Working Paper Series
ISSN 1170-487X

A Sight-Singing Tutor

**by Lloyd A. Smith and
Rodger J. McNab**

Working Paper 97/8

March 1997

Department of Computer Science
The University of Waikato
Private Bag 3105
Hamilton, New Zealand

A Sight-Singing Tutor

Lloyd A. Smith and Rodger J. McNab
{las,rjmcnab}@cs.waikato.ac.nz

Department of Computer Science
University of Waikato
Private Bag 3105
Hamilton, New Zealand

Abstract

This paper describes a computer program designed to aid its users in learning to sight-sing. Sight-singing—the ability to sing music from a score without prior study—is an important skill for musicians and holds a central place in most university music curricula. Its importance to vocalists is obvious; it is also an important skill for instrumentalists and conductors because it develops the aural imagination necessary to judge how the music should sound, when played (Benward and Carr 1991). Furthermore, it is an important skill for amateur musicians, who can save a great deal of rehearsal time through an ability to sing music at sight.

Background

The principles underlying the most widely used methods of teaching sight-singing were set out by the eleventh century music theorist Guido of Arrezo (ca. 1030), inventor of the musical staff. The basic requirement is that the student form a mental model which relates notes on the staff to one another; most common is to use the musical scale as the model, either relating all notes to C (*fixed do*) or to the tonal center (*movable do*) (Roe 1970). A third method is to attempt to internalize the sounds of all intervals individually, although, as McNaught (1899) points out, key context can make intervals difficult to use. Whatever method is used, a great deal of practice, with correction from a listener, is necessary in developing sight-singing skill. The nature of this loop—continual drill with feedback—suggests that computer assisted instruction would be useful in helping people learn to sing music at sight.

In principle, any system that tracks the fundamental frequency of acoustic input can be used to provide feedback to sight-singing students. There have been several systems

developed in recent years, however, which are targeted specifically at providing feedback for singers. Welch, Howard and Rush (1989) describe a system used to train primary school students. Their system, SINGAD (Singing Assessment and Development), consists of a peak picking device linked to a BBC microcomputer. SINGAD displays each frequency estimate, in real time, as a dot on the computer's screen. The system places targets on the screen that the child must hit. By observing how the dots move as they sing, children learn that their voices can be consciously controlled. Welch and his colleagues showed that their system was more effective than traditional primary school training involving singing along with the teacher, and that improvement was sustained six months after the training.

A somewhat similar system was described by Kuhn (1990). He developed a pitch tracker that was used to provide feedback to singers by showing the progression of pitch as a thick horizontal line on a musical staff. In a separate mode, Kuhn's system produced a musical transcription of the user's singing, requiring the user to tap the space bar to signal note boundaries.

Lorek and Pembrook (1989) describe a system specifically targeted at sight-singing tuition. Their program displayed a test melody, then evaluated a user's attempt to sing that melody. The user was asked to sing using the syllable *ta* in order to separate notes; pitch was tracked by a Roland VP-70 Voice Processor, and some provision was made to adjust to the user's tuning and tempo changes. No feedback was presented until after the user finished singing. Neither Kuhn nor Lorek and Pembrook report their sources for test melodies.

There are several commercial systems designed to teach sight-singing. For the most part, these systems assess performance by accumulating the amount of time the user is singing the correct pitch according to a strict tempo.

Our system takes a different approach from those described above, transcribing the user's sung input separately, then matching it against the test melody. In this way, the user sees not only the intended melody, but also the melody that was actually sung, in common music notation. Test melodies are drawn from a database of 9500 folk tunes, and

evaluation is carried out using an algorithm designed for comparing melodic sequences (Mongeau and Sankoff 1990), thus allowing for inserted or deleted notes, and finding the best alignment of the transcription with the test melody.

Figure 1 is a screen display illustrating the major features of the sight-singing tutor. The music window displays the test melody (on top) and the transcription of the user's singing; the lines connecting the two melodies show the way in which the system matched the notes. Other windows enable the user to set the tempo, to hear the starting note, to record, and to choose test melodies and phrases from the database. Each of these functions is described in more detail below.

Melody Transcription

Our design philosophy is that sight-singing tuition is one application of a general melody transcription program. The focus of this paper is on the sight-singing application. This section briefly describes the melody transcription module; a more complete description may be found in McNab and Smith (1996).

Preliminary Processing

The sight-singing tutor runs on a Power Macintosh and uses the built-in sound I/O of that system. Acoustic input is sampled at 22 kHz and quantized to an eight bit linear representation. In order to remove harmonics that might confuse the pitch tracker, the signal is then passed through a digital low pass filter with a cutoff frequency of 1000 Hz, stopband attenuation of 14 dB and passband ripple of 2 dB. This filter is implemented as a linear phase finite impulse response (FIR) filter having nine coefficients. The filtered signal is used for all further processing.

