PAPER SUMMITTED TO WORKSHOP ON MULTIMEDIA TECHNOLOGIES

A SIMPLE MUSIC TRANSPOSITION METHOD FOR COMPUTERS

Shingchern D. You

Department of Computer Science and Information Engineering

National Taipei University of Technology

1, Sec. 3, Chung-hsiao East Road,

Taipei 104, Taiwan

Tel: +886-2-2771-2171 Ext 4234

Fax: +886-2-8773-2945

Email: you@csie.ntut.edu.tw

ABSTRACT

This paper presents a simple approach to implement the music transposition function in personal computers. This approach uses an inaccurate but simple method for sampling rate conversion and an overlap-add method for compensating the duration change due to different sampling rates. The approach, implemented by C on a Celeron 466 MHz PC, consumes less than 3% of CPU power.

Index terms: Music transposition, Sampling rate conversion, Overlap-add.

1. INTRODUCTION

The key-control function to adjust the pitch of music is almost a standard function in any karaoke-based amplifier or player. Despite the popularity of the function, papers discussing the implementation details were only a few [1]. In addition, commercial PC-based players are still not equipped with such a function. By using the limited references, we implement such a function suitable for players running on PC.

2. BACKGROUND

The key-control function in a karaoke-based machine is to shift (transpose) the key of a piece of recorded music up or down by a certain music intervals (semitones). The equal-tempered scale, used in pianos and other musical instruments, divides an octave into 12 intervals each containing a semitone. The frequency ratio between two consecutive semitones, e.g., C and $C^{\#}$, is a constant equal to $2^{(1/12)} = 1.059463$. Therefore, if we transpose all the notes of a piece of music up by one semitone, then we observe that the frequency components of the music are all shifted by a ratio of 1.059463. It is quite easy to implement frequency shift in discrete-time system. For example, if one second of 1 kHz sinusoidal signal is sampled at 3 ks/s and reconstructed by a D/A converter at 6 ks/s, then the reconstructed signal is a 2 kHz sinusoidal signal with a duration of one-half second, as shown in Figure 1. This example implies that if the sampling rate of the pre-recorded music is different from the reconstructed one, then music transposition occurs although the playback time of the music is different from the record time. Along with the change of the playback time, the tempo (speed) of the music is changed also. This situation is not desirable. Thus, the contents of the digitized music must be modified in order to keep the tempo. In addition, using a lower reconstruction rate than the sampling rate implies that aliasing effect may occur

during reconstruction, and a digital low-pass filter may be needed before reconstruction. Therefore, a key-control (transposition) system must be able to change the reconstruction rate and to modify the contents and the length of the sampled music. Figure 2 depicts the block diagram of such a system. The system performs the key shift by changing the reconstruction rate. To have the same tempo on the modified music, the length of the sampled-version of signal is also modified in the system. Such system may be embedded in a karaoke-based machine. However, this approach is not practical to be realized on a PC. One reason is that changing the sampling rate on a PC requires to flush the audio buffer and to reset the audio codec. This will produce an audible music pause (gap). The second, and more important, reason is that the conversion rates available to choose from are very limited on a PC, thus a required conversion rate may not be supported.

3. THE PROPOSED APPROACH

Based on the previous discussion, we know that the key-control system realized on a PC should use a fixed reconstruction rate. In addition, the available inputs to the system are PCM samples at pre-defined sampling rates, e.g., 44.1 ks/s or 48 ks/s, dependent on the signal sources. Under these conditions, we must change the sampling rate of the incoming PCM samples in digital domain so that the sampling and reconstruction rates are different. Accordingly, the change of sampling rate increases or decreases the number of PCM samples in a fixed time. This implies that the total playback time of the modified PCM samples is shortened or prolonged. Therefore the data length adjustment is also required. Combining these two parts, the proposed approach, shown in Figure 3, is suitable to be realized on a PC. Note that Figure 3 is for shifting up the music scale. To shift down the scale, we should perform decimation before low-pass filtering. We now describe the proposed approach in details in the following.

