Signal Processing for Melody Transcription

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

Department of Computer Science, University of Waikato,
Hamilton, New Zealand.

{rjmcnab, las, ihw}@cs.waikato.ac.nz

Abstract

MT is a melody transcription system that accepts acoustic input, typically sung by the user, and displays it in standard music notation. It tracks the pitch of the input and segments the pitch stream into musical notes, which are labelled by their pitches relative to a reference frequency that adapts to the user's tuning. This paper describes the signal processing operations involved, and discusses two applications that have been prototyped: a sightsinging tutor and a scheme for acoustically indexing a melody database.

Keywords Music transcription, pitch tracking, computer assisted instruction, sight reading.

1 Introduction

With advances in digital signal processing, music representation techniques, and computer hardware technology, it is becoming feasible to transcribe melodies automatically from an acoustic waveform to a written representation, using a small personal computer. For example, a person could sing a tune and have it printed in ordinary musical notation. Such a scheme would have novel and interesting applications for professional musicians, music students, and non-musicians alike. Songwriters could compose tunes easily without any need for a MIDI keyboard; students could practice sight-singing with a computer tutor; and ordinary people could identify those elusive melodies by humming a few bars and having the tune's name retrieved from a database.

Although pitch trackers, which identify the fundamental frequency of a waveform and follow its evolution over time, have been around for 30 years or more, only a few projects have undertaken the systems engineering required to create a music transcription system, and they have invariably suffered from serious restrictions. Askenfelt [1] describes the use of a real-time hardware pitch tracker to notate folk songs from tape recordings. People listened to output synthesised from the pitch track and used a music editor to correct errors. However, it is not clear how successful the system was: Askenfelt reports that "the weakest points in the transcription process at present are in pitch detection and assignment of note values." Pitch trackers have been used to transcribe instrumental music, but require the input to have constant pitch—no vibrato or glissando. This restriction rules out vocal sources. For example, Moorer [9] describes a system that transcribes two-part music, and his example inputs were violin and guitar duets. He reported problems in finding the beginnings and ends of notes. Piszczalski and Galler [13] restricted input to recorders and flutes playing at a consistent tempo. These instruments are relatively easy to track because they have strong fundamental frequencies and weak harmonics.

More recently, Kuhn [8] described a system that transcribes singing by displaying the evolution of pitch as a thick horizontal line on a musical staff to show users the notes they are producing. No attempt is made to identify the boundary between one note and the next: the only way to create a musical score is for users to tap the computer's keyboard at the beginning of each note. There are at least two commercial systems that claim to teach sightsinging—but they generate the melodies randomly, which inevitably creates very bad examples for users to work with. More fundamentally, they require the user to keep mercilessly to the tempo set by the machine, and the overall success measure is calculated from the accumulated amount of time that he or she is not singing the correct pitch. Such partial solutions to the melody transcription problem are not very useful in practice.

Considerable work is required to build a useable melody transcription system that works with the singing voice. Pitch trackers suffer from well-known problems of accuracy, particularly at the beginnings and ends of notes and at transitions between frequencies, and very large errors (e.g. octave displacement) are common. Most people are not good singers, which introduces another source of variability that must be addressed for a transcription device to be useful. Furthermore, there are different ways of defining musical intervals from pitch, and it is an open research question as to which kind of scale people adhere to when singing unaccompanied. Determining the boundaries between notes is not easy, particularly for vocal input, although users can help by singing *da* or *ta.* The relationship between pitch acquisition time and the duration of the shortest expected note is an important factor in assigning rhythm.

This paper describes MT, a prototype system for melody transcription that runs on an Apple Macintosh PowerPC using its built-in sound capture capability. Although still in an early stage of

Proceedings of the 19th Australasian Computer Science Conference, Melbourne, Australia, January 31–February 2 1996.

development, MT is able to identify a sung melody, in real time, and transcribe it into Common Music Notation, familiar to most people in European countries as the system in use for several hundred years to notate Western music. Applications include transcribing folk songs or ephemeral musical performances such as jazz improvisations, computer assisted instruction in music, music information retrieval from acoustic input, and even intonation training for the deaf.