Pitch Tracking and Note Segmentation

The system is designed to accept input ranging from F2 (87 Hz), just below the bass staff, to G5 (784 Hz), just above the treble staff. While higher and lower pitches are possible, particularly for trained singers, we are not currently considering applications likely to make use of notes outside this range; certainly students should not be expected to challenge their singing techniques when practicing sight-singing.

We are primarily interested in applications of melody transcription, rather than in performing research into frequency identification. Consequently, we chose to use the Gold-Rabiner algorithm (Gold and Rabiner 1969) to track the singer's pitch. We chose the Gold-Rabiner algorithm because it is well known and well documented and is robust when the shape of the waveform is not distorted (Hess, 1983). The algorithm is implemented as described by Gold and Rabiner (1969), although, because the algorithm was designed for speech, minor modifications were necessary to enable the system to track the higher frequencies of sung input. Frequency estimates, returned every 20 milliseconds (ms), have a resolution of ±4 cents, which approximates human frequency resolution above 1000 Hz (human frequency resolution is less acute below 1000 Hz) (Backus 1969).

The system segments notes based on RMS power, calculated over 10 ms frames. In order to separate notes, we ask the user to sing using the syllable *da.* Adaptive thresholds are then used to determine note boundaries.

Pitch/Rhythm Labeling

The melody transcription system represents all notes in terms of their distance, in cents, above MIDI note 0 (8.176 Hz). Representing notes in this way makes it easy to assign musical pitch labels; on the equal tempered scale, semitones fall at intervals of 100 cents, so C4, or middle C, is 6000 cents, while A4 is 6900 cents. This scheme accommodates alternate tunings, such as Pythagorean or just, by simply changing the relationship between cents value and musical pitch label; it can also readily represent nonWestern or experimental musical scales.

A further convenience of the relative-cents representation is that it supports a simple way of adapting to the user's own tuning. In some applications, it is appropriate for the system to begin by assuming the user is singing to the equal tempered scale, but then to adjust the scale during transcription. This is easily done by using a constantly changing offset. For example, if a user sings three notes, 5990, 5770 and 5540 cents above MIDI note 0, the first is labeled C4, corresponding to 6000 cents above MIDI 0, and the offset is set to 10. The second note is labeled Bb3, corresponding to 5800 cents above MIDI 0, and the offset is set to 30. The third note is labeled Ab3, corresponding to 5600 cents above

MIDI 0, even though it is closer, on the equal tempered scale, to G3. For applications in which fixed tuning is appropriate, the offset is fixed at 0. The default in the sight-singing tutor is fixed tuning, but the user can, through the Preferences dialog, direct the system to use adaptive tuning.

Determining intended rhythms from performed note durations is a difficult problem that is receiving a great deal of attention from music researchers. Rosenthal (1992), Widmer (1995) and Berndtsson (1996) propose various approaches to solving this problem. While we hope to be guided by such research in developing future versions of our transcription system, the system currently takes the expedient route of quantizing each note to the nearest allowable rhythm, based on the tempo, both of which are set by the user.

Displaying the Music

There are a number of problems encountered by a general purpose music display system (Blostein and Haken 1991). The method used by our sight-singing tutor is simple, restricting symbols to treble and bass clefs, staffs, barlines, ledger lines, ties, sharps, flats, naturals, rests, dots, stems, and standard note heads. All these symbols are stored in the Macintosh PICT format, and melodies are displayed in common music notation on a grand staff. Staffs are not broken; if the melody is too long to fit on a single staff within a window it simply extends beyond the right edge of the window. The user can use the mouse to scroll any part of the melody into the window.

Melodies from a database, as, for example, the test melody in Figure 1, are displayed using time signatures and bar lines. Transcribed melodies, however, are displayed without bar lines or time signatures. The reason for this is that the system does not attempt to infer time signatures for acoustic input. Furthermore, even if the correct time signature is known, unless the rhythms sung by the user are perfectly sung, and perfectly transcribed, the display splits a number of notes between bars, making the melody very difficult to read and obscuring, rather than clarifying, downbeats.

Key signatures, for transcribed melodies, depend on the application. If it is clear in what key the user is (or should be) attempting to sing, the melody is displayed in that key. Otherwise, the melody is displayed with the shortest key signature (in number of sharps or

flats) that accurately captures all sung intervals while requiring the fewest accidentals. For example, if a user sings the first six notes of *Three Blind Mice* in the key of F, the notes are A–G–F–A–G–F, and the melody is displayed in C. If the user goes on to add the next four notes, however, C–Bb–Bb–A, then the melody is displayed in the key of F.

The system can display whole notes, half notes, quarter notes, eighth notes, sixteenth notes, and the corresponding rests; any of these may be dotted, but not double dotted. Note heads are all the same size, but notes vary slightly in size because of flags. The space allocated to a note is calculated, in number of pixels, by adding the width of the note to a value determined by multiplying the note's duration by a constant. Longer notes, therefore, receive more space than shorter notes, but there is no attempt to make all bars the same width—in other words, half notes do not receive exactly twice as much space as quarter notes, and so forth.