3.1. Sampling-rate conversion

The process of sampling-rate conversion is to change the sampling rate of the PCM samples so that the reconstruction rate can be fixed. Assuming that at first the pre-recorded music is sampled at 48 ks/s. The D/A converter on a PC is also set to 48 ks/s. To shift up the scale of the music by one semitone, we have to set the sampling rate of the music to

$$f_s' = f_s * r = 48 * 0.9438743 = 45.31 \text{ ks/s}$$

where r is a scale factor. If the re-sampled piece of music is reconstructed at 48 ks/s on a PC, the frequency scale of the music is shifted up by 48 / 45.31 = 1.0594, i.e., one semitone. On the other hand, shifting down the music by one semitone requires changing the sampling rate of the music to

$$f_s' = f_s * r = 48 * 1.059463 = 51.33 \text{ ks/s}$$

Tables 1 and 2 summarize the scale factor r and the corresponding target sampling rates for shifting up or down based on the sampling and reconstruction rates of 48 ks/s.

In the above discussion, we need to convert the sampling rate of the PCM samples from 48 ks/s to 45.31 ks/s. Widely used in digital signal processing, this type of conversion can be achieved in three steps. As shown in Figure 4, the first step of the conversion is to interpolate the samples by (M – 1). The second step is to pass the resultant samples through a low-pass filter, and the final step is to decimate the filtered samples by M. The value of M is calculated based on the scale factor r. For example, r = 0.9438743 is approximated by

$$0.9438743 \approx \frac{M-1}{M}$$

Therefore, an approximate value of M is M = 18. Tables 1 and 2 also contain the values of M for various scale factors r and the approximation errors. Efficient implementation of sampling-rate conversion is

available by using the polyphase Decomposition [2]. However, since the entire system is to be implemented on a PC, the computational load is a major concern. Besides that, a PC-based karaoke system is by no means to be a professional one. At least the quality of the reproduced sound is not very good due to the PC speakers. Considering these factors, we decided not to use the re-sampling approach mentioned previously. Instead, we used an imprecise, yet low-complexity, approach to perform the conversion as follows. To reduce the sampling rate, the PCM samples are filtered by a second-order IIR low-pass filter. One out of *N* filtered samples is then discarded. If the sampling rate is to be increased, one zero is inserted out of N samples. Then, the IIR filter mentioned previously is used to smooth out the zeros. Based on this method, we are unable to shift up or down the musical scale by more than 6 semitones. The subjective experiments showed that the quality of the modified music was mainly determined by the overlap-and-add procedure mentioned later, not the sampling-rate conversion.

3.2. Resize the PCM samples

The PCM samples after the sampling-rate conversion have a total length different from its originals. For the musical scales to be shifted up, we are in short of samples. For the scales to be shifted down, we have too many samples. Therefore, we need to add or discard samples according to the shift condition. Note that if samples are evenly added or discarded over the entire sample stream, then its effect is to undo the sampling-rate conversion. Therefore, the samples to be added or discarded should be on a block base. That is, a segment of samples in a block is to be added or discarded in one length-adjustment process. In order to smoothly add or discard samples in a block, we use the overlap-add (OLA) [3] technique. The following gives a brief explanation of adding samples. At first, the input samples are converted to the desired sampling rate. The re-sampled results are stored in a buffer, and the OLA procedure is performed on a block-base with each block containing $N = 2048$, samples. Along with the sampling rate change, the number of samples discarded, N_d, is also calculated. Once a block of samples is collected, we then expect

to add N_d samples at the end of block to keep the tempo of the music unchanged. As shown in Figure 5, the block with additional samples are obtained by putting together the original block and a replica of the original block through the OLA operation. The window function used in the operation is a simple triangular window with height of 1 and length of $W = 2048$. Note that only one-half the number of the window length, i.e., 1024, samples are to be added together. The place in a block where the OLA operation is performed greatly affects the reproduced sound quality. Fewer artifacts will be detected if the similarity is high between the original and the replica in the region performing OLA. Therefore, we should calculate individual similarities for a range of samples and choose the one having highest similarity as the beginning point. The target number of samples N_t to be added is

