Keith Lent
Departments of Music and Electrical Engineering
University of Texas at Austin
Austin, Texas 78712 USA

An Efficient Method for Pitch Shifting Digitally Sampled Sounds

Altering the pitch of digitally sampled sounds is an operation that is performed by a number of commercially available devices (sampling synthesizers, harmonizers, vocoders) as well as by the software of many computer music languages. However, the algorithms for resampling and formant analysis/resynthesis that these devices implement have some drawbacks: they can be computationally expensive and can produce audible side effects, among other things.

Resampling, which is achieved by modifying the sampling rate, changes not only the pitch of the sound, but also its length and formant shape. The length change is usually compensated by a compression or expansion of the sound (Lee 1972), but this often introduces pops, clicks, and other modulations. The shift in the formant structure, which creates the familiar chipmunk-like sound, also causes the sound to be unrecognizable after only one octave or so of pitch shift.

The analysis/resynthesis methods involving linear predictive filters (Oppenheim and Schafer 1975) and Fourier transforms (Dolson 1986) have the ability to retain the formant shape but require a large amount of computation.

The goal of this paper is to present an alternative algorithm for altering the pitch and length of pitched (pseudo-periodic) sampled sounds. It has a computational efficiency near that of the resampling method while maintaining the formant characteristics of the unshifted sound.

The Algorithm

The algorithm we've implemented can be divided into three sections: a pitch tracker, a compressor/expander, and a pitch shifter. The pitch tracker dynamically determines the fundamental pitch of the input sound. Because this algorithm is designed to work on monophonic, pitched (pseudo-periodic) sources, such as speech or monophonic instruments, it is possible to track the pitch using a crude but simple and efficient method. The input sound is first passed through a fourth order bandpass *infinite impulse response* (IIR) filter—the equivalent of two Music-V-style resonator unit generators. This filter is designed to pass only the fundamental harmonic of the source. Hence, if the source was a sung middle A, then the filter could have a center frequency of 440 Hz and a bandwidth of perhaps 200 Hz. This would clearly allow the fundamental harmonic near 440 Hz to pass and attenuate all of the others. The output of this filter is then passed through a zero crossing detector. Since only the fundamental frequency should pass through the filter, the filtered signal should move through or cross the zero point only twice per period. The period length, and thereby the pitch, can be determined from the location of these zero crossings.

The process of time compression/expansion is necessary in order to compensate for the lengthening or shortening of the source sound caused by the pitch shifter. Once the pitch of the original sound is known, the waveforms for each cycle or period of the source can be identified. To lengthen a sound, individual cycles of the input signal can be repeated in the output signal at appropriate points; to shorten a sound, cycles can be removed. For example, to expand or double the length of the sound, one simply needs to repeat each cycle of the original sound twice. Likewise, to halve the length of the sound, every second cycle of the source should be removed. This simple but effective method for changing the length of sounds is very similar to that which is performed in the hardware of real-time time compressor/expanders.

To understand the pitch shifter, note that in resampling, the simplest method of pitch shifting, the sounds are played back at a rate different from

Computer Music Journal, Vol. 13, No. 4, Winter 1989,
© 1989 Massachusetts Institute of Technology.

Fig. 1. A periodic sampled sound.

Fig. 2. A rectangular window.

Fig. 3. A cycle of the sampled sound extracted using a rectangular window.

Fig. 4. Adding the windowed versions recreates the sound.

Fig. 5. A lower pitched version of the sound.