The structure of the paper is as follows. Section 2 lays the foundation by discussing background requirements for sound capture and music representation. Section 3 examines the problem of pitch tracking, with a brief review of the well-known Gold-Rabiner algorithm that MT uses, and a more extensive discussion of the post-processing that is required to obtain a meaningful pitch contour. Section 4 considers the question of note segmentation, and introduces two separate methods, one based on amplitude and the other on the pitch contour. Section 5 considers how the notes so identified can be labelled with musical note names. Section 6 describes two applications we have prototyped using melody transcription, and the final section presents conclusions from current work and plans for future development.

2 Preliminaries

Before describing the pitch tracking and note identification processes, let us dispense with some preliminaries regarding sound capture and note representation. The first step in melody transcription is to capture the analog input and convert it to digital form, filtering it to remove unwanted frequencies. The next is to identify its frequency, as described in Section 3. Whereas frequency is a physical attribute of a periodic or quasi-periodic signal, pitch is a perceptual attribute evoked in the auditory system. In general, there is an orderly and well-behaved correspondence between frequency and pitch which is only breached under carefully-controlled conditions in the psychoacoustic laboratory; hence in this paper the terms frequency and pitch are used synonymously. In order to represent the pitches musically, it is necessary to consider how musical scales are defined.

2.1 Sampling and filtering

MT runs on an Apple Macintosh PowerPC 8100, which has built-in sound I/O. The acoustic waveform is filtered at 10 kHz, sampled at 22.05 kHz, and quantised to an 8-bit linear representation.
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 lie beyond the scope of most applications of melody transcription.

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 ninth-order finite impulse response (FIR) filter. The filtered signal is passed to the pitch tracker, which identifies its fundamental frequency.

2.2 The musical scale

A musical scale is a logarithmic organisation 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 above that is soprano high C (C6) at 1046.4 Hz. The octave below middle C (C3) is 130.8 Hz, and below that, at 65.4 Hz, is C2—which has ensured the fortunes of a few extraordinary jingle-singing basses.
Although the octave seems to be a perceptual unit in humans [4], pitch organisation within the octave takes different forms across cultures. In Western music, the primary organisation since the time of Bach has been the equal-tempered scale, which divides the octave into twelve equally spaced *semitones*. The octave interval corresponds to a frequency doubling and semitones are equally spaced in a multiplicative sense, so ascending one semitone multiplies the frequency by the twelfth root of 2, or approximately 1.059.

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. Physicists and psychologists have agreed on the logarithmic unit of *cent*, 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 stabilises at about 4 cents.

While pitch may be perceived categorically in terms of octaves, semitones and cents, frequency is continuous. Assigning a musical pitch to a given frequency involves quantisation. In order to quantise pitches in Western music based on a particular tuning standard (for example, A-440), semitone resolution is sufficient. To accommodate different tuning systems, however—including adapting to users, who inevitably sing slightly sharp or flat—higher resolution is essential. We have designed the system around a pitch resolution of five cents, which is about the limit of its pitch tracking accuracy.

2.3 The MIDI note representation

Since musical units—octaves, cents and so forth—are relative measures, a distance in cents could be calculated between each individual interval sung by the user. It is useful, however, to set a fixed reference point, making for easier development and debugging. 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 sytem 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.

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.

3 Pitch tracking

Pitch determination is a common operation in signal processing. Unfortunately it is difficult, as testified by the hundreds of different pitch tracking algorithms that have been developed [7]. These algorithms may be loosely classified into three types, depending on whether they process the signal in the time domain (sampled waveform), frequency domain (amplitude or phase spectrum) or cepstral domain (second order amplitude spectrum). One of the best known pitch tracking algorithms, and one against which other methods are often compared, is the Gold-Rabiner scheme [6]. This is a time-domain method: it determines frequency by examining the structure of the waveform. Because it is well understood and well documented, we chose to implement it as the pitch determination method.

3.1 Gold-Rabiner algorithm

A sound that has pitch is periodic (or, more accurately, quasi-periodic)—its waveform is made up of repeating segments or *pitch periods*. This observation is the rationale for time-domain pitch trackers, which attempt to find the repeating structure of the waveform. In music and speech, a pitch period is usually characterised by a high-amplitude peak (caused by a puff of air from a vibrating reed or buzzing vocal folds) at the beginning of the pitch period, followed by peaks of diminishing amplitude as the sound dies away. The high-amplitude peak at the beginning of each pitch period is the primary waveform feature used by time-domain pitch trackers.

