, pitch and rhythm, melodic contour or contour and rhythm, or an approximate match of pitch and rhythm or contour and rhythm. The system allows the choice of matching from the beginnings of songs or searching for themes within songs. Approximate searching incorporates fragmentation and consolidation, as described by Mongeau and Sankoff (1990), and all retrievals are ranked—exact retrieval simply means that only tunes that match with the maximum possible score are retrieved. Our dynamic programming match algorithm is a *minimization* technique, with the perfect score being zero. In order to provide a more intuitive score for users, the dynamic programming score of each song is subtracted from 1000, with a lower limit of 0, so scores can range from 0 to 1000. Songs are ranked by score; songs with equal scores are listed alphabetically by title.

Figure 6 shows the tune retrieval screen following a search. The names and corresponding scores of retrieved melodies are displayed in a text window and the tune of the best match is displayed in a melody window. The user may select other melodies from the list for display. The figure displays the best match, *The Ash Grove,* following an approximate search on pitch and rhythm. In this instance, the search returned 22 songs. The number of collisions for approximate matches can be controlled using a variable retrieval threshold; the search illustrated in Figure 6 returned songs with a score of 950 or better. Table 2 shows the number of songs retrieved, using the input pattern from Figure 6, for the various matching methods. Most matching schemes return a manageable number of songs, although, if the input string is allowed to occur anywhere in the song, approximate contour and rhythm returned over a third of the database; matching exact contour did not do a great deal better. These results clearly indicate that a longer query pattern is needed for matching exact contour or contour and rhythm.

On the PowerPC 8500, with a clock speed of 120 MHz, pitch tracking and display of 10 seconds of audio input takes less than half a second (the input for Figure 6 was processed in 230 ms). Our system currently uses approximate algorithms for all matches, with exact retrieval simply returning only songs with a perfect score, but we project exact searches using a fast string matching algorithm, such as Boyer-Moore (Sedgewick, 1988), to complete in one second or less.

Approximate matching takes much more time. Retrieval based on the seven note sequence in Figure 6, for example, takes 10.9 seconds; searching for embedded themes takes 19.1 seconds. If fragmentation and consolidation are disallowed, matching from the beginning takes 10.4 seconds, while searching for themes takes 17.5 seconds. While this may be a reasonable time to ask users to wait for a retrieval, much larger databases—a million folk songs, for example, or a thousand symphonies—might take an unacceptably long time to search. The time taken for matching increases linearly with the size of the database (we assume search patterns sung by users will be approximately the same length regardless of the size of the database). There are approxmimate string matching algorithms that have the potential to speed up approximate searches (Wu and Manber, 1992). We feel it is necessary for any such algorithm to provide comparable retrieval performance to the Mongeau and Sankoff algorithm; the results of our human performance experiments will be useful in specifying how those algorithms might operate. 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), the theme database can be searched much more quickly; furthermore, it is unnecessary to search for embedded patterns in a database containing only themes.

6. Conclusion

We have presented and analyzed methods for accessing an online musical score database using microphone input. Searching such a database requires efficient string matching algorithms. Previous experiments that test melody recognition suggest that search should be carried out on the basis of melodic contour and/or musical intervals. The results of a new experiment testing people's accuracy in singing well known melodies suggests that there should be provision for approximate matching of the input, and that the music transcription module of the interface should adapt to the user's musical tuning, which may vary during input.

Analysis of two folk song databases provides some idea of the number of notes needed to perform a useful retrieval under various matching regimes. For a database of ten thousand songs, four to six notes are usually enough for exact retrieval if rhythm is included in the match. If rhythm is not included, one or two more notes are needed. Approximate search, in general, requires twelve notes or more to keep the number of retrieved songs manageable. For all search methods, several more notes are needed if the query pattern is allowed to occur anywhere in the song.

We have implemented a prototype tune retrieval system which accepts input from a user singing into a microphone, transcribes the input into musical notation, and retrieves songs from a database of 9400 folk tunes. The system supports both exact and approximate searches based on either pitch or melodic contour; rhythm is optional for exact searches. Approximate searches perform relevance ranking on retrieved tunes, with scores ranging from 0 to 1000.

We have some concern over the time taken to perform approximate matches in large databases of musical scores. 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.

