William B. Kuhn
Questar Products
1252 Westover Trace
Acworth, Georgia 30101 USA

A Real-Time Pitch Recognition Algorithm for Music Applications

A number of methods for transcribing audio input into a printed musical score have been described in the literature (Moorer 1977, Piszczalski et al. 1977). Most of this work has concentrated on the relatively difficult problem of transcribing music from polyphonic audio sources. A classic example is provided by Moorer (1977), who considered the difficulties involved in transcribing two guitars played simultaneously. In contrast, the simpler problem of transcribing a monophonic source (single voice) has received little attention. This neglect is unfortunate, since one of the most popular musical instruments, the human voice, is fundamentally monophonic.

This paper describes a commercially viable technique for recognizing the pitch of the human voice in music applications. The method presented is capable of recognizing notes over a three-octave range or greater, in real-time, with a typical accuracy/resolution of 1 percent (less than 20 cents). It has been implemented successfully in both hardware and software configurations. Applications for this technology include melody transcription, computer-assisted vocal instruction, games, and voice-to-MIDI conversion products.

Requirements Analysis

In developing technology for commercial applications, it is important to consider the performance requirements. For a system designed to recognize the pitch of the human voice in musical applications, pertinent requirements include pitch range, audio input dynamic range, accuracy and resolution, the rate at which pitch samples are computed, and the processing time required to perform the recognition. The minimum requirements for general use in musical pitch recognition applications are listed below and discussed in the following sections.

Pitch range	87 Hz–784 Hz
Dynamic range	20–40 db
Accuracy/resolution	1–2 percent
Pitch sample rate	12–24 Hz
Processing time	operation in real-time

Pitch Range

The pitch range requirement is determined by the range of the human voice. In musical applications, voices are usually classified as bass, tenor, alto, or soprano. The practical ranges covered by these voice types is shown in Fig. 1. Although exceptional voices may cover ranges slightly broader than those indicated, written music is usually confined to this range.

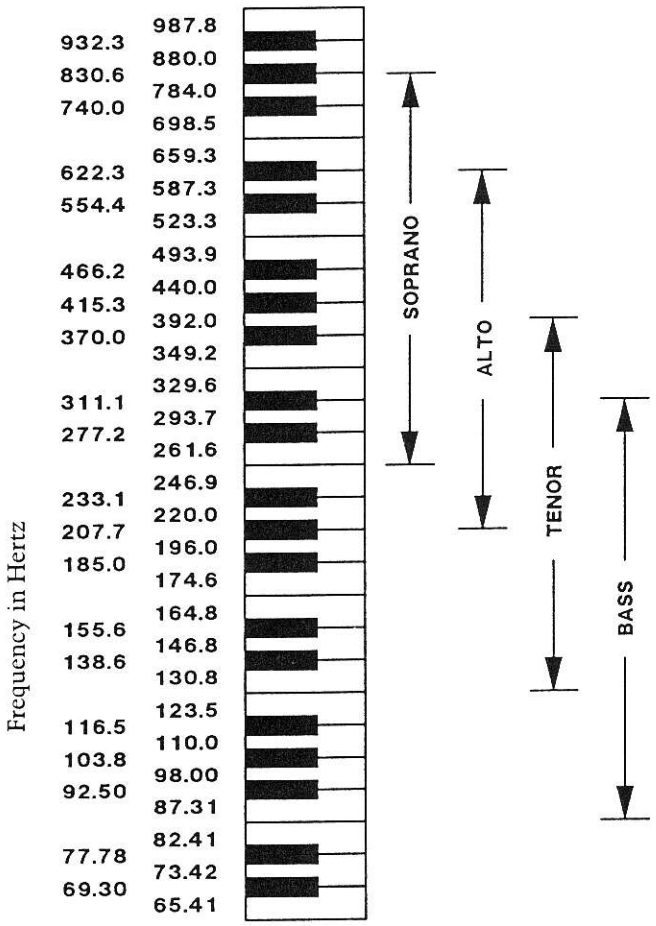
A pitch recognition system designed to cover the full vocal range must therefore cover 39 notes, or approximately three octaves. If range switching is used, a reduced range of approximately two octaves may be satisfactory. However, range switching introduces potential restrictions on the application of the system and should therefore be avoided where possible.

Dynamic Range

The dynamic range of a system determines how accurately the audio input gain must be set, how much movement a vocalist is allowed in holding the microphone, and how much loudness variation the vocalist is allowed to use in his or her voice. Typical sound level measurements obtained with a moderately expensive condenser microphone are

Computer Music Journal, Vol. 14, No. 3, Fall 1990,

Fig. 1. Ranges of four basic singing voice classifications.



shown in Fig. 2. From these measurements it can be seen that a practical dynamic range requirement lies somewhere between 20–40 db. 20 db would allow some microphone movement or limited dynamics, but not both. 40 db reduces restrictions on the user and is therefore preferred.

Accuracy/Resolution

The accuracy and resolution requirement is determined by the pitch spacing between adjacent notes on the musical scale. As illustrated in Fig. 1, this spacing is logarithmic rather than uniform. Therefore, accuracy/resolution figures are best expressed

Fig. 2. Sound level measurements.

Distance	Volume	mV Peak-to-Peak	dBr
0 inches	Loud	100	+20
0 inches	Normal	10	0
8 inches	Normal	1	-20

in percentages. The standard chromatic scale consists of 12 notes per octave, yielding a pitch difference between adjacent notes of approximately 6 percent. Practical accuracy/resolution specifications would be some fraction of this distance. For example, a 2 percent specification may be acceptable for melody transcription applications, whereas a 1 percent limit would be better suited to computer-assisted vocal instruction.