The Gold-Rabiner algorithm uses six independent pitch estimators, each working on a different measurement obtained from local maxima and minima of the signal. The goal is to take into account both the regularity and "peakedness" of a periodic signal, and to provide a large measure of fault tolerance. The final pitch estimate is chosen on the basis of a voting procedure among the six estimators. When the voting procedure is unable to agree on a pitch estimate, the input is assumed to be aperiodic—silence, or an unvoiced sound such as *s* or *sh*.

The algorithm was designed for speech applications, and performs over a range of input frequencies from 50 Hz to 600 Hz. Our implementation allows the higher frequencies necessary for singing (up to 1000 Hz) by changing the manner in which pitch period lengths are determined. We also made modifications in implementation to speed up pitch tracking.

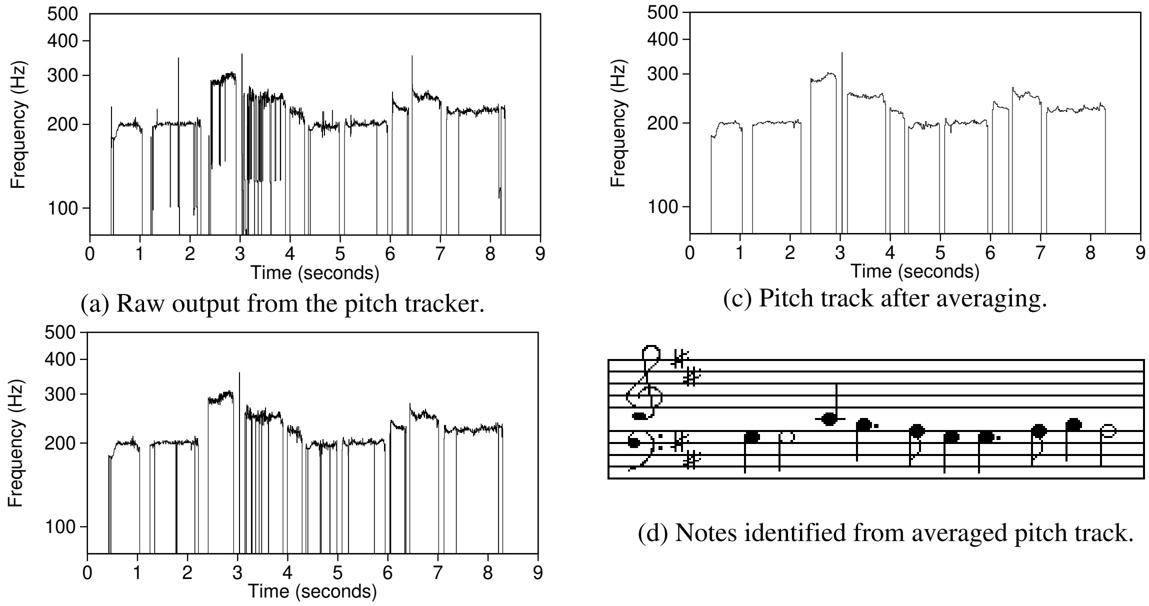
3.2 Postprocessing

No pitch tracker returns perfect output. Figure 1 illustrates the various stages in pitch tracking, from raw output to musical notes. Figure 1a shows the output directly from the Gold-Rabiner algorithm, which consists of a pitch estimate at every sample location. Errors can be characterised as gross or fine [7]. We define gross errors to be estimates greater than 10% (about a whole tone) from the input frequency, while fine errors are less than 10% from the true frequency.

Gross errors generally occur at times when the signal is unstable, at the start or end of pitched sound, for example, or during note transitions. The most common is an octave error, which occurs when the pitch estimate is twice or one half the true frequency; a problem for all pitch trackers [2]. In time-domain pitch trackers, octave errors occur when the pitch tracker locks onto the first overtone (twice the frequency) or misses a pitch period and estimates the period to be twice its true length (half the frequency).

We use a simple "island building" strategy to deal with gross errors. First, areas of stability are found within the sequence of pitch estimates. A stable area consists of two or more consecutive pitch periods with no gross errors in the estimated pitch. The post-processor then extends these areas of stability in both directions, correcting octave errors and labelling the pitch of other gross errors as undefined. Figure 1b shows the pitch track after island building.