Fig. 1

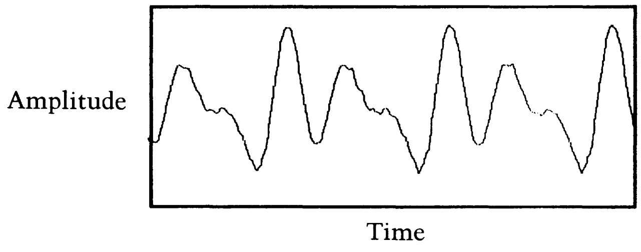


Fig. 2

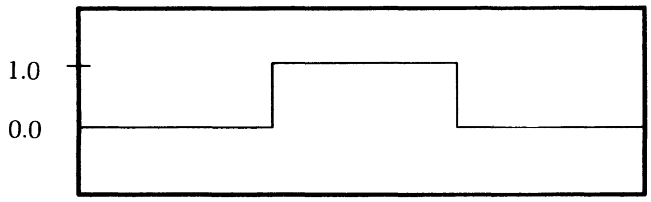


Fig. 3

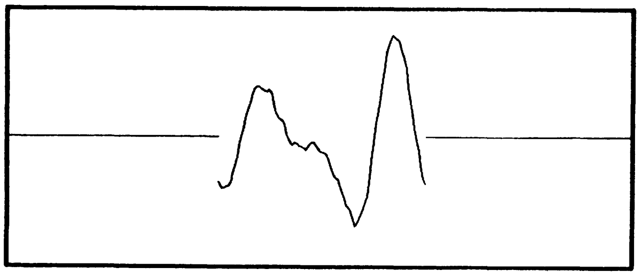


Fig. 4

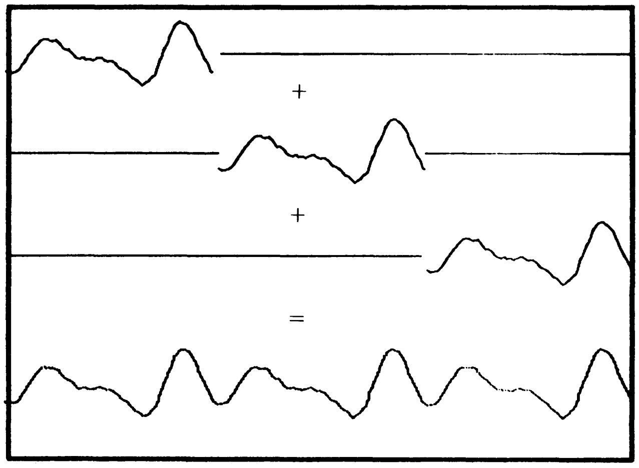
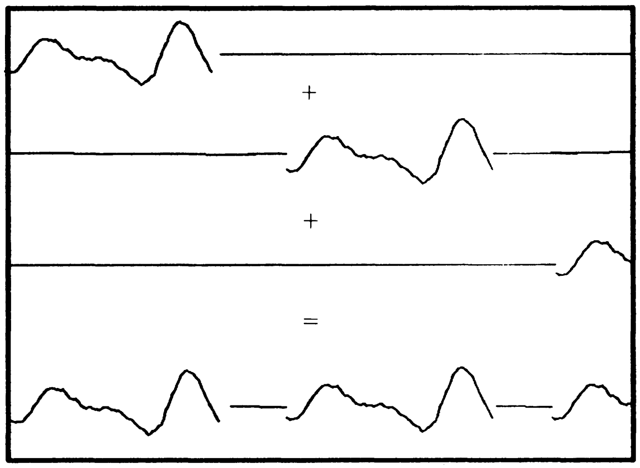


Fig. 5



that at which they were recorded. This means each period is simply stretched to make the pitch lower and compressed to raise it. In the algorithm presented here, however, each period is made longer, and thereby lower, by playing each original period at the normal rate and then adding some silence (i.e., zero-valued samples) until the next period begins. Likewise, to raise the pitch each period is played back normally but overlapping the previous period. This causes the resultant period to be shorter than the original by the amount of overlap. The advantage of this method of pitch shifting is that the overall spectral envelope of the sound is not changed when the pitch changes; hence, we achieve results with quality comparable to *linear predictive coding* (LPC) analysis/resynthesis.

Actually, the previous description is something of an oversimplification. If this algorithm were implemented exactly as described, it would properly shift the pitches, but it would also introduce some higher frequency artifacts. This is primarily due to the discontinuities that would occur when zeros are added to the end of each cycle without smoothing. This problem is rectified by a process known as *windowing*.

Windowing (Ziemer, Tranter, and Fannin 1983) is a method for viewing or extracting a small part of a continuous signal. In this case, that signal is a periodic sampled sound like the one shown in Fig. 1. In order to extract a single period of the signal, a *rectangular* window having length equal to one period of the sound could be used, as in Fig. 2. By multiplying every point in the signal by the correspond-

Fig. 6. The spectrum of the original sound.