Pitch Sample Rate

The pitch sample rate is a specification of how often the pitch of the audio input signal is measured. Requirements on pitch sample rate are determined by the tempo and note durations expected in the applications. An examination of vocal music reveals that sixteenth note durations are common. Using a typical metronome setting of 90 beats per minute, the duration of a sixteenth note in 4/4 time is 166 msec, corresponding to six notes per second. For transcription and vocal instruction applications, two pitch samples per note may be acceptable, although three to four samples per note would provide better timing resolution and allow for faster tempos. Therefore, the minimum pitch sample rate specification lies in the range of 12–24 samples per second.

Processing Time

The last requirement, processing time, is determined by the application. For systems designed for use in the pitch-to-MIDI application, real-time op-

Fig. 3. Pitch recognition using the FFT.

eration is necessary. For transcription, vocal instruction, and game applications, it may or may not be acceptable for the system to operate slower than real-time. Non-real-time systems are usually constructed using a two-pass design where the input signal is recorded during the first pass and processed in the second pass. This has the disadvantage of delaying feedback to the user and often introduces restrictions on the length of recording that can be made because of signal storage limitations. For vocal training and game applications, the delay in feedback to the user may be especially critical and may rule out may potential uses. Therefore, real-time operation is essential for general application.

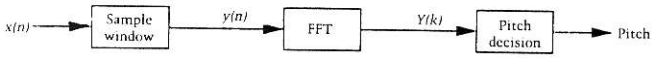
Existing Techniques

Before describing the pitch recognition approach developed in this paper, it is instructive to consider existing techniques and their limitations. For the sake of brevity, only two techniques will be examined—the FFT method and the short-time, autocorrelation method.

FFT Method

Perhaps the best known technique for pitch recognition is the Fast Fourier Transform or FFT (Oppenheim and Schafer 1975). This technique has been used extensively in examining musical waveforms because it maps a sequence of digitized sound samples into the frequency domain where pitch is directly represented.

Figure 3 illustrates the use of the FFT in analyzing a monophonic musical source. A continuous sequence of sound samples $x(n)$ is fed to a windowing function that restricts the view of the waveform to a short segment of time (e.g., 100 msec). The FFT algorithm is then performed on the windowed samples $y(n)$, producing a vector $Y(k)$ of frequency domain coefficients. The pitch of the sound source is then determined by scanning the $Y(k)$ values to locate the first local maximum.



Although the FFT approach is straightforward and appealing, it suffers from one major drawback in the musical pitch application. The frequencies at which the coefficients $Y(k)$ are computed are spaced evenly rather than logarithmically in frequency. Therefore, achieving the desired frequency resolution at the low end of the pitch range necessitates a high dimension FFT, which results in a large number of computations. For example, a system designed to cover a three-octave range while maintaining a resolution of 2 percent or better at all frequencies requires a 512-point transform. The number of computations required in such a transform is approximately $2 \times 512 \log2(512) = 9216$ real multiply-and-add operations plus control flow overhead (Rabiner and Schafer 1978). For a pitch sample rate of 20 Hz, 20 such transforms must be performed per second, for a total of approximately 184,000 multiplications and additions per second, excluding overhead. This computational load is well beyond the capabilities of most currently available personal computers and inexpensive microprocessors.

Autocorrelation Method

The problem of determining the pitch of the human voice has been studied for many years in the field of speech recognition, and several methods have been investigated (Rabiner et al. 1976). Many of these methods fall into the category of time domain approaches, where the algorithms attempt to identify the pitch period directly from the sampled audio waveform without performing an FFT or related transform.

The motivation for the time domain approaches comes from the fact that voiced speech produces regular waveforms where the periodic nature of the signal is readily apparent. An example waveform for the vowel sound *ä* is shown in Fig. 4. The human eye and brain can easily identify the fundamental

Fig. 4. Waveform of vowel sound ă.

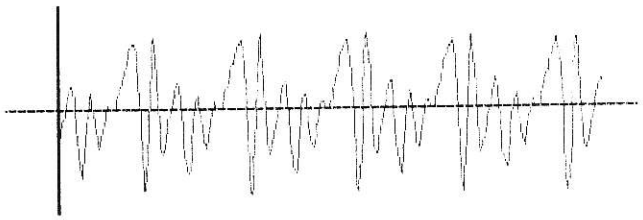


Fig. 5. Pitch recognition using shorttime autocorrelation.

Fig. 5

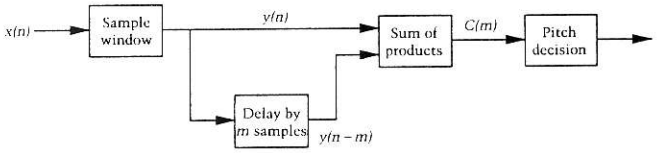
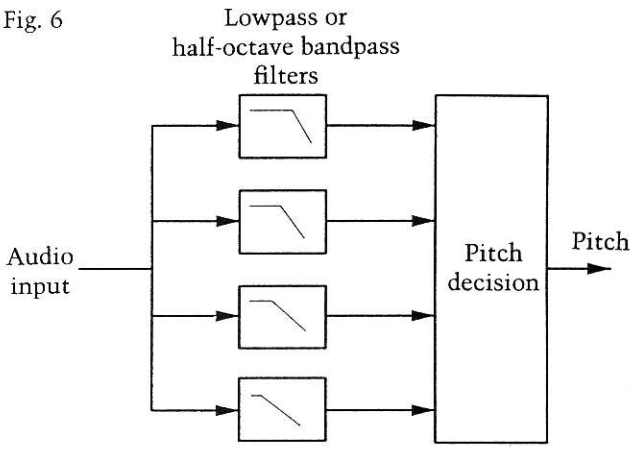


Fig. 6. Pitch recognition using the FPM method.