Choosing a Test Melody

Test melodies may be drawn from three databases: the Essen database (Shaffrath 1992), comprised of more than 8000 folk tunes, primarily of German and Eastern European origin, the Digital Tradition database (Greenhaus 1994) of 1700 folk tunes, most of North American origin, and a database of 100 chorale melodies. Together, the three databases hold a total of 9454 unique melodies; all duplicates have been removed.

Searching for Test Melodies

Having a large number of available tunes is useful only if the user can find songs with particular musical features. If a singer is having trouble singing minor thirds, for example, he or she should be able to search the databases for songs containing a large number of minor thirds. This feature is implemented in the sight-singing tutor using a fast string matching algorithm (Baeza-Yates and Gonnet 1992).

Any combination of databases may be active at a given time. To search the active databases, the user first records a phrase containing the search pattern, then selects the salient part of the phrase by clicking and dragging the mouse over the notes. Selecting Find Selection, from the File menu, displays a dialog similar to Figure 2. The user selects whether to use rhythm alone, rhythm and pitch, or pitch alone as the search criteria. A

search involving pitch can be carried out using either intervals or absolute pitches. The user then sets the minimum number of hits on the pattern necessary for a song to be returned by the search. Selecting Count Matches displays the number of tunes returned, while Find First closes the dialog box and displays the first returned song in the music window. Find Next, from the File menu, then displays the next matching tune. Using this mechanism, pertinent examples can be found quickly and easily. Figure 2 shows the result of searching for minor thirds in the Essen and Digital Tradition databases; 27 tunes were returned that contained 25 or more minor thirds.

Selecting Songs and Phrases

When a database is opened, a window similar to that shown in Figure 3 appears. The top half of the window displays a scrollable list of all the songs in the active database(s); the bottom half contains a list of the phrases making up the selected tune. Only one tune may be selected at a given time, but any number of contiguous phrases may be selected, giving the user control over the length of the test melody. Selected phrases appear in the music window. Phrase selection relies on phrase identification information in the database. This information is contained in the Essen and chorale databases, but not in the Digital Tradition database; in other words, tunes from the Digital Tradition are listed as having only one phrase, which encompasses the entire tune.

The user can transpose the tune in order to put it into a comfortable range, or to practice reading music in different keys. When a tune is transposed, the music display changes in the music window to reflect the new key.

Recording

After choosing a test melody, the user selects the tempo and starts recording. The tempo is set in the Tempo Box, a floating window that also allows the user to listen to a metronome click at the chosen tempo. The starting pitch can be played from the Record Box, another floating window. To record, the Record button is held down by clicking and holding the mouse button. While recording, the program displays the percentage of the record buffer which is filled with samples. Recording stops when the buffer is full or when the Record

button is released. The user can set the maximum recording time to values ranging from 5 seconds to 2 minutes.

Scrolling While Recording

If the musical score is too long to fit in the window, it must scroll to the left during recording. Determining the speed of scrolling is difficult because it depends on the tempo and the notes being sung—eighth notes must scroll considerably faster than whole notes. Furthermore, because the system captures the entire recording before transcribing and matching, there is no way to know which note the user is singing at a given time. In fact, even if transcription were carried out during recording, it would still be difficult to identify which note the user is singing because it would require real time tracking of tempo changes—while also allowing for the possibility that the user is simply singing incorrect pitches and/or rhythms. Several approaches to scrolling were attempted before a satisfactory solution was found; all approaches assumed that the user sings at the set tempo. First, some terms will be defined.

The *target note* is the note in the test melody which the user should be singing if he or she starts singing as soon as Record is pressed, has no restarts and keeps strictly to the specified tempo. The *target position* is the position within the test melody where the eye would be if it were travelling at a constant speed between the target note and the following note. Positions are expressed either as a percentage of the window width, with 0% being the left edge and 100% the right edge, or as a number of pixels from the start of the melody.

The *note speed* is the speed at which a pointer would move if it travelled from the target note to the next note over the duration of the target note. Speed is expressed in pixels per beat, which is independent of the tempo; it is calculated by dividing the distance between the notes, in pixels, by the length of the target note, in beats. The *minimum note speed of a melody* is the minimum note speed for any note within the melody. The *maximum note speed of a melody* is the maximum note speed for any note within the melody.

The *scrolling position* is the position of the left edge of the window within the melody, expressed in pixels. The *scrolling speed* is the speed that the scrolling position is moving, in pixels per beat.

Three requirements were identified from initial scrolling attempts. These three requirements are, first, that scrolling should occur as often as possible. The less often scrolling occurs the jerkier the scrolling will be. Jerky scrolling is hard to follow; if the scrolling is by a large amount the user will lose his or her place in the music.

Second, there should be no sudden changes in speed. Sudden changes in speed are distracting. The user is unable to anticipate such changes, thus forcing him or her to concentrate more on the scrolling and less on singing.