$$N_t = \min (N_d , (6/14) * W)$$

The OLA procedure is postponed to next block if the value of N_t is smaller than a pre-determined value, say 125. Since the exact place to perform OLA varies from one block to another, this number is used to find the range where similarities should be measured. Let the samples of the block be numbered from 1 to N. The search for the best-fit point is from the sample $N_1 = (N_t - N_0)$ to $N_2 = (N_t + N_0)$. The value of N_0 is chosen to be 100. The similarity is measured by correlation. High correlation means high similarity. Specifically, similarity S_k for starting point at sample k with $N_1 \leqq k \leqq N_2$ is calculated as

$$S_k = \sum_{n=k}^{k+W/2} s[n]s[n-k]$$

When S_k is found, the highest similarity value S_{km} and its corresponding index km can be obtained. The OLA procedure is then applied to the block, i.e.,

$$y[n] = s[n] \bullet w[n] + s[n+km] \bullet w[n+W/2], \quad n = 1 \ldots W/2$$

where $y[n]$ holds the results of the OLA. With the overlap-add procedure, the number of samples in the block becomes $(N + km - 1)$ samples. Then, N_d is updated accordingly to reflect the true number of samples to be added in the following block. During the correlation calculation, to increase the

computation speed, index *k* increments by 3 instead of 1. This greatly reduces the computational burden of the correlation computation. The procedure for deleting samples is done in a similar procedure.

4. THE RESULTS

The proposed approach was implemented in a PC running a Linux OS. The CPU is a Celeron 466. The program, written in C, was implemented as a filter program. The input to the program was PCM samples and the output from the program was the altered PCM samples. A simple GUI was also implemented to select the desirable key shift value. Several different kinds of music were used to test the performance of the proposed approach. The subjective experiments indicated that the sound quality was acceptable for most music for the entire key-control range. The only defect is that when key control was set to +/- 1, sometimes we could hear trembling in female-vocal music. Overall speaking, the reproduced sound quality is adequate for the PC-based applications. Also the reproduced music has no obvious playback time difference between the original and the reproduced one. This confirms that the adaptive selection of overlapping area is effective in adjusting the music length.

5. CONCLUSION

We have presented a low-complexity music transposition method suitable for PC implementation. The proposed approach uses a non-precise, yet simple, sampling-rate conversion, and it is followed by an overlap-and-add scheme to adjust the duration of the transposed music. The actual implementation confirms that the performance is adequate and the computational load is light.

6. REFERENCES

[1] B. Lawlor and A. D. Fagan, “A Novel Efficient Algorithm for Music Transposition,” *Proceedings of the IEEE EUROMICRO*, pp. 48-54, 1999.

[2] G. Zelniker and F. Taylor, Advanced Digital Signal Processing: Theory and Applications, chapter 7, Marcel Dekker, New York, 1994.

[3] H. Sanneck, et al., "A New Technique for Audio Packet Loss Concealment," *Proceedings of the IEEE GLOBECOM*, pp. 48- 52, 1996.

#	$r = f_s' / f_s$	r * 48 (Ks/s)	M	(M -1)/ M * 48 (Ks/s)	Error %
1	0.9438743	45.31	18	45.33	0.07 %
2	0.8909001	42.76	9	42.67	0.21 %
3	0.8408964	40.36	6	40	0.89 %
4	0.7937005	38.10	5	38.4	0.79 %
5	0.7491535	35.96	4	36	0.11 %
6	0.7071067	33.94	3	32	5.72 %

Table. 1. The shift-up scale and the corresponding re-sampling factor M.

b	$r = f_s' / f_s$	r * 48 (Ks/s)	M	(M +1)/ M * 48 (ks/s)	Error %
1	1.0594361	50.85	17	50.82	0.05 %
2	1.1224602	53.88	8	54	0.22 %
3	1.1892071	57.08	5	57.6	0.91 %
4	1.259921	60.48	4	60	0.79 %
5	1.3348399	64.07	3	64	0.11 %
6	1.4142136	67.88	2	72	6.07 %

Table. 2. The shift-down scale and the corresponding re-sampling factor M.