To this point, our investigations have focused on retrieval of musical scores—we have not yet considered retrieval of audio files and recordings. 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. The acoustic interface would then be just one aspect of a multimedia system such as that described by Loeb (1992).

We believe that acoustic interfaces to online music databases will form an integral part of the digital music library of the future.

References

Backus, J. (1969). *The Acoustical Foundations of Music.* Norton and Co., New York.

Bainbridge, D. and Bell, T.C. (1996) "An extensible optical music recognition system." *Proc. 19th Australasian Computer Science Conf.,* 308–317, Melbourne, January.

Bauer, B. (1988) *The new Real Book*. Sher Music Co, Petaluma, CA.

Bishop, M. J. and Thompson, E. A. (1986) "Maximum likelihood alignment of DNA sequences." *J. Molecular Biology 190:* 159–165.

Carter, N.P. (1989) *Automatic recognition of printed music in the context of electronic publishing*. Ph.D. thesis, University of Surrey, UK; February.

Cohen, A. and Cohen, N. (1973) "Tune evolution as an indicator of traditional musical norms." *J. American Folklore 86*, 339: 37–47.

Deutsch, D. (1972) "Octave generalization and tune recognition." *Perception and Psychophysics 11*(6): 411–412.

Dowling, W. J. (1978) "Scale and contour: Two components of a theory of memory for melodies." *Psychological Review 85*(4) 341–354.

Galil, Z. and Park, K. (1990) "An improved algorithm for approximate string matching." *SIAM J. Comput. 19* (6): 989–999.

Ghias, A., Logan, J., Chamberlin, D. and Smith, B.C. (1995) "Query by humming." *Proc ACM Multimedia 95.* San Francisco, November.

Gold, B. and Rabiner, L. (1969) "Parallel processing techniques for estimating pitch periods of speech in the time domain." *J. Acoust. Soc. Am. 46*(2) 442-448.

Goodrum, C. A. and Dalrymple, H. W. (1982) *Guide to the Library of Congress.* Library of Congress, Washington, D. C.

Hess, W. (1983) *Pitch Determination of Speech Signals.* Springer-Verlag, New York.

Hawley, M. (1990) "The personal orchestra." *Computing Systems 3*(2): 289–329.

Loeb, S. (1992) "Architecting personalized delivery of multimedia information." *Commun. ACM 35*(12), 39–50; December.

Lowrance, R. and Wagner, R.A (1975) "An extension of the string-to-string correction problem." *J ACM 22*(2): 177–183; April.

McNab, R.J., Smith, L.A. and Witten, I.H. (1996) "Signal processing for melody transcription." *Proc. 19th Australasian Computer Science Conf.,* 301–307, Melbourne, January.

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.* Spencer Brown, Cambridge, 1975.

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

Sedgewick, R. (1988) *Algorithms*. Addison-Wesley, Reading, Massachusetts.

Selfridge-Field, E. (1994) “Optical recognition of music notation: a survey of current work.” *Computing in Musicology, 9*: 109–145.

Sloboda, J. (1982) “Music performance.” in *The Psychology of Music*, edited by D. Deutsch, Academic Press, pp. 479–496.

Steiglitz, K., Parks, T.W., and Kaiser, J.F. (1992) “METEOR: A constraint-based FIR filter design program.” *IEEE Trans. Signal Proc., 40*(8), 1901–1909.

Sundberg, J. and B. Lindblom (1976) “Generative theories in language and music descriptions.” *Cognition 4*: 99–122.

Wagner, R.A. and Fischer, M.J. (1974) “The string-to-string correction problem.” *J ACM 21*(1): 168–173; January.

Waibel, A. and Yegnanarayana, B. (1983) “Comparative study of nonlinear warping techniques in isolated word speech recognition systems.” *IEEE Trans Acoustics, Speech, and Signal Proc. 31*(6): 1582–1586

Wu, S. and Manber, U. (1992) “Fast text searching allowing errors.” *Commun. ACM 35*(10), 83–91; October.

List of figures and tables

Figure 1. Acoustic waveform of *ah*

Figure 2. Amplitude segmentation

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

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 5. Number of collisions for different lengths of input sequence when matching embedded patterns. Lines correspond, from left to right, to those in Figure 3.