Fig. 6



pitch of this waveform by looking for the shortest segment that can be used as the basis for the repeating pattern. Unfortunately, implementing such simple pattern recognition tasks on general purpose computers has proven to be computationally intensive and difficult.

Autocorrelation, a classic, time-domain approach, is illustrated in Fig. 5. In this approach as in the FFT, the continuous sequence of sound samples $x(n)$ is fed to a windowing function to restrict the view of the waveform to a short segment of time. To perform the autocorrelation, the windowed samples $y(n)$ are then delayed by m samples and combined with the undelayed $y(n)$ samples to produce a correlation coefficient $C(m)$. This computation is then repeated at varying values of delay m. The correlation coefficients $C(m)$ so generated have the property such that $C(m)$ is a maximum for $m = 0$ and for values of m corresponding to integer multiples of the pitch period (Rabiner and Schafer 1978). In theory, the $C(m)$ values can be scanned to locate the value of the delay m where $C(m) = C(0)$ to determine the fundamental pitch. However, in practice the input waveform is not perfectly periodic, and ambiguities result. Methods have been implemented that successfully deal with this problem, such as center clipping (Rabiner 1977).

The autocorrelation method and its derivatives, although successful in the speech recognition area, have two major disadvantages for musical pitch recognition. First, they require a great deal of computation. For example, if the input waveform is sampled at 10 KHz to provide usable time resolution and a window length of 50 msec is used to encompass a sufficient number of cycles of low pitch inputs, 500 multiplications and additions are needed to compute each coefficient $C(m)$. If 100 values of $C(m)$ are computed at a pitch sample rate of 20 Hz, the total number of multiplications and additions required per second is 1,000,000. The second and equally serious problem with the autocorrelation method is a decreasing resolution at higher pitches. For the same example where the input waveform is sampled at 10 KHz, musical pitches at a frequency of 784 Hz are computed to a resolution of only 7.8 percent, an unacceptable value for musical applications.

Fundamental Period Measurement Method

The essential concept behind the pitch recognition approach developed in this paper is illustrated in Fig. 6. A monophonic musical waveform is fed to a bank of bandpass or lowpass filters whose upper cutoff frequencies are spaced at half octave intervals. If the filters are sufficiently sharp, this arrangement guarantees that one of the filter outputs will contain the input waveform in essentially sinusoidal (i.e., overtone-free) form. The period of this sinewave and the pitch of the input signal can then be determined simply by measuring the time between zero crossings. This approach, which we

Fig. 7. Three-octave FPM implementation.

Fig. 8. FPM pitch decision algorithm.