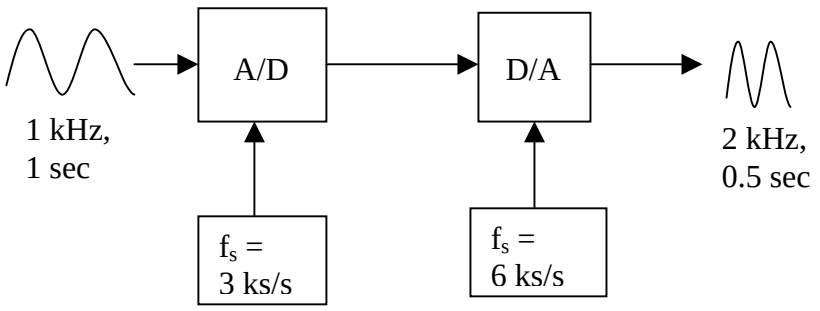


Figure 1. A simple system for music transposition.

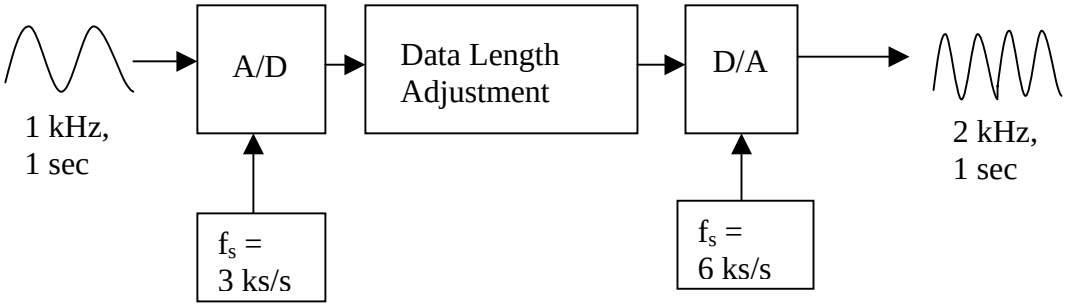


Figure 2. A practical system for music transposition.

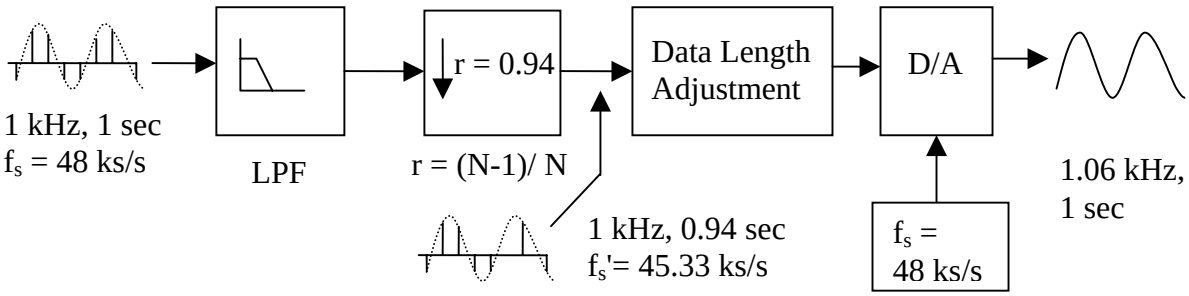


Figure 3. The proposed approach.