Third, the target position should be kept close to the center of the window. The user will not sing exactly to the correct tempo. Keeping the target position close to the center of the window allows for the widest range of tempo variation.

The method described here meets these requirements. It is based on the scrolling speed limits illustrated in Figure 4. S1 is defined to be the minimum note speed, and S2 the maximum note speed, of the test melody. The goal is to keep the scrolling speed between the minimum and maximum speeds shown in the figure. These limits are based on S1 and S2 and the current point of the target position. If, at any time, the scrolling speed is less than the minimum speed for the target position then the scrolling speed is increased to the minimum speed for that target position. If the scrolling speed is greater than the maximum speed allowed for the target position then the scrolling speed is decreased to the maximum speed for that target position. Scrolling takes place as often as possible, which depends on how fast new notes can be drawn. Because the minimum and maximum speeds increase gradually as the target position moves left to right across the window, there will be no sudden changes in speed (except in extreme cases where the window width is very small). Because the maximum speed within the first quarter of the window is always less than S1, once the music starts scrolling, it will not stop until the end of the music is reached. Because the minimum speed within the last quarter of the window is S2, the note the user

should be singing will never reach the end of the window. Most of the time will be spent between a quarter and three quarters of the window width.

Bach's chorale number 71 is a good melody for testing the scrolling algorithm; the melody has a total width of 1536 pixels and a duration of 52 beats. The chorale is shown in Figure 5; it is broken into three segments for printing, but the sight-singing tutor displays it on one long staff. For this melody, S1 is 14.5 and S2 is 64 pixels per beat. As displayed by the system, the bars differ greatly in length. Bar 8 is almost a third the size of bar 1, meaning that with a small window it must scroll at one third the pace of bar 1. The melody starts with three bars which scroll quickly, a slow bar (bar 4), and then another three fast bars. Bars 8 and 9 are slow; they are followed by two medium speed bars, one fast bar and a final medium speed bar.

This melody was tested using three different window sizes—a large window 1000 pixels wide, a window 500 pixels wide, and a small window 250 pixels wide. All tests used a tempo of 120 beats per minute.

The first test, illustrated by Figure 6, uses the large window. This is the ideal case, with the target position staying near the middle of the window (the target position does not start at the left edge of the window—0%—because of the space taken by the clef symbols, key signature and time signature). Testing with a large window on other melodies, and at other tempos, gave similar results. In fact, because the scrolling speed is based on pixels per beat, changing the tempo does not affect the shape of the curves for a given melody and window size—it simply changes the elapsed time represented by the horizontal axis in Figure 6. In other words, changing the tempo, for a given melody and window width, compresses the curves into less time or stretches them over more time, but the curves are the same shape.

The second test, illustrated by Figure 7, uses a smaller window width. This example is only slightly less ideal, with the target position reaching 60% of the window before being pulled back to the left. The algorithm over-corrected, but not seriously, with the target position reaching about 30% before moving back to the right near the end of the melody.

The speed stayed constant for much of the recording and there were no large jumps in speed.

The third test, illustrated by Figure 8, uses a small window. Even so, the music is very readable, with only five major changes in speed over the 52 beats. The changes in speed follow the changes in the music. The scrolling starts fast, decreases in speed for bar 4 (time = 7 seconds), increases for bars 5 and 6, decreases again for bars 8 to the first half of bar 12, and increases again for the second half of bar 12.

Transcribing the Recording

As soon as the Record button is released, the system starts transcribing the acoustic input, with transcription taking 2.8% of the duration of the recording on a Power Macintosh 8500 with 120 MHz clock. From Preferences, on the File menu, the user can select the minimum notes and rests which the system will notate, as well as whether to use fixed or adaptive tuning. Minimum note lengths include sixteenth, eighth, quarter, and half note. Minimum rest lengths include sixteenth, eighth, quarter, and half rest, or no rests at all; if no rests is selected, the system adds embedded silence to the duration of the preceding note.

The transcription is displayed in the music window along with the test melody (Figure 1). The user can record and transcribe without having first displayed a test melody; if a test melody has been selected, the key signature of the transcription is the same as that of the melody—otherwise, the key is set so as to minimise the number of sharps or flats. For reasons discussed above, the transcribed melody is displayed without bar lines.

Once the sound has been transcribed, the user can play either the recorded sound or a synthesized melody based on the transcription.