Fig. 7

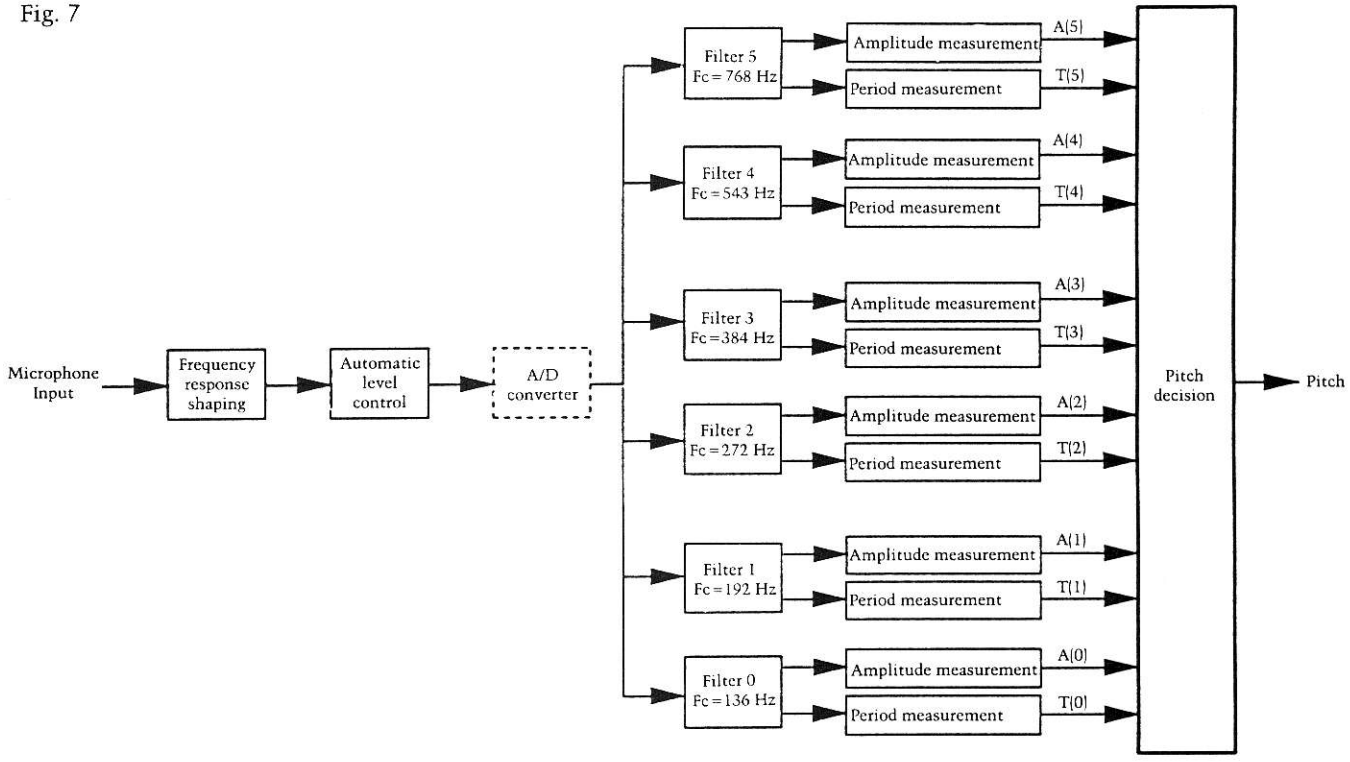


Fig. 8

```
max_amplitude := max( A(0), A(1), ..., A(5) )
if max_amplitude < SILENCE_THRESHOLD
  period := -1
else
  threshold := max_amplitude / 4
  period := -1
  filter := 0
  got_period := FALSE
  while filter <= 5 and got_period = FALSE
    if A(filter) > threshold
      if T(filter) is reasonable for filter
        period := T(filter)
      elseif filter < 5 and T(filter+1) reasonable for filter+1
        period := T(filter+1)
      else
        period := -1
      endif
      got_period := TRUE
    else
      filter := filter + 1
    endif
  endwhile
endif
```

will call the Fundamental Period Measurement (FPM) method, is both accurate and computationally efficient and is suited for both hardware and software implementations.

A more detailed view of a three-octave implementation is shown in Fig. 7 and described in the following paragraphs. The algorithm used to locate the fundamental and make the pitch decision is given in Fig. 8.

Microphone Input

The microphone used with the FPM method must have good low-frequency response characteristics since the information to be analyzed is contained in the fundamental of the audio input waveform. To work well with bass voices, the response should extend down to approximately 80 Hz. Condenser microphones are excellent choices since their response usually extends to 20 Hz or lower. High quality dynamic microphones can also be used.

Inexpensive dynamic microphones often begin their low-frequency rolloff at 150 Hz or higher and should be avoided.

Response Shaping

The frequency response shaping block shown in Fig. 7 consists primarily of a bass boost function and an optional antialias filter. The bass boost should begin at approximately 250 Hz and should provide a pre-emphasis of three to six db per octave. This boost is needed to equalize the amplitude measurements in the output of the filter bank since the fundamental component of the audio input signal is normally weaker than the harmonics, especially in the low pitch ranges. For systems implemented entirely in software, the bass boost can also be implemented in software, but performing the response shaping before the A/D conversion results in better dynamic range.

An optional antialias lowpass filter may be required when the system is implemented in software or when switched capacitor filters are used in the filter bank of a hardware configuration. However, if the system is intended exclusively for vocal inputs, the requirements on the antialias filter are not as stringent. In fact, the filter may not be required at all if the sampling rate is sufficiently high. Speech recognition studies have shown that voiced speech sounds have an inherent lowpass characteristic and that frequencies above about 4 KHz are usually attenuated by more than 40 db (Rabiner and Schafer 1978). Thus, an input sample rate as low as 5 KHz may be adequate. Note that the Nyquist criteria does not need to be satisfied in this application since frequencies from 2.5–4 KHz, although aliased, will be folded into the frequency range above 1 KHz when a 5 KHz sample rate is used. These aliases will be removed by the filter bank and will not affect the amplitude and period measurements.

Finally, note that if the filter bank is to be implemented using lowpass filters, the response shaping function should also be designed to highpass filter the input signal to remove very low frequency noise that may be present in the environment. Air conditioning systems are typical sources of such noise, often producing substantial audio energy at frequencies of 20 Hz or below.

Automatic Level Control

An automatic level control (ALC) function is recommended, especially for software filter bank implementations employing an 8-bit, analog-to-digital converter. Although some level immunity is provided by the algorithm used in the final pitch decision, the useful dynamic range in a software implementation with an 8-bit resolution is limited to about 20 dB since quantization noise in the digital filter bank will degrade performance at low level inputs.

An ALC circuit with carefully selected cut-in threshold and attack and decay times can virtually eliminate level adjustment requirements and restrictions on the user. An attack time of 20 msec and a decay rate of 25 db per second have been used with good results. The ALC cut-in threshold should be chosen so that ambient room noise falls below the silence threshold used in the pitch decision algorithm, and DC offsets associated with ALC gain changing should be carefully minimized, especially in implementations employing lowpass filter banks.

Analog-to-Digital Converter

The analog-to-digital converter (A/D) is drawn with dashed lines in Fig. 7 to indicate that its placement is flexible. Depending on the processing power of the computer or microprocessor available, it may be possible to implement the filter bank entirely in software. In this case, the A/D converter could be positioned immediately following the ALC circuit as shown. However, successful software filtering requires at least 16-bit integer math computing power comparable to that provided by an Intel 8086 family processor (the type used in IBM PC/XT/AT personal computers).