Fig. 7. The spectrum of the windowed version (Fig. 3) superimposed on the original sound's spectrum.

Fig. 8. The spectrum of the windowed version (Fig. 3) superimposed on the spectrum of the lower pitched version (Fig. 5).

Fig. 6

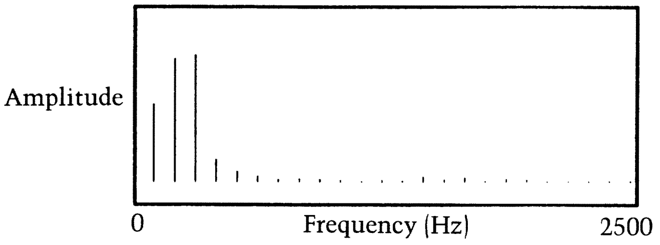


Fig. 7

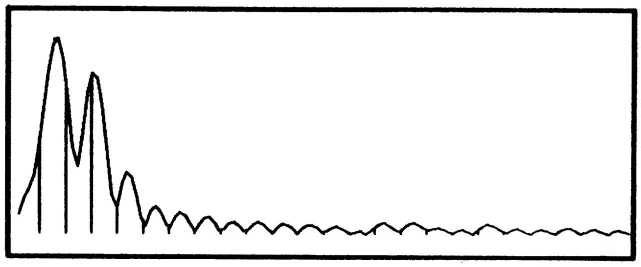
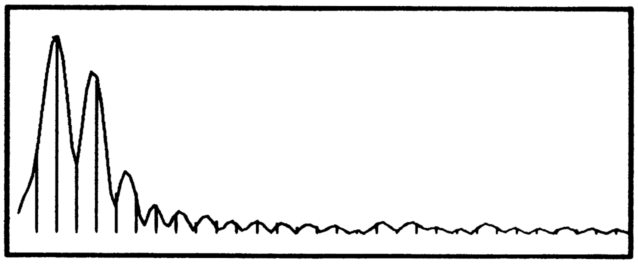


Fig. 8



ing value of the window function, a signal that is zero everywhere except for one cycle of the original sound is obtained and is shown in Fig. 3. If a different, delayed rectangular window is used, then the next period of the sound can be similarly extracted. Notice that the original signal can be recreated from the windowed versions by adding all of the windowed versions together, as shown in Fig. 4. Also, if a slight delay is introduced between each of the windowed signals before they are recombined, then a slightly lower pitched version of the original sound results as in Fig. 5. This process is in fact exactly the same as the method of pitch lowering (i.e., inserting zeros) that was described earlier.

It is clear from Fig. 5 that this method would introduce some higher frequency components because of the discontinuities that are present when the zero valued sections begin and end. The greater presence of higher frequencies in the pitch shifted sound can also be seen by viewing the Fourier transforms or spectral plots of the signals. First, Fig. 6 shows the spectral content of the original period signal of Fig. 1. Notice that because the sound is periodic it has very clear spectral lines at the fundamental frequency and each of its harmonics. Figure 7 shows the spectral content of the windowed signal of Fig. 3. This windowed signal is clearly not periodic and therefore its spectral plot does not contain specific spectral lines. Instead, it has a continuous "lumpy" spectrum characteristic of rectangularly windowed signals. It is important to notice, however, that the values of the spectral plot in Fig. 6 are the same as those in Fig. 7 at each of the harmonic frequencies.

This result can be clearly demonstrated through Fourier analysis and is true for all periodic signals that are windowed in this manner (see Appendix A).

Using this type of analysis, the spectral plot for the pitch lowered signal of Fig. 5 can be inferred from the windowed spectrum of Fig. 7. One only needs to use the values of the windowed spectrum at the new, lower pitched, harmonic frequencies. This is demonstrated in Fig. 8. In other words, the shape of the windowed spectrum remains the same as the harmonics are shifted, and therefore, the formant characteristics of the sound will be retained by this method of pitch shifting.