Assessing the User's Performance

The system scores the transcribed melody against the test melody using a dynamic programming algorithm designed to compare musical sequences (Mongeau and Sankoff 1990). Dynamic programming finds the optimal alignment between two strings, allowing for insertions, deletions and replacements; it has been used extensively in pattern matching for speech recognition (Rabiner and Juang 1993), and in biological sequence comparison (Collins and Coulson 1987). In addition to insertions, deletions and replacements,

Mongeau and Sankoff allow fragmentations and consolidations, thus allowing a half note in one string to fragment into two quarter notes in the other, for example, or, conversely, for two quarter notes to consolidate into a half note. The algorithm returns the alignment, showing how the notes of the two strings match, and a score indicating the closeness of the match. In matching notes against one another, Mongeau and Sankoff use a distance measure based on musical consonance; a fifth is more consonant than a minor second, for example, so matching two notes a fifth apart incurs a lower penalty than matching notes a semitone apart (a unison match incurs no penalty). While such a scheme may be appropriate for matching transpositions and variations of musical themes, it is not appropriate for assessing sight-singing performance. Therefore, in our implementation, we replace the consonance score with the distance of the notes in semitones. The default is to match the transcription against the test melody using absolute pitches and fixed tuning. The user has the option (by setting Preferences) of choosing to match using intervals and/or adaptive tuning; adaptive tuning is a feature of the transcription front end and is described above. If the match is carried out on the basis of intervals, then, if one note is wrong but the following intervals are correct, the notes following the wrong one will be marked correct, even though they are not actually the correct pitches.

The score returned by dynamic programming is a distance; the higher the score, the farther apart—the more different—the two melodies. The sight-singing tutor normalises the score, so that longer melodies are not penalized relative to short melodies, by dividing it by the length of the test melody. Then, in order to make it more intuitive to the user, the score is subtracted from 100 so that, if the user perfectly sings the test melody, the result is 100. Figures 9 and 10 illustrate relatively high and low matching transcriptions of a user's attempt to sing the first four bars of *Three Blind Mice;* as can be seen in Figure 1, as well as in Figures 9 and 10, the score is displayed in the lower left corner of the music window.

Adding Tunes to the Database

Any transcribed tune may be added to the system's collections of melodies by saving it as a database containing one tune. When saving a melody, the user can specify the title, time signature, and whether the key is major or minor.

Because the tune is entered by recording acoustic input, it is necessary for the user to have some way of correcting singing and transcription errors. Double-clicking on a note displays an edit dialog such as the one in Figure 11. The pitch is represented as a MIDI pitch number (C4 is 60), and the rhythm is represented as a duration in number of sixteenths. In the figure, a dotted quarter note, with a duration of six sixteenths, has been selected; the pitch is correct, but the duration will be changed to eight sixteenths, representing a half note. The user is also able to correct notes in this manner before searching the tune databases. Editing a note does not change the score for the transcription's match against a test melody.

Conclusion

This paper describes a sight-singing tutor based on a melody transcription front end. The system displays a test melody, then accepts acoustic input which is transcribed and matched against the test melody; the transcription is displayed below the test melody, with connecting lines showing how the system aligned the two melodies for the match. The score returned by the match is normalized by the length of the test melody and subtracted from 100, then displayed in the lower left corner of the music window. The user has the option of selecting a match of absolute pitches or of intervals, as well as the option of using fixed or adaptive tuning for transcription.

The system provides three databases, totalling approximately 9500 tunes, from which the user may choose test melodies. These databases can be searched in order to find tunes that exercise particular musical patterns. Transcriptions can be edited and saved in order to add new tunes and exercises to the system's collection of melodies.

The sight-singing tutor could be improved in several ways. One improvement, for example, would be to provide more convenient ways to add melodies. This could be done by reading the file format of a generally available music editor, such as Lime (Haken and Blostein 1993). Furthermore, the user should be able to specify which database a tune is added to, so that exercises that logically belong together could be grouped in a common database.

Perhaps the most useful improvements would be to the melody transcription front end. Automatically inferring the singer's tempo, for example, would make it unnecessary for the user to set a tempo before starting to sing. In addition, real time transcription could improve scrolling by following the music while recording. This would allow the system to keep the target position near the center of the window even when the user varies the tempo. Finally, the system currently requires the user to separate each note using a stop consonant; a segmentation algorithm based on pitch rather than amplitude would enable the user to sing words or solfege syllables. Note boundaries might not be present, then, when notes are repeated, but the approximate string matching algorithm would handle those cases by allowing repeated notes in the test melody to consolidate into longer notes in the transcription. We have experimented with a pitch based segmentation algorithm, but, at present, it is not as reliable as the amplitude based method.

The system is functional, however, and we feel it is useful in its current form. Further development will be guided, in large part, by feedback from users and music educators.

References

Backus, J. 1969. *The Acoustical Foundations of Music.* London: John Murray.

Baeza-Yates, R. A. and G. H. Gonnet. 1992. "A New Approach to Text Searching." *Communications of the ACM* 35(10): 74–82.

Benward, B. and M. A. Carr. (1991) *Sightsinging Complete.* Des Moines, IA: Wm. C. Brown.

Berndtsson, G. 1996. "The KTH Rule System for Singing Synthesis." *Computer Music Journal* 20(1): 76–91.

Blostein, D. and L. Haken. 1991. "Justification of Printed Music." *Communications of the ACM* 34(3): 88–99.