If sufficient processing power is not available in the computer or microprocessor for digital filtering, the A/D conversion can be performed either di-

rectly after the filter bank or after the amplitude and period measurements. Placing the A/D directly after the filter bank requires a high conversion rate A/D to sample all the filter outputs and significant processing power to perform the amplitude and period measurements in software. Alternatively, all functions except the note decision algorithm could be implemented in hardware. In this case, the A/D need only sample the amplitude and period measurements at the pitch sample rate of 12–24 Hz. This approach requires more hardware, but reduces the digital processing load to a level acceptable for any computer.

Filter Bank

The primary requirement in designing the filter bank is to attenuate the second and higher harmonics sufficiently to produce a near sinusoidal output from the filter containing the fundamental of the input waveform. Here, *near sinusoidal* means that the filter output contains the same number of zero crossings as the fundamental. A sufficient condition to prevent unwanted zero crossings can be stated approximately as $a1 > 2 \times a2$, where $a1$ is the amplitude of the fundamental in the filter output and $a2$ is the amplitude of the second harmonic in the filter output. This approximation is derived by comparing the maximum derivative of the second harmonic to the maximum derivative of the fundamental at the point of zero crossing and noting that third and higher harmonics will usually be attenuated to levels where they can be ignored.

To determine the amount of second harmonic attenuation required to satisfy this requirement, the maximum ratio of second harmonic to fundamental amplitude in the input signal must be known. While this issue has not been investigated rigorously, an attenuation of the second harmonic by 25 db or more has been found to be sufficient for a wide range of voices tested. This attenuation can be achieved with fourth order 0.5 db lowpass Chebyshev filters or with sixth order, half-octave, bandpass Butterworth filters.

When choosing a filter, it is also important to consider the filter's time domain response characteristics. Ringing and highly nonlinear phase response should be minimized, especially in the low-octave filters, to avoid waveshape distortions that could affect period measurements. This requirement translates to using the lowest order filter consistent with the attenuation requirement and avoiding filters with excessively sharp cutoffs, such as high ripple (> 1 db) Chebyshev and elliptical designs.

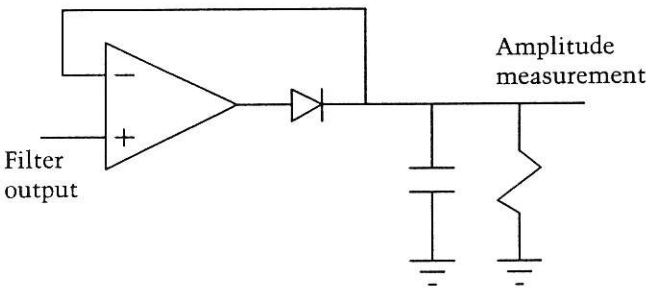


Fig. 9. Amplitude measurement circuit.

Amplitude Measurement

The amplitude measurement function can be implemented in hardware with a straightfoward precision rectifier circuit and smoothing capacitor, as illustrated in Fig. 9. To follow variations in the input signal, the output of the rectifier should be designed to decay rapidly to zero when the input signal is removed. Since the lowest frequency of interest is approximately 100 Hz, a reasonable decay rate is 300 db per second. This rate limits the maximum ripple in the amplitude measurement to 3 db while allowing the measurement to decay by 30 db in 0.1 sec.

The amplitude measurement can be implemented in software simply by determining and saving the maximum input value. In this case, no decay is needed since the maximum can be reset to zero for each new amplitude measurement sample.

Period Measurement

The period measurement function can be implemented in hardware as a zero-crossing detector circuit that enables a counter for a predetermined

Fig. 10. Period measurement circuit.

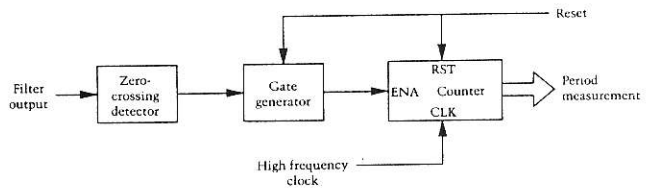
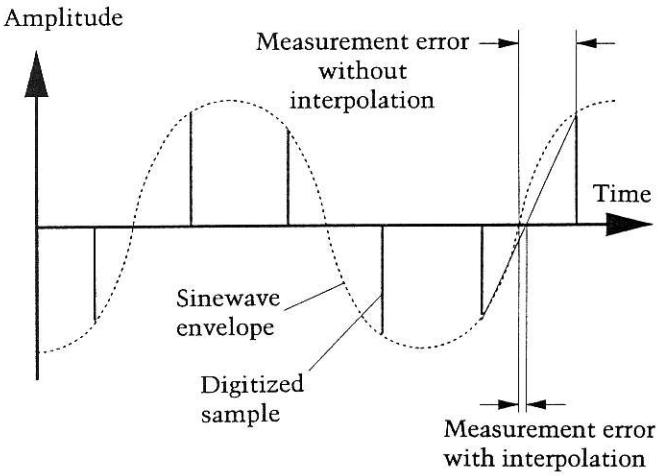


Fig. 11. Linear interpolation of zero-crossing location.



number of zero crossings, as illustrated in Fig. 10. To eliminate errors resulting from DC offsets in the input to the zero-crossing detector, an even number of zero crossings should be used in gating the counter. Note that holding the counter open for multiple cycles improves the accuracy of the measurement and lowers the counter clock frequency required to achieve a specified resolution. The number of cycles that can be averaged in this way depends on the pitch sample rate and may range from two to three cycles in the lowest octave, to 10 or more in the upper octave.