Figure 6. Display from the tune retrieval system

Table 1. Results of performance experiment

Table 2. Number of songs retrieved using Figure 5 input pattern

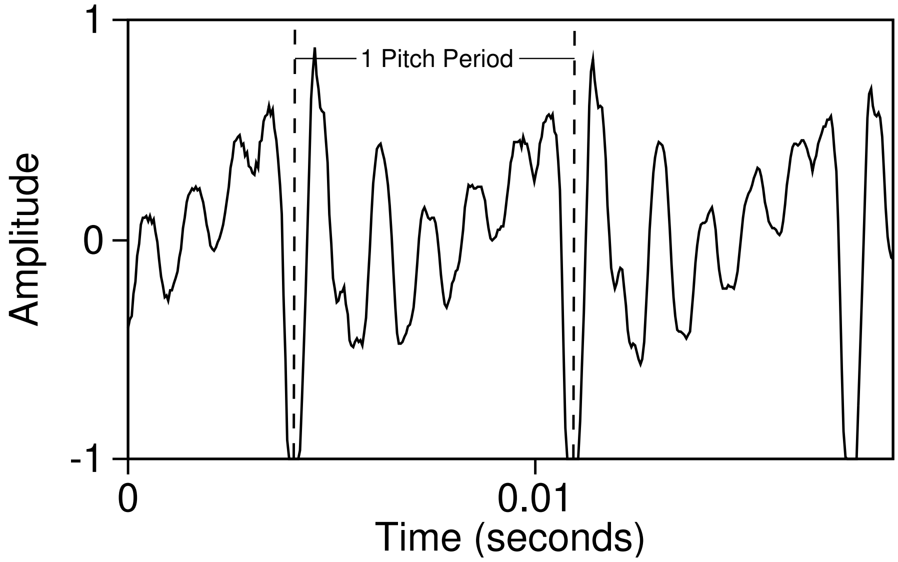

Figure 1. Acoustic waveform of *ah*

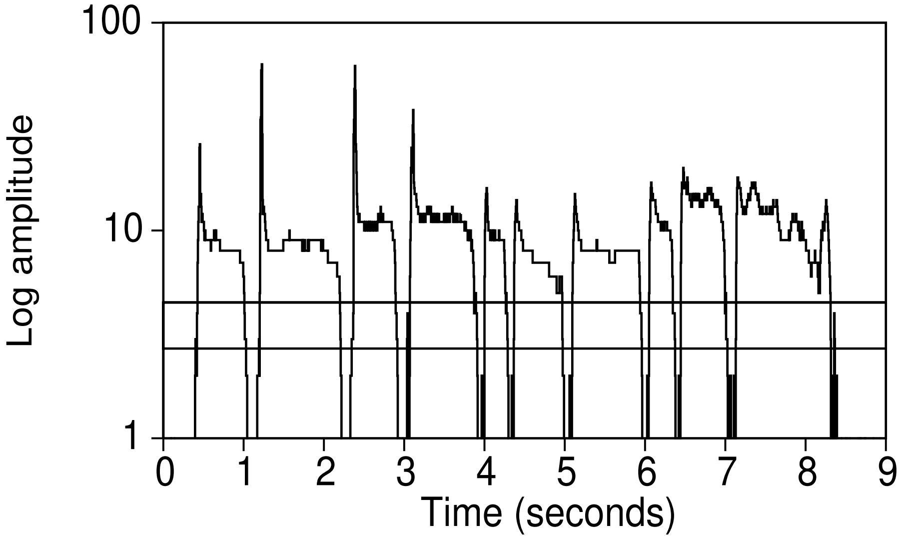

Figure 2. Amplitude segmentation

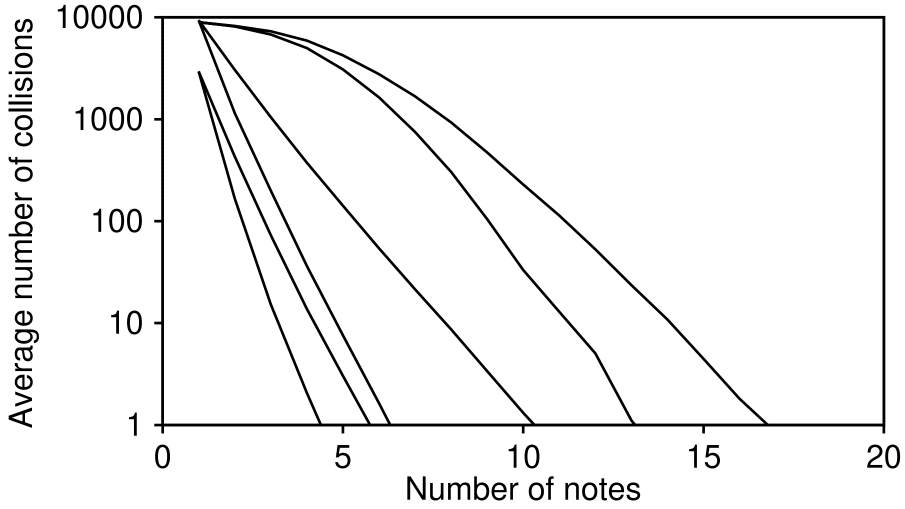

Figure 3. Number of collisions for different lengths of input sequence when matching start of song. From left to right:

- exact interval and rhythm
- exact contour and rhythm