Collins, J. F. and A. F. W. Coulson. 1987. "Molecular Sequence Comparison and Alignment." in *Nucleic Acid and Protein Sequence Analysis.* ed. M. J. Bishop and C. J. Rawlings. Oxford: IRL Press.

Gold, B. and L. Rabiner. 1969 "Parallel Processing Techniques for Estimating Pitch Periods of Speech in the Time Domain." *Journal of the Acoustical Society of America* 46(2): 442–448.

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

Guido of Arezzo ca. 1030. "Epistola de ignoto cantu." reprinted (trans.) in *Source Readings in Music History.* ed. O. Strunk. New York: W. W. Norton & Co. 1950.

Haken, H. and D. Blostein. 1993. "The Tilia Music Representation: Extensibility, Abstraction, and Notation Contexts for the Lime Music Editor." *Computer Music Journal* 17(3): 43–58.

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

Kuhn, W. B. 1990. "A Real-Time Pitch Recognition Algorithm for Music Applications." *Computer Music Journal* 14(3): 60–71.

Lorek, M. J. and R. G. Pembrook. (1989) "Present and Future Applications of a Microcomputer-based Frequency Analysis System." *Psychomusicology* 8(2): 97–109.

McNab, R. J. and L. A. Smith. 1996. "Melody Transcription for Interactive Applications." Working Paper 96/32. Hamilton, New Zealand: Dept. of Computer Science, University of Waikato.

McNaught, W. G. 1899. "The Psychology of Sight-Singing." *Proceedings of the Royal Musical Association* 26: 35–55.

Mongeau, M. and D. Sankoff. 1990. "Comparison of musical sequences." *Computers and the Humanities* 24: 161–175.

Rabiner, L. and B. Juang. 1993. *Fundamentals of Speech Recognition.* Englewood Cliffs, NJ: Prentice-Hall.

Roe, P. 1970. *Choral Music Education.* Englewood Cliffs, NJ: Prentice-Hall.

Rosenthal, D. 1992. "Emulation of human rhythm perception." *Computer Music Journal* 16(1): 64–76.

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

Welch, G. F., D. M. Howard, and C. Rush. 1989. "Real-time visual feedback in the development of vocal pitch accuracy in singing." *Psychology of Music* 17: 146–157.

Widmer, G. 1995. "Modeling the rational basis of musical expression." *Computer Music Journal* 19(2): 76–96.

Figures

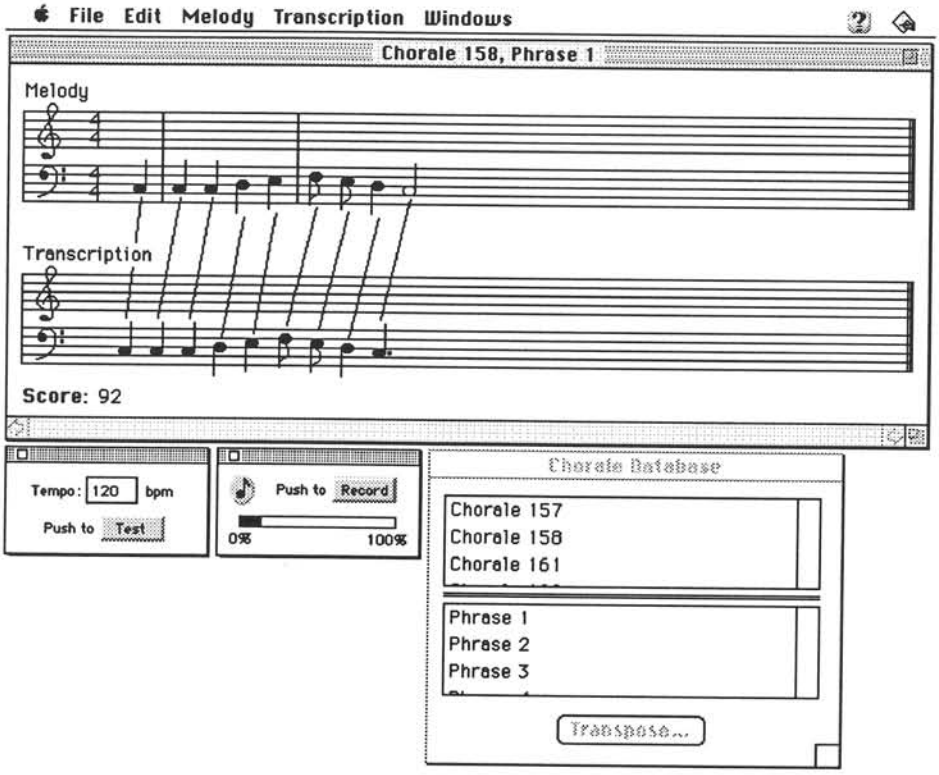

Figure 1. A screen display showing most aspects of the sight-singing tutor.