It can now be seen that the pitch-lowered version will contain a different and probably larger amount of higher frequency components. This is due to the lumpy interpolation between the original signal's harmonics that occurs with rectangular windowing. As the harmonics are shifted, their amplitudes will rise and fall with the lumps in the higher end of the windowed spectrum. It is important to realize that the lumpiness in the spectral domain is directly related to the discontinuities caused by rectangular windowing. To correct this problem, a different, smoother window function can be used.

Figure 9 shows the shape of a Hanning window. It is smooth and continuous so that when it is multiplied with the original sound it does not introduce discontinuities. These windows are twice as long as

Fig. 9. A Hanning window.

Fig. 10. Once again adding windowed versions recreates the signal.

Fig. 11. The spectrum of the Hanning windowed signal superimposed on the original sound's spectrum.

Fig. 9

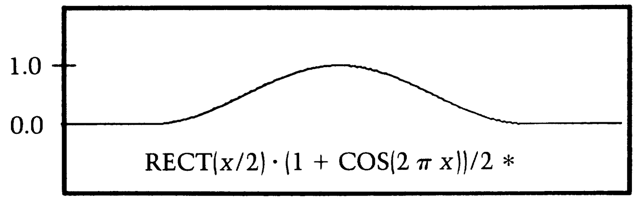
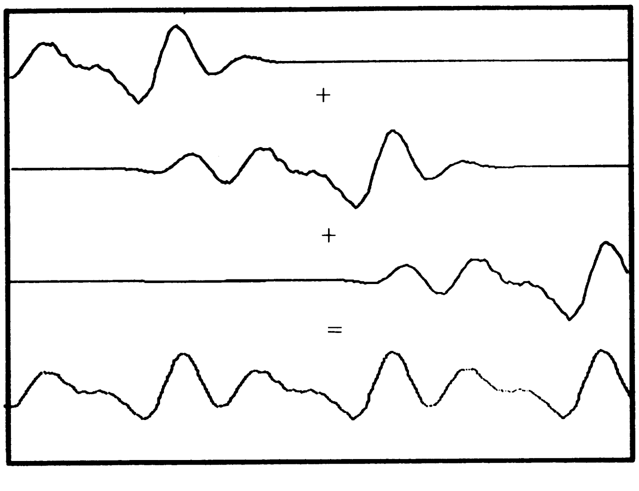
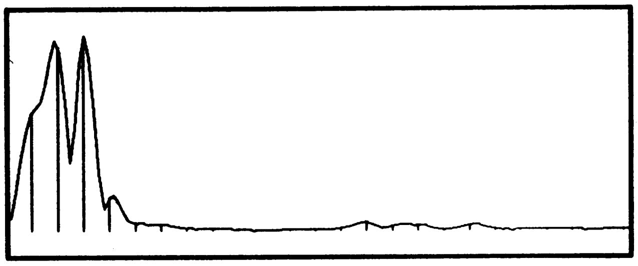


Fig. 10





the rectangular windows, i.e., two periods in length, and therefore they will overlap when each cycle is windowed. However, because of the symmetric shape of the Hanning window, simply adding the windowed versions together will once again yield the original signal, as shown in Fig. 10. Also, when the windowed signals are delayed to lower the pitch, the resulting signal will still be smooth and free of discontinuities.

In the frequency domain the effect of the Hanning window can also be seen. Figure 11 shows the spectral content of the Hanning windowed signal. Notice that there is very little of the lumpiness that was present in the rectangular window's spectrum and that the interpolation between the harmonic amplitudes of the original sound is much more smooth. This smoothness is especially evident in the higher frequencies. Clearly, when this window is used, the amplitudes of the higher frequency harmonics will not be significantly changed when the pitch—and thereby the harmonics—is shifted. This means that by using Hanning windows only a negligible amount of higher frequency artifacts will be introduced when the pitch is shifted.