The same basic concept can be used in a software implementation. Here, however, the signal sample rate takes the place of the counter clock. The software processes the output of the filter to determine the number of samples between a preset number of zero crossings. Since the sample rate will usually be low to minimize signal processing demands on the computer, a simple linear interpolation can be performed (as illustrated in Fig. 11) to improve the measurement by determining fractional sample periods at the beginning and ending zero crosses. This linear interpolation leads to substantial improvements, provided the ratio of sample rate to signal frequency is approximately four to one, or higher.

Pitch Decision Algorithm

The algorithm used to determine the pitch of the fundamental from the amplitude and period measurements is shown in Fig. 8. The maximum of the amplitude measurements $A(0)$–$A(5)$ is determined and compared against a silence threshold to determine if the input is silence, which is encoded as a period of -1. Otherwise the amplitude measurements are scanned, beginning at the lowest frequency filter, for the first filter with an appreciable amplitude. The fundamental of the input signal is then assumed to lie in the passband of this filter or the filter immediately above it. The period in the filter's output is read and checked to see if it is reasonable for the filter (within the half octave covered by that filter). If so, it is taken as the period of the input signal. If not, the period in the next higher filter is read. If it is reasonable for its respective filter, it is taken as the period. Otherwise, the period is set to -1 (equivalent to silence), representing a problem in recognizing the pitch.

Note that the threshold used in searching for the filter containing the fundamental is computed as a fraction (shown as 1/4) times the maximum amplitude. This relative threshold is used because the fundamental may be relatively weak compared to the second harmonic. If the fraction used in computing the relative threshold is too large, the *while* loop may skip over the filter containing the fundamental and locate the second harmonic instead. Conversely, if the fraction is too small, low frequency noise in the input signal may be incorrectly interpreted as the fundamental pitch. Experience has shown that a factor of 1/4 provides good results when the filter bank is composed of lowpass filters. When bandpass filters are used, the choice of this fraction is less critical because low frequency noise is reduced in the filter outputs, and signals emerging from the filters above the one containing the fundamental will encompass fewer harmonics.

Fig. 12. PC card used in a hardware implementation.

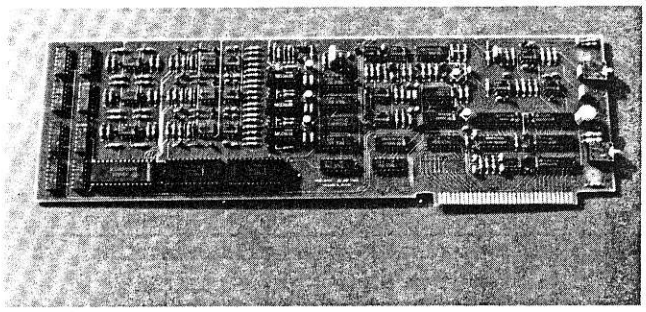
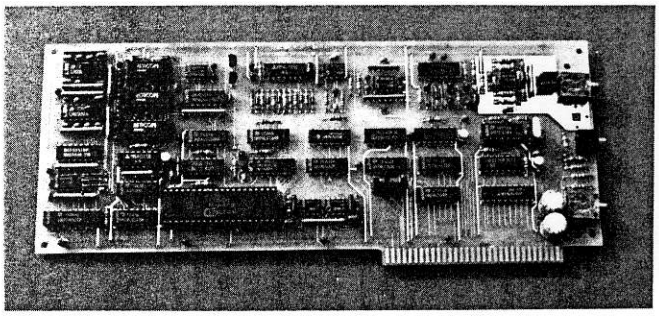


Fig. 13. PC card used in a hybrid implementation.



Implementation

The FPM pitch recognition method has been implemented in hardware, hybrid (hardware and software), and software configurations. The hardware and hybrid implementations were developed on an IBM PC compatible computer using the plug-in cards shown in Figs. 12 and 13, respectively. The filter bank in each of these implementations consists of switched capacitor, half-octave, bandpass filter ICs. In the hardware configuration, the amplitude and period measurements are performed on the circuit board and the computer simply reads the measurements and performs the pitch decision algorithm at the IBM PC's clock interrupt rate of 18.2 Hz. In the hybrid configuration, the filter outputs are digitized and the software performs the amplitude and period measurements in addition to the pitch recognition algorithm. On a standard IBM PC operating at 4.77 MHz, this processing consumes approximately 50 percent of the available CPU power, although a significant fraction of this time is spent controlling the A/D converter through the processor's I/O ports and could be reduced by using direct memory access (DMA) circuitry for writing the samples directly to memory.

In 1988, Tandy Corporation introduced the first IBM PC compatible computers with built-in microphone input and A/D circuitry. These computers employ 8 MHz 8086 and 80286 processors and have been used to implement the FPM method entirely in software by designing the filter bank with digital lowpass filters. The A/D operates with the computer's DMA circuitry to write converted samples directly to memory. Therefore, nearly the entire processing power of the CPU is available even when conversion is in progress. The software implementation of the FPM method consumes approximately 50 percent of the available CPU power on the 8086-based model and significantly less on the 80286-based one.