Most fine errors are caused by sampling resolution. Because digital algorithms deal with discrete samples, the resolution of the pitch estimate depends on fundamental frequency. Sampling at 22 kHz, a pitch period of 100 Hz is sampled 220 times, whereas a pitch period of 1000 Hz is sampled 22 times. Since the sample chosen as the beginning of a pitch period may be half a sample away (on either side) from the true beginning, the pitch estimate may be off by as much as 5%, or about a semitone, at 1000 Hz. A common way to increase the resolution of time-domain pitch estimates is to interpolate between samples [3, 8]. We chose an alternative approach of averaging pitch estimates over a given time period; the result is a perceptually constant resolution, independent of frequency. Pitch estimates are averaged over 20 ms time frames, giving a resolution of approximately 5 cents (0.29%). An undefined pitch value is assigned the average pitch of the frame that it falls within.



(a) Raw output from the pitch tracker.

(b) Pitch track after island building.

(c) Pitch track after averaging.

(d) Notes identified from averaged pitch track.

Figure 1: Pitch tracking.

Figure 1c illustrates the pitch track after averaging and interpolation of undefined pitch estimates, and Figure 1d shows the corresponding musical notes.

Averaging over time frames is equivalent to the solution inherently employed by frequency domain pitch trackers, where the time/frequency resolution tradeoff is a well known constraint. At a tempo of 120 beats/minute the duration of a semiquaver is 125 ms, and six pitch frame estimates will be calculated.

As a final step before segmentation, the (averaged) frequency of each 20 ms time frame is represented by its distance in cents above MIDI note 0.

4 Note segmentation

Two methods of note segmentation have been developed, one based on amplitude and the other on pitch. Amplitude segmentation is simpler and more straightforward to implement. The user is required to separate each note, which is accomplished by singing *da* or *ta*. Amplitude segmentation has the advantage of distinguishing repeated notes of the same pitch. However, segmenting on pitch is more suitable for real-time implementation and relaxes constraints on the user's singing. In either case, rhythmic values are determined by simply quantising the note duration according to the tempo set by the user. The most appropriate rhythmic unit for quantisation depends on the application and on tempo restrictions. A program intended for coloratura vocal training, for example, might require quantisation to the nearest 64th note in order to capture trills or other rapid musical ornaments. For our current applications—sightsinging tuition and song retrieval—we believe quantisation to the nearest semiquaver is sufficient, and we have designed the system accordingly. This parameter can easily be changed for future applications.

4.1 Segmentation based on amplitude

Amplitude segmentation depends on a drop in amplitude between notes in the input signal. This is most easily accomplished by asking the user to sing a syllable such as *da* or *ta*—the consonant will cause a drop in amplitude of 60 ms or more at each note boundary.

The first step is to obtain the root-mean-squared power of the input signal. This is calculated over 10 ms time frames, and the resulting signal is used to segment notes in the input stream. The simplest way to segment notes is to set a threshold, denoting a note start when the power exceeds it, and a note end when the power drops below it. There are three problems with this simple segmentation procedure. First, an extraneous sound, such as a crackling microphone lead or door slam, may send the power shooting above the threshold for a very short time. Second, a signal may cross the threshold several times as it ramps up to or down from its "steady state" level. Third, a fixed threshold may not suffice for all microphones and recording conditions.

The first problem is solved by weeding out short spikes with a time threshold: if the note is not long enough, ignore it. Given semiquaver rhythm quantisation, we assume that each note lasts for at least 100 ms, a threshold that can be reduced for music with shorter note values or faster tempos. The second problem is dealt with using hysteresis: a high threshold is set for the note start boundary and a lower one for the end boundary. The third problem calls for adaptive thresholds. Having calculated the power over 10 ms frames, an overall power figure is calculated for the entire input signal and the note start and end thresholds are set to 50% and 30% of this value. These values were arrived at through experimentation. Figure 2 illustrates the use of thresholds in segmentation. The lines in the Figure are note begin and end thresholds—the note starts when its rising amplitude crosses the upper threshold, and ends when its falling amplitude crosses the lower threshold.