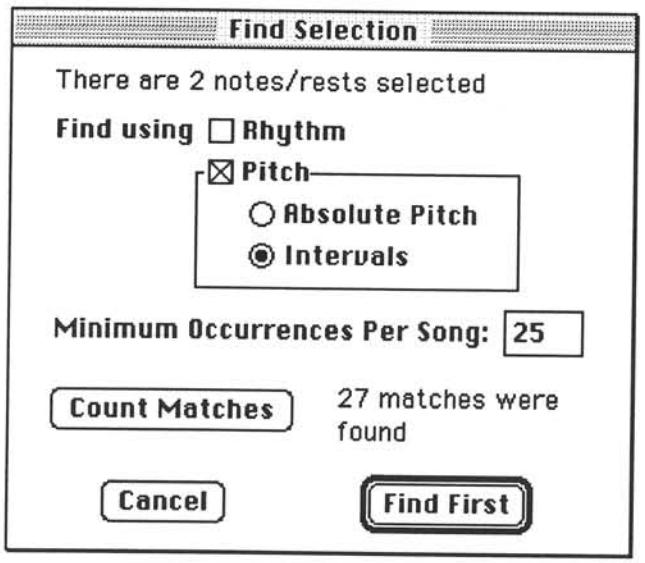

Figure 2. Dialog for searching a database.

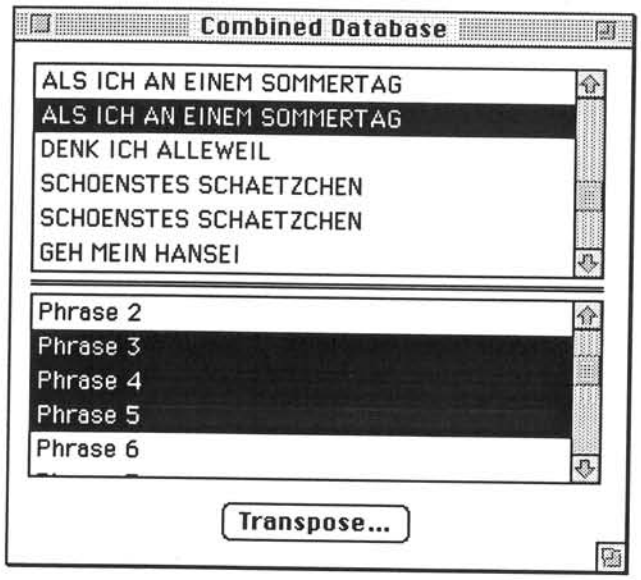

Figure 3. Window for selecting test melodies and phrases.

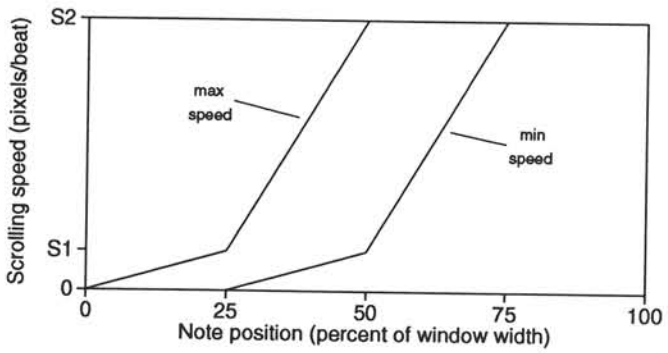

Figure 4. Speed limits for scrolling.

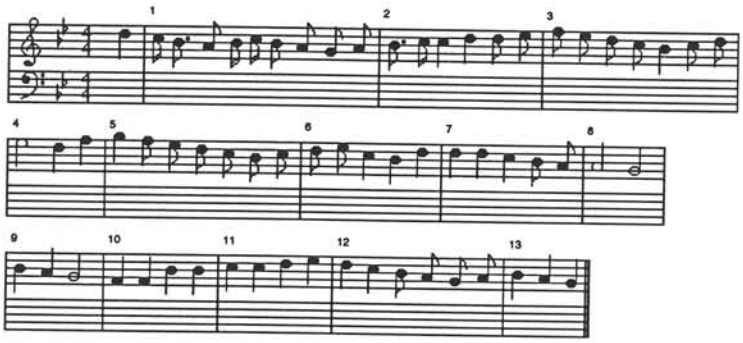

Figure 5. Bach chorale number 71.

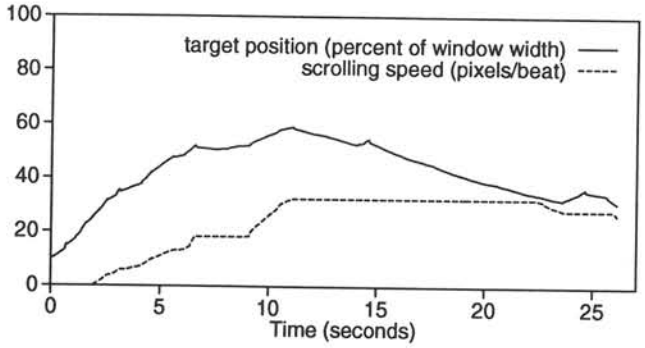

Figure 6. Target position and scrolling speed; window width = 1000 pixels.