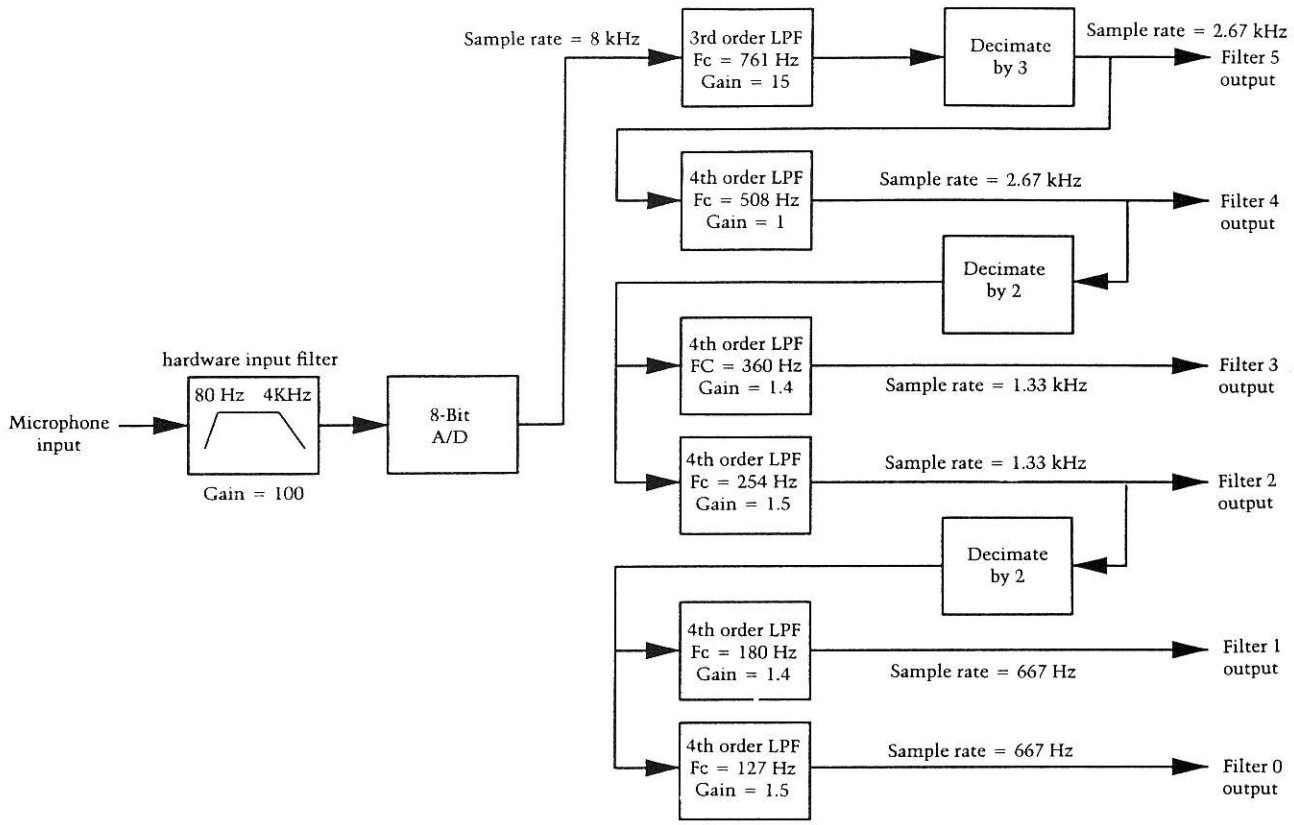
Digital Filter Bank Implementation

The most computation-intensive part of a software FPM implementation is the digital filter bank. Therefore, to estimate the feasibility of implementing the method entirely in software on a given processor, an estimate of the filter bank processing requirements should be performed.

For the 8086 and 80286 processors, computation time in performing the digital filtering operation is dominated by the 16-bit multiply instructions. Hence, a reasonable lower bound on the processing requirements can be found by determining the number of multiply operations performed. For example, using a sample rate of 8 KHz and a bank of six fourth order lowpass filters, each requiring four nontrivial multiply operations, the number of multiplies required per second is $8000 \times 6 \times 4 = 192{,}000$. Since a 16-bit multiply operation on the 8086 requires an average of 141 clock cycles, or 18 μsec at an 8 MHz clock rate (Intel 1981), real-time processing is not feasible with this straightforward approach on such machines.

To reduce the number of multiplications, careful consideration must be given to the choice of sample rates. The configuration used in the final

Fig. 14. implementation using digital filters.



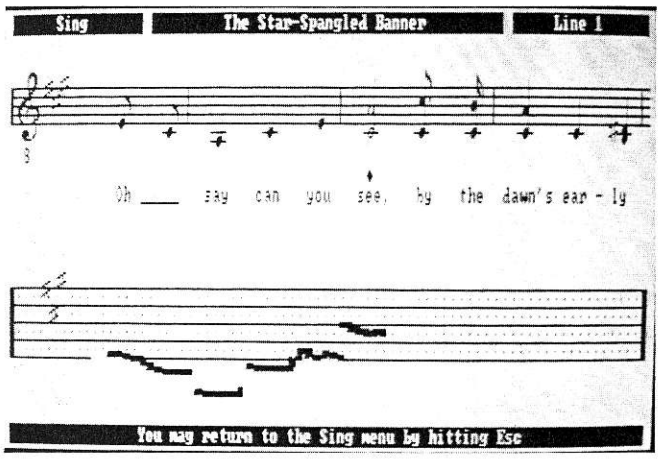
implementation is shown in Fig. 14. This configuration takes advantage of the fact that lowpass filters are used, so the sample rate can be progressively reduced as lower octaves are processed. With the cutoff frequencies and sample rates used, aliases that fall into the lower filter passbands with this configuration are at least 35 dB down, which is well within acceptable limits to prevent distortions in the amplitude and period measurements.