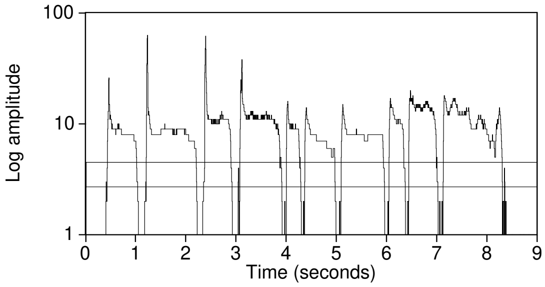
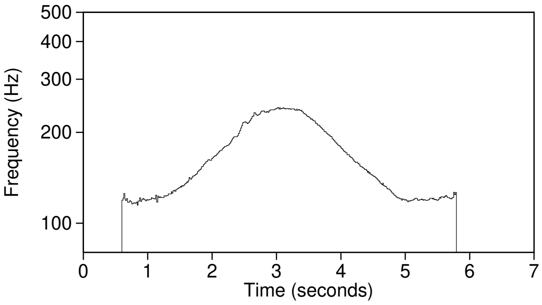


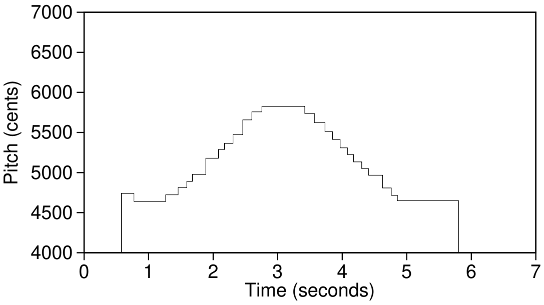
Figure 2: Using thresholds to segment notes from the amplitude signal.

4.2 Segmentation based on pitch

The alternative to amplitude segmentation is to segment notes directly from the postprocessed pitch track by grouping and averaging frames. Frames are first grouped from left to right. A frame whose frequency is within fifty cents of the average of the growing segment is included in the average. Any segment longer than 100 ms is considered a note. For the purpose of determining note durations, notes are extended first by incorporating any contiguous short segments on the right until encountering a change in direction of frequency gradient, unvoiced segments or another note. These newly incorporated segments are considered transitional regions—their frequency estimates are not modified. Notes are then similarly extended on the left. Figure 3 shows the effect of pitch-based segmentation on a sung glissando.



(a) Smoothed pitch track



(b) Segmented pitch track

Figure 3: Pitch-based segmentation on a sung glissando.

5 Musical pitch identification

Labelling the pitch with a musical note name may seem a simple operation, but mapping frequency, which is continuous, onto a musical scale, which is discrete, causes problems because the pitch within a given note may vary over its duration. Seashore [14] reports that professional singers vary frequency in several ways, often starting a long note up to 90 cents flat, but that the average over the note is usually the notated pitch. Notes with a regular vibrato are perceived at the average pitch of the vibrato [15]; similarly, a short slide, or glissando, is perceived as the geometric mean of its extremes [11].

5.1 Identifying note frequencies

Pitch-based segmentation assigns each note the weighted average of its component frequencies. Amplitude segmentation, however, leaves each note as a sequence of individual frames. A single pitch estimate is assigned to these frames using a histogram with bins one semitone wide overlapped at five cent intervals. The range of histograms to be computed for a given note is determined during segmentation. For example, given a note whose frame pitches fall between 4000 and 4750 cents above MIDI note 0, a histogram is calculated with bins at 4000–4100 cents, 4005–4105 cents, 4010–4110 cents, and so forth. For efficiency, a sliding window is used to calculate the histogram. The bin with the highest number of occurrences is chosen as the basis for calculating note frequency. Figure 4 displays a note histogram, with the winning bin indicated by a broken line. Note frequency is the weighted average of frequencies falling within the winning bin.

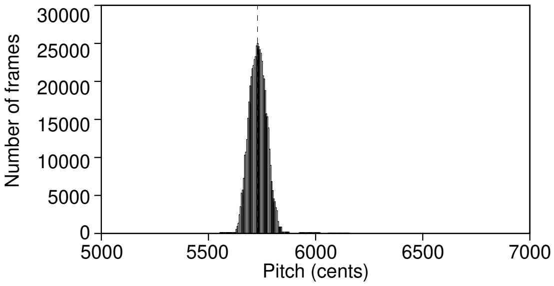


Figure 4: Using a histogram to determine frequency.