Implementation

The algorithm can be implemented as shown in the pseudocode of Fig. 12. Its input variables are the following:

Nin—number of input samples
IN(n) for $n = 1, 2 \ldots Nin$—the array of input samples
Nout—number of output samples
periodratio—factor by which the period length should be altered: i.e., the reciprocal of the frequency shift ratio
OUT(n) for $n = 1, 2 \ldots Nout$—the array of pitch shifted output samples

The algorithm in Fig. 12 begins with an array of sound samples *IN(n),* which is to be transformed (pitch shifted and compressed/expanded) into an array of output samples *OUT(n).* Both an input pointer (inptr), for the *IN* array and an output pointer (outptr), for the *OUT* array are initially set to the beginning of their respective arrays. As the algorithm proceeds, these pointers will increment through each of the array's samples until the input pointer exceeds the number of input samples: i.e., inptr $\geqslant$ *Nin.*

The first part of the algorithm, the pitch tracker, works by bandpass filtering the input samples *IN(inptr)* and storing the bandpassed sample in a variable called *x.* The variable, *oldx,* is set to contain the previous value that came out of the filter.

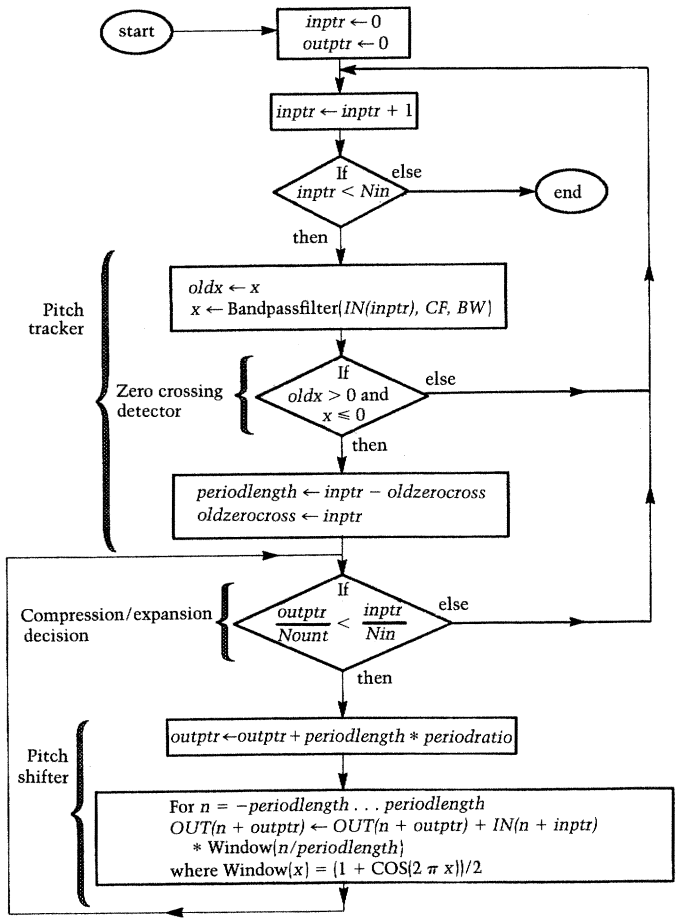

start
inptr ← 0
outptr ← 0
inptr ← inptr + 1
If inptr < Nin
else
end
then
Pitch tracker
oldx ← x
x ← Bandpassfilter(IN(inptr), CF, BW)
Zero crossing detector
If oldx > 0 and x ≤ 0
else
then
periodlength ← inptr − oldzerocross
oldzerocross ← inptr
Compression/expansion decision
If outptr/Nount < inptr/Nin
else
then
Pitch shifter
outptr←outptr + periodlength * periodratio
For n = −periodlength . . . periodlength
OUT(n + outptr) ← OUT(n + outptr) + IN(n + inptr) * Window(n/periodlength)
where Window(x) = (1 + COS(2 π x))/2

Fig. 12. The pitch shifting algorithm.

Then, if the value of *oldx* is greater than zero, and the value of *x* is less than or equal to zero, the output of the filter will have passed through zero. This condition enables the program to determine where a zero crossing—and thereby a new period—is located. When this condition is satisfied, the *periodlength* can be determined by taking the difference between the present location (*inptr*) where a zero crossing has occurred and the location where the previous zero crossing (*oldzerocross*) occurred. This difference yields the length of the period of one cycle of the input signal in samples (*periodlength*). The final step in the pitch tracker is to set *oldzerocross* to *inptr* so that the next time around, *oldzerocross* will again contain the previous zero crossing location.