The total number of nontrivial multiplications required per second with this design is only 50,670, or a reduction of nearly 4:1 over the previous approach. Unfortunately, while feasible for faster 80286-based processors, this computation requires more than 85 percent of the 8086 CPU time and is still unacceptable.

Reducing the processing load further requires that each multiplication operation in the digital filters be replaced by a sequence of a small number of shift-and-add operations. This is equivalent to using coefficients that are not ideal in the digital filters, which distorts the filter frequency responses from their ideal shapes. However, by careful selection of coefficients, the cutoff frequencies and passband ripple of the resulting filters can be kept sufficiently close to the desired filter characteristics.

By writing the filter processing code in assembly language, the shifts and adds can be performed exclusively with register operands. The time for each shift-and-add operation on the 8086 processor is then determined by the instruction fetch time, which is four clock cycles. When operating at 8 MHz, the time per instruction is 0.5 μsec. Thus, by limiting the average number of shifts plus adds to eight, a simulated multiply will require only 4 μsec, and real-time operation is easily achieved.

Fig. 15. Real-time pitch display.



Performance of the Finished Product

The FPM method has been programmed as a set of background assembly language routines in both the hybrid and all-software configurations. In the all-software configuration, these routines use a doubled-buffered-memory approach to continuously sample and filter the input signal. Pitch values are generated by the algorithm of Fig. 8 at a rate of 18 times per second and then smoothed by a simple three-point median filter to minimize possible errors, especially during rapid rise and fall times of an input waveform. The finished routines have been used to create a music education program intended for melody transcription and beginning level vocal music instruction.

In the melody transcription application, the program allows a user to create a musical score by singing into a microphone while tapping on the computer's keyboard at the beginning of each note. At each tap, the program places the note sung onto a musical staff in common-practice music notation. Since pitch recognition occurs in real-time, the notes appear instantly and there is no limit to the length of melody that can be transcribed.

In the vocal training application, the program provides various real-time pitch displays to help beginning–level singers assess and improve their skills. In the display shown in Fig. 15, the pitch contour of notes played by the computer is plotted as a dotted line moving from left to right. The pitch contour sung is similarly plotted as a solid line moving from left to right. When the solid line overlays the dotted line, the user knows he or she is on-key. By using this display to achieve proper pitch, and by concentrating on the sound of his or her voice, an uncertain singer can learn basic pitch matching skills in a self-paced, non-threatening environment. More advanced singers can use the same display to test and improve sight singing and harmony skills.

A special "tuning indicator" display also provided in the program has been used to confirm that the FPM method provides an accuracy and resolution of better than 20 cents at all notes in the three–octave vocal range. Use of the program with a wide range of male and female adult singers has further verified that the algorithm provides smooth, glitch-free pitch contours when singing either standard vowels, solfège syllables, or arbitrary lyrics.

The number of children tested to-date is insufficient to draw any firm conclusions, but no problems are anticipated with either male or female children's voices. The only observed limitation of the algorithm is a tendency to incorrectly select the second or higher harmonics of musical tones that contain a very large number of overtones in their spectra, or which have an excessively low fundamental amplitude. Examples of this problem have been seen when testing the program with inexpensive synthesizers that use speakers too small to reproduce the fundamental.

The program described above is currently available as a shareware product for Tandy 1000 SL/TL series computers and can be obtained either by downloading from the PC-LINK information service software library (search for keyword "MTS"), or by writing to the author. Source code for the assembly language routines used in pitch recognition can also be obtained by contacting the author.

Summary and Conclusions

The recognition of musical pitch from monophonic audio sources, particularly the human voice, offers numerous potential applications including melody transcription, vocal instruction, games, and pitch-

to-MIDI conversion. A practical pitch recognition algorithm for these applications should operate in real-time, cover a three–octave range with an accuracy/resolution of 1–2 percent, a dynamic range of 20–40 dB, and a pitch sample rate of 12–24 Hz. Existing approaches such as the FFT and autocorrelation method suffer from the relatively large number of computations required for their implementation on inexpensive computers and from accuracy/resolution problems associated with the wide pitch range and logarithmic spacing of the musical scale.

The FPM method described in this paper provides a computationally efficient and accurate solution to the monophonic pitch recognition problem. The FPM method has been implemented in real-time on IBM PC compatible computers in hardware, hybrid, and software configurations and can be implemented easily on other computers and microprocessor-based music products.

References

Intel. 1981. *iAPX 88 Book* Santa Clara, California: Intel Corporation.

Moorer, J. 1977. "On the Transcription of Musical Sound by Computer." *Computer Music Journal* 1(4):32–38.

Oppenheim, A., and R. Schafer, 1975. *Digital Signal Processing.* Englewood Cliffs, New Jersey: Prentice-Hall.

Piszczalski, M. et al. 1977. "Automatic Music Transcription." *Computer Music Journal* 1(4):24–31.

Rabiner, L. et al. 1976. "A Comparative Performance Study of Several Pitch Detection Algorithms." *IEEE Transactions on Acoustics, Speech, and Signal Processing* 24(5):399–418.

Rabiner, L. 1977. "On the Use of Autocorrelation Analysis for Pitch Detection." *IEEE Transactions on Acoustics, Speech and Signal Processing* 25(1):24–33.

Rabiner, L., and R. Schafer. 1978. *Digital Processing of Speech Signals.* Englewood Cliffs, New Jersey: Prentice-Hall.