- exact interval
- exact contour
- approximate interval and rhythm
- approximate contour and rhythm

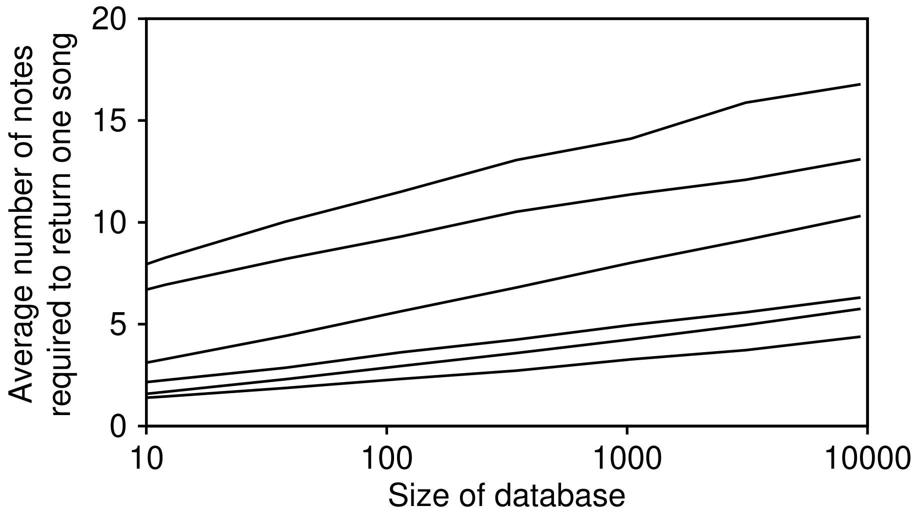

Figure 4. Number of notes for unique tune retrieval in databases of different sizes. Lines correspond, from bottom to top, to the matching regimes listed in Figure 3.

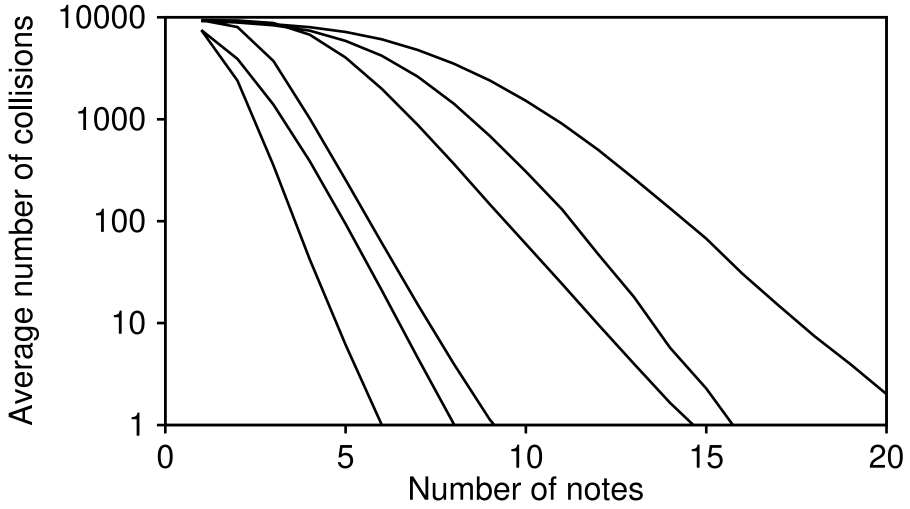

Figure 5. Number of collisions for different lengths of input sequence when matching embedded patterns. Lines correspond, from left to right, to those in Figure 3.