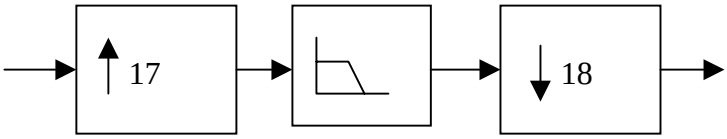
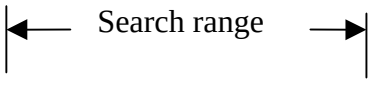


Figure 4. The sampling-rate conversion.



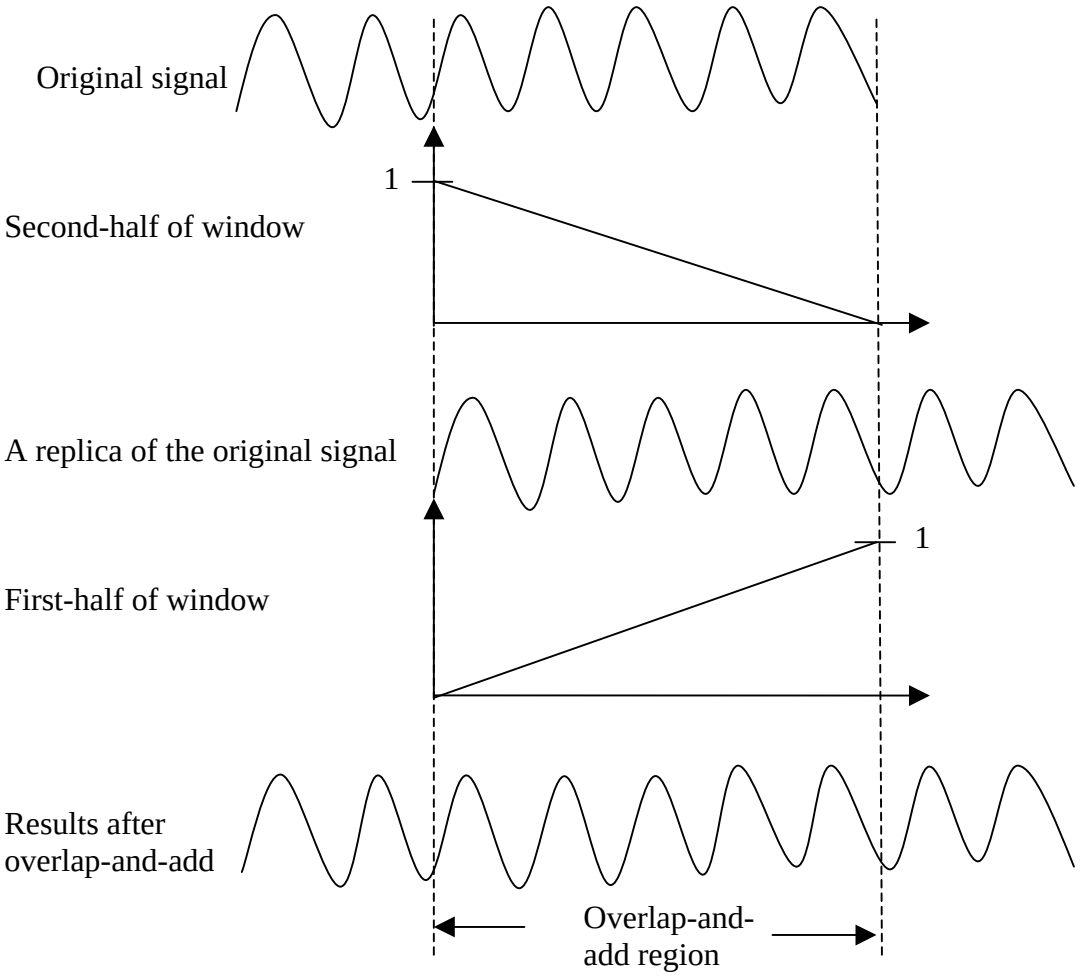


Figure 5. The overlap-add operation to increase the data length.