5.2 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. For example, the sightsinging tutor is normative, using a fixed reference frequency of A-440, while the melody indexing application is adaptive, adjusting its reference according to the user's singing.

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 labelled 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* will be transcribed. This scheme is not only compassionate in dealing with untrained singers—it also allows the user to sing in the other commonly espoused tunings for Western music, just and Pythagorean.

While constantly changing the reference frequency may sound 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.

6 Applications

Two applications of MT have been prototyped. The first is computer-aided tuition in sightsinging—teaching the skill of singing a melody without prior study. The other is music retrieval from acoustic input. Both applications currently use amplitude-based note segmentation, with the user singing *da* or *ta*.

6.1 Sightsinging tutor

The sightsinging application displays a melody and evaluates the user's attempt to sing it. Melodies are drawn from a database of 100 Bach chorales. First, the system displays a melody on the screen. Users are able to set the tempo and hear the starting note using pull-down menus. Then they sing the melody, using the mouse to start and stop recording. Next the system matches the input against the test melody using a dynamic programming algorithm designed to match discrete musical sequences [10]. Dynamic programming finds the best match between two strings, allowing for individual elements to be inserted or deleted.

Figure 5 shows the result of matching a user's sung input with the test melody. A global score is calculated which takes account of both pitch and rhythm, and returns an alignment showing the best match between the two strings. Notes are penalised for each semitone difference in pitch (by 0.1 units), and for each semiquaver difference in duration (by 0.05 units). The sightsinging tutor accepts melodies sung in any octave. The individual pitch and rhythm scores from the best alignment are accumulated to determine a global distance: a perfect score of zero represents no difference between the input and the test melody.

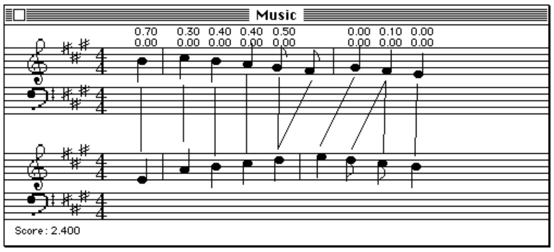


Figure 5: Display from the sight-singing tutor.

6.2 Melody indexing

The indexing application listens to the user sing a few notes of a melody, then returns all melodies that contain that phrase. The current database comprises a set of 1700 North American folk songs.

Given the user's input, it would be easy to match the sung phrase directly against all songs in the database. However, the fact that a melody is equally recognisable whatever key it is played in indicates that the search should be conducted on the basis of pitch ratios, or intervals. Furthermore, a number of experiments have shown that interval *directions*, independent of interval sizes, are an important factor in the recognition of melodies [5]—indeed, Parsons [12] has produced an index of melodies based entirely on the sequence of interval directions, which is called the "melodic contour" or "pitch profile." This suggests the possibility of accessing the database according to contour alone. One cardinal advantage of searching on contour, at least for casual singers, is that it releases them from having to sing accurate intervals.

The prototype melody indexing program allows the user the option of indexing on contour or on musical intervals; at this point, rhythm is not used for indexing. The user starts singing on any note, and the input is notated in the key that yields the fewest accidentals given the notes sung by the user. Because retrieval is based on musical intervals rather than on absolute pitches, the system will return melodies with those intervals regardless of their keys as stored in the database.

Figure 6 shows the melody indexing screen following a search. The names of matching melodies are displayed in a text window, with the first displayed in a melody window. The user may select other melodies from the list for display. In the Figure, the user has selected *Three Blind Mice*.

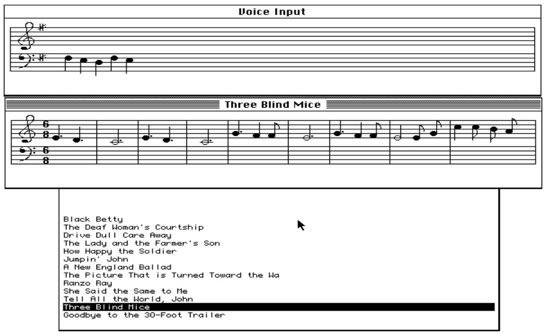


Figure 6: Display from the melody indexing system.