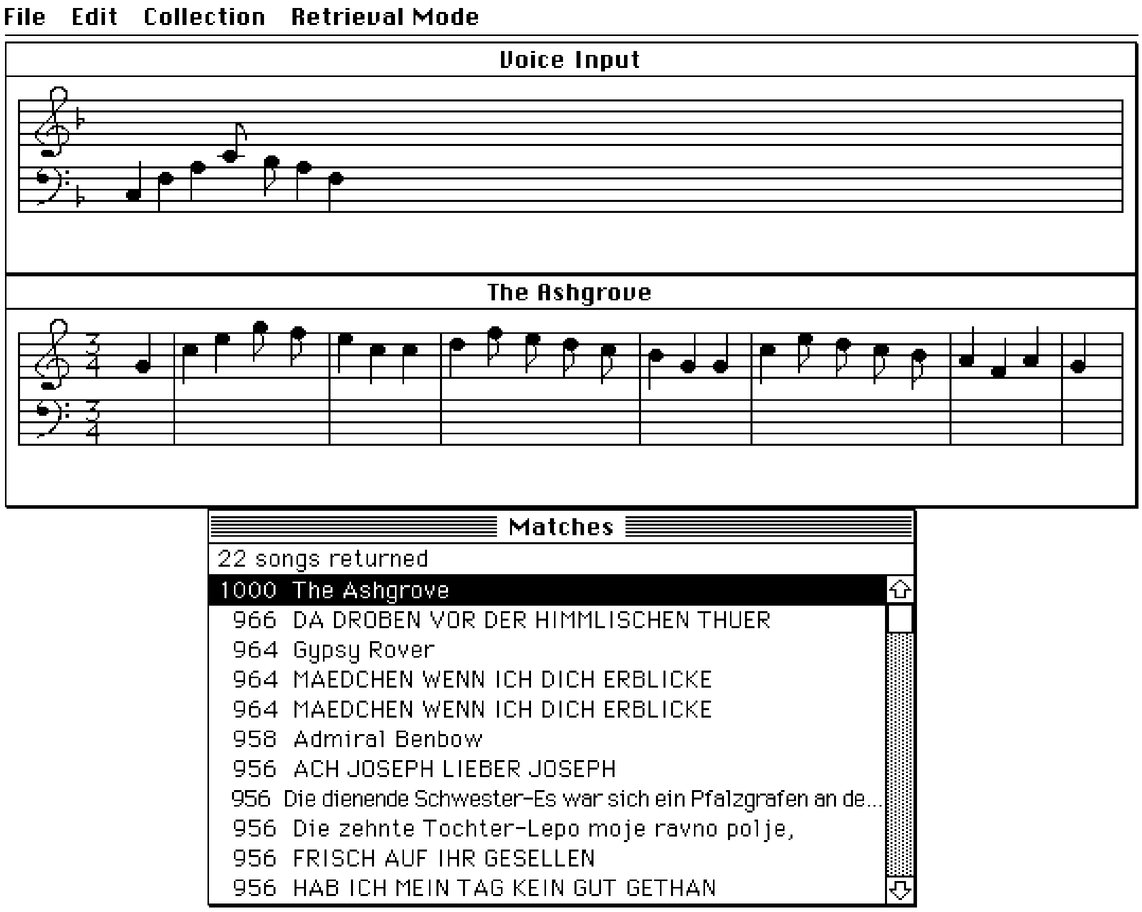

Figure 6. Display from the tune retrieval system

Song	Number of singers	Ending "in key"	No. starting at beginning
Bridge Over Troubled Water	7	5	2
Hound Dog	8	8	8
King of the Road	8	3	8
Memory	10	8	10
Moon River	10	4	10
Pokare kare ana	10	7	10
Puff, The Magic Dragon	10	8	10
Summertime	9	4	9
Yankee Doodle	10	9	10
Yesterday	9	3	9

Table 1. Results of performance experiment

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

Table 2. Number of songs retrieved using figure 6 input pattern