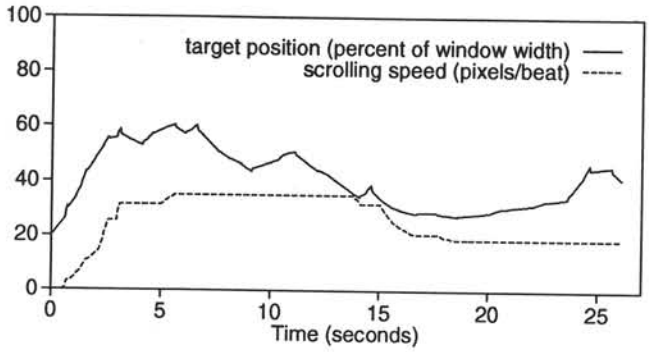

Figure 7. Target position and scrolling speed; window width = 500 pixels.

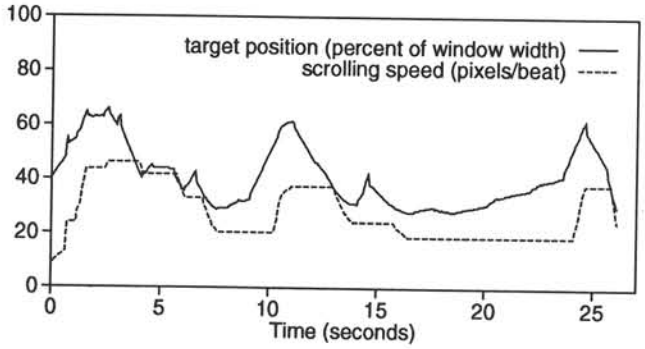

Figure 8. Target position and scrolling speed; window width = 250 pixels.

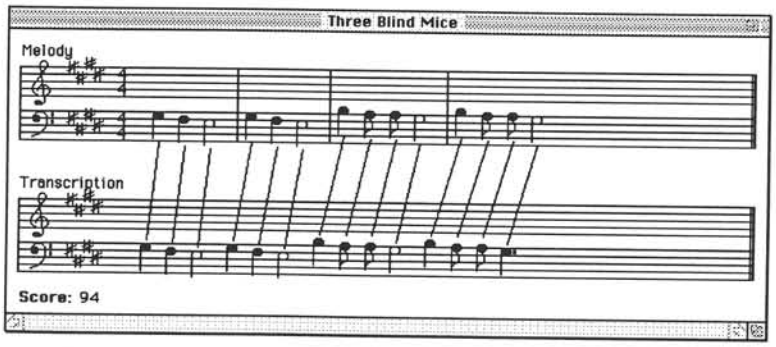

Figure 9. A high scoring match.

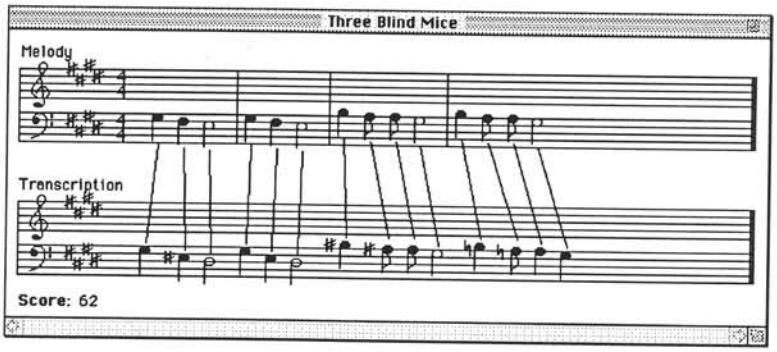

Figure 10. A lower scoring match.

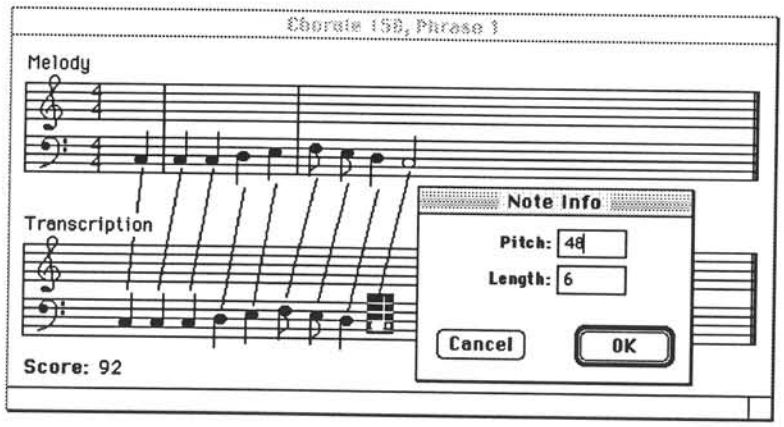

Figure 11. Editing a note's pitch and/or duration.