The second part of the algorithm, the time compression/expansion decision, is reached when a zero crossing has occurred: i.e., for each period in the original signal. The function of this time compression/expansion decision is to determine whether the output pointer is running proportionally ahead of or behind the input pointer. Remember that since this algorithm allows the user to specify both the input and output lengths (*Nin* and *Nout*), independent of the frequency shift (determined by *periodratio*), it is necessary to determine whether cycles of the input signal should be repeated or eliminated when creating the output signal. Hence, by comparing the fraction of output samples that have been calculated, (*outptr/Nout*) with the fraction of input samples (*inptr/Nin*), the program can determine if, at the present location, the samples for the input period should be placed in the output array or should simply be skipped.

The third part of the algorithm, the pitch shifter, is responsible for filling the output sample array *OUT* with the appropriately windowed samples from the input array. Notice that this part of the algorithm can be reached only when the compression/expansion decision has decided that more periods need to be added to the output array. Hence, the pitch shifter begins by updating the output pointer (*outptr*) to point one period length (scaled by the pitch shift factor *periodratio*) ahead of its current position; i.e., *outptr* becomes *outptr* + (*periodlength* * *periodratio*). Then, a windowed cycle of the input signal is added (overlapped) into the output array. This is done by taking the samples from *IN(inptr-periodlength)* to *IN(inptr+periodlength)*, multiplying them by the appropriate window factor, and adding them into the output array at locations *OUT(outptr-periodlength)* to *OUT (outptr+periodlength)*. In this way, one windowed cycle of the input signal has been transferred to the output array at the appropriate period shifted location. At this point, control returns to the compression/expansion decision to determine whether more cycles should be repeated in the *OUT* array before going to the next period of the input signal. Finally, the algorithm will terminate when the input pointer reaches *Nin* signaling that all of the input signal has been processed.

It is important to note that the pseudo-code does not implement interpolation when calculating the period length and when reading and writing the samples. However, interpolating calculations can be easily added and will greatly improve the quality of the output sound.

Platforms

This algorithm has been implemented and tested on an IBM-PC running PCMIX, a version of Paul Lansky's CMIX. From these tests, two conclusions can be made. First, the algorithm was found to work quite well on speech sounds. Spoken words, transposed up and down over two octaves, remained clearly intelligible and could be easily identified with the speaker. Also, piano tones could be transposed by nearly an octave before sounding artificial. Second, this method can be made very efficient computationally. The pitch tracking can be implemented with only four multiplies per sample, while the windowing requires only two multiplies. This is clearly small enough to be realistically performed by a microcomputer and in real-time in a specialized signal processor. In fact, a project has recently been completed at the University of Texas at Austin to implement this algorithm in real-time using a Motorola 56000 microprocessor (Lent, Pinkston, and Silsbee 1989).

Discussion

It is interesting to make a comparison between this method and two other synthesis techniques, *granular synthesis* (Roads 1985), and *formant wave function synthesis* (FOF) (Rodet 1984). Both of these techniques, as well as the method presented here, synthesize sound by adding and overlapping a series of windowed or enveloped signals. These windowed signals are called *grains* in granular synthesis and *formant wave functions* in the FOF method.

In granular synthesis, finite duration "Gaussian" envelopes are used to effectively window portions of sine tones at various frequencies. These windowed tones or grains can then be combined (added and overlapped) to create a wide variety of sounds. It is interesting to notice that the Gaussian envelopes closely resemble the Hanning windows used in this paper.

In both methods the envelopes serve to extract a short portion of a continuous signal without introducing discontinuities. Both methods also synthesize their sound by adding and overlapping these enveloped signals. However, the similarity between these two methods ends here. In granular synthesis, the grain envelopes often have fixed durations. In contrast, this algorithm determines the window's duration and spacing dynamically, based on the current pitch. In granular synthesis, spectrally rich grains are created by combining many different pitched grains together, whereas in this algorithm, the windowed signals derive their spectral structure from the spectral information in each period of the original sound.

In FOF synthesis, the goal is to calculate directly the response of a filter to a given excitation function in the time domain. This response is typically on the order of 10–30 ms long. A periodic sound can then be generated by adding copies of this FOF spaced by the fundamental period of the sound to be created. Note that this is equivalent to driving a series of impulses into a filter that has the same response or wave function. The FOF method is particularly attractive because it can generate sound much more efficiently than the equivalent digital filter.