So far, our work on melody indexing has focused on developing and utilising the signal processing front end. In the future we intend to pay more attention to search and indexing functions. For example, we anticipate offering users the option of taking account of rhythm when matching the database. More importantly, we plan to use approximate string matching, as in the sightsinging tutor. This dispenses with the requirement that the user separate each note by singing *da* or *ta*, because it is no longer necessary to distinguish repeated notes. It is especially advantageous for accessing folk songs since a given song has many variants—and even for a database of popular or classical melodies it allows people to retrieve songs they remember imperfectly. Finally, approximate matching will allow ranked retrieval, and melodies that best match the user's input can be presented first. A major issue will be development of approximate string matching algorithms that avoid computational explosion as the size of the database increases. The next addition to the database will be the Essen collection of 10,000 folk songs, mostly of European origin.

7 Conclusions

We have presented MT, a scheme for transcribing melodies from acoustic input, and described two of its applications. At present, the system records the user's input before undertaking any processing. However, processing time is approximately equal to the real time represented by the signal, and the only aspect that requires the signal to be captured before processing is the calculation of thresholds for amplitude segmentation. Pitch-based segmentation overcomes this problem, allowing the signal to be segmented while sound is being captured. Pitch-based segmentation is appropriate for CAI in sightsinging, and we believe, with some modification in search procedures, it will also suit melody information retrieval. As well as allowing real-time processing, pitch-based segmentation relaxes constraints on the user, who may sing anything, including sol-fa syllables or the words to a song.

An anticipated development is the use of approximate string matching for melody indexing. Approximate string matching allows melodies to be ranked, and those that match best can be presented first. In addition, approximate string matching allows retrieval of folk song variants that differ somewhat from the input, and enables the user to retrieve songs based on imperfectly remembered melodies or themes. One issue with the use of approximate string matching is the development of algorithms to keep the search computationally tractable as the size of the database increases.

A standard pitch is not necessarily imposed on users. In adaptive mode the system starts by assuming A-440, but is able to adjust to the user's own tuning. The system accommodates equal tempered scales as well as other intonations of Western music (such as just or Pythagorean). It may be used with music of non-Western cultures by modifying the way in which frequencies are tied to pitch labels, although non-Western music may also require modification of music display routines.

Acknowledgments

This work was supported by a research grant from the University of Waikato.

References

[1] A. Askenfelt. Automatic notation of played music: the Visa project. In *International Association of Music Libraries Conference,* pages 109–121, Lisbon, Portugal, 1978.

[2] J. Brown. Musical fundamental frequency tracking using a pattern recognition method. *Journal of the Acoustical Society of America,* Volume 92, Number 3, pages 1394–1402, 1992.

[3] J. Brown and B. Zhang. Musical frequency tracking using the methods of conventional and 'narrowed' autocorrelation. *Journal of the Acoustical Society of America,* Volume 89, Number 5, pages 2346–2354, 1991.

[4] D. Deutsch. Octave generalization and tune recognition. *Perception and Psychophysics,* Volume 11, Number 6, pages 411–412, 1972.

[5] W. J. Dowling. Scale and contour: Two components of a theory of memory for melodies. *Psychological Review,* Volume 85, Number 4, pages 341–354, 1978.

[6] B. Gold and L. Rabiner. Parallel processing techniques for estimating pitch periods of speech in the time domain. *Journal of the Acoustical Society of America,* Volume 46, Number 2, pages 442–448, 1969.

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

[8] W. B. Kuhn. A real-time pitch recognition algorithm for music applications. *Computer Music Journal,* Volume 14, Number 3, pages 60–71, 1990.

[9] J. A. Moorer. On the transcription of musical sound by computer. *Computer Music Journal*, pages 32–38, November, 1977.

[10] M. Mongeau and D. Sankoff. Comparison of musical sequences." *Computers and the Humanities,* Volume 24, pages 161–175, 1990.

[11] I. V. Nabelek, A. K. Nabelek and I. J. Hirsh. Pitch of tone bursts of changing frequency. *Journal of the Acoustical Society of America,* Volume 48, pages 536–553, 1970.

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

[13] M. Piszczalski and B. A. Galler. Automatic music transcription. *Computer Music Journal*, pages 24–31, November, 1977.

[14] C. E. Seashore. *Psychology of Music.* McGraw-Hill, New York, 1938.

[15] J. Sundberg. Perception of singing. In *The Psychology of Music,* ed. D. Deutsch, Academic Press, 1982.