It is interesting that the FOFs, in their simplest form, are windowed sine tones, and, as in granular synthesis and the method presented here, these windowed tones form the basis from which sound can be created. Although the final sound in the FOF technique is generated by adding and overlapping a number of FOFs, it differs from the granular technique in that the grainlike FOFs are spaced at variable lengths rather than at fixed length intervals. These intervals then determine the fundamental pitch of the resulting sound. This is much like the way this new algorithm's pitch shifted output sound is generated.

The FOF method differs from the pitch shifting algorithm in the way that the overall spectral structure of the final sound is formed. In FOF synthesis, the final sound is usually the sum of a number of wave functions that have been chosen to simulate a particular spectrum. The pitch shifting method, as was mentioned before, derives its spectrum from the information in each period of the input signal.

Conclusion

In conclusion, this algorithm and its variants should allow high quality, realistic pitch shiftings of sampled sounds to be performed on even the most modest of computer music systems.

Acknowledgments

I would like to thank Dr. Russell Pinkston for his assistance in preparing this paper, as well as Dr. Harold W. Smith and David Mielke for their helpful comments and encouragement.

References

Dolson, M. 1986. "Phase Vocoder: A Tutorial." *Computer Music Journal* 10(4): 14.

Lee, F. 1972. "Time Compression and Expansion of Speech by the Sampling Method." *Journal of the Audio Engineering Society* 20(9): 738–742.

Lent, K., R. Pinkston, and P. Silsbee. 1989. "Accelerando: A Real-Time General Purpose Computer Music System." *Computer Music Journal* 13(4): 54–64.

LoCascio. M. 1987. "Audio Time Companson for Studio and Performance Synchronization." *Proceedings of the 1987 International Computer Music Conference.* San Francisco: Computer Music Association.

Oppenheim, A. V., and R. W. Schafer. 1975. *Digital Signal Processing.* New York: Prentice-Hall, pp. 236–237.

Roads, C. 1985. "Granular Synthesis of Sound." In C. Roads and J. Strawn, eds. *Foundations of Computer Music.* Cambridge, Massachusetts: MIT Press.

Rodet, X. 1984. "Time-Domain Formant-Wave-Function Synthesis." *Computer Music Journal* 8(3): 9–14.

Ziemer, R. E., W. H. Tranter, and D. R. Fannin. 1983. *Signals and Systems Continuous and Discrete.* New York: Macmillan.

Appendix A

Given any sampled windowed signal w(n) with discrete Fourier transform W(f)

$$W(f) = \sum_{n=-\infty}^{\infty} w(n)e^{i2\pi fn}$$

consider the periodic signal w$'(n)$ created by adding delayed versions of w(n) together.

$$w'(n) = \sum_{m=-\infty}^{\infty} w(n + mD)$$

where D is the period length in samples and $1/D = f'$ is the fundamental frequency in w$'(n)$.

Then the Fourier transform W$'(f)$ of the periodic signal is

$$W'(f) = \sum_{n=-\infty}^{\infty} \left(\sum_{m=-\infty}^{\infty} w(n + mD) \right) e^{i2\pi fn}$$

$$= \sum_{n=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} w(n + mD)e^{i2\pi fn}$$

Letting $k = n + mD$

$$W'(f) = \sum_{m=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} w(k)e^{i2\pi f(k-mD)}$$

$$= \sum_{m=-\infty}^{\infty} \left(\sum_{k=-\infty}^{\infty} w(k)e^{i2\pi fk} \right) e^{-i2\pi fmD}$$

$$= \sum_{m=-\infty}^{\infty} W(f)e^{-i2\pi fmD}$$

$$= W(f) \sum_{m=-\infty}^{\infty} e^{-i2\pi fmD}$$

$$= W(f) \sum_{n=-\infty}^{\infty} \delta(f - nf')$$

where δ is the standard delta function.

Hence, the periodic signal w$'(n)$ has harmonic amplitudes W$'(nf')$ that are the same as the original windowed signal's amplitudes W(f), evaluated at the harmonic frequencies $f = nf'$. i.e.

$$W'(nf') = W(nf').$$