A Novel Efficient Algorithm for Music Transposition

Bob Lawlor* and A. D. Fagan**
*Dublin Institute of Technology[1]
**University College Dublin

Abstract

We present a novel efficient algorithm for scaling the frequency content of an audio signal by any desired factor in the range 0.5 (minus one octave) to 2.0 (plus one octave) enabling a recording to be played in any desired key without affecting the tempo. The algorithm uses an adaptive overlap-add (AOLA) technique to realise the desired frequency scaling without affecting the duration. Informal listening tests show output quality equal to that of a conventional overlap-add algorithm used in many commercially available systems, but offering significant computational saving relative to that algorithm. The algorithm is also used to simultaneously scale both the tempo and key of a recording.

1. Introduction

Transposing a piece of recorded music consists of scaling its frequency content by a fixed multiplicative factor. Such a modification allows the piece of music to be played back in a different key without affecting the tempo (speed). This can be useful, for example, in situations where a musicianship student (instrumental or vocal) wishes to practice along with a previously recorded piece (instrumental, orchestral and/or choral part(s)) in a key of their choice. The tempo and key of a recording are generally set at those intended by the composer. Often, only the most proficient musicians can play a piece at its intended tempo and in its intended key. In the case of a vocalist, the intended key may not suit their voice. Hence, a facility to independently control both the tempo and key of a recording is of potential benefit to both aspiring instrumental musicians and singers. In recent years a wide range of products have become available with features to enable independent control of tempo and key in the playback of an audio recording. A comprehensive list of all such products would be exhaustive, however, some widely used such products include the Entropic Timescale Modification (ETSM™) [1], Goldwave [2], Cooledit [3], Transcriber [4], Musician's CD player [5], SlowGold [6]. A more extensive list can be found at [7]. Being commercial products, the technical details of the time- and frequency-scaling algorithms used are generally not disclosed. However, from the limited data available it appears that most of these products realise their scaling using methods based on a time-domain process called the Synchronised Overlap-Add (SOLA) algorithm [8]. For example, the ETSM system, which performs extremely well on speech signals, uses a so-called Time-Domain Pitch Synchronous Overlap-Add (TD-PSOLA) algorithm. Sample outputs from this system can be downloaded from the Entropic URL [1].

2. Background

2.1 Frequency Perception

The term *pitch* is often used in the contexts of speech and music. The definition of pitch, however, is context dependent. For example, in the case of voiced speech, pitch refers to the frequency with which the vocal cords open and close. For music, however, the definition of pitch is based on the frequency perception properties of the human auditory system. For a musical instrument, pitch is related (but not necessarily equal) to the fundamental frequency of the note being played. The pitch of a pure tone is directly related to its frequency. We perceive a low-frequency tone as having low pitch and a high-frequency tone as having high pitch. However, psychoacoustic experiment shows that our judgement of the pitch of a pure tone is not linearly related to its frequency. Other non-linear processes are also involved such as the masking of some frequency components by others. For these reasons the performance of a frequency-scaling algorithm is best evaluated by subjective listening tests. Audio signal processing algorithms such as perceptual coders are designed to take advantage of known properties of the human auditory system. A popular psychoacoustic experiment involves presenting a listener with a pure tone of frequency f_1 and having them adjust the frequency, f_2, of a second tone until its pitch is judged to be equal to half that of f_1. For f_1 = 440 Hz, the average setting of f_2 is equal to 220 Hz, indicating that for low frequencies a halving in frequency corresponds to a halving in pitch. However, for f_1 = 8 kHz, the average

[1] This work was funded by the Dublin Institute of Technology.

setting of f_2 is equal to 1300 Hz. Pitch determined by comparing two tones in this way is called *ratio pitch*. Because ratio pitch is related to our sensation of *mel*odies, it was assigned the dimension *mel*. Therefore, a pure tone of 125 Hz has a ratio pitch of 125 mel and the tuning standard, 440 Hz has a ratio pitch of 440 mel. For frequencies below 500 Hz, the ratio pitch in mels is numerically equal to the frequency in Hertz. At higher frequencies, however, the curve bends more and more to reach a ratio pitch of only 2400 mels at a frequency of 16000 Hz. The *mel-scale spectrogram* is based on using a parallel filterbank to resolve the frequency content of a signal into mels and has been used in the analysis of speech and other audio signals for many years. There are two theories relating to the perception of pitch, namely, the *timing (or temporal) theory* and the *place theory* [9]. The timing theory suggests that pitch is perceived in terms of time-synchronous firings of the neurons connected to the haircells near the basilar membrane apex. The place theory suggests that spectral information is decoded via the basilar membrane locations of the neurons that fire most. The two theories warrant a distinction between two types of pitch, namely, *normal pitch* and *spectral pitch*. Normal pitch corresponds to the fundamental frequency of a sound and spectral pitch corresponds to the frequency distribution of the sound energy among the overtones or harmonics of the fundamental frequency component. Normal pitch is often referred to as *residue pitch* or *virtual pitch* as it can be perceived even in the absence of its fundamental harmonic. It is believed that the eight lowest harmonics influence the perception of pitch. The timing theory is limited to low and middle frequencies because the synchronisation of neural spikes to tonal inputs disappears above 4-5 kHz due to neural firing latency effects. A shortcoming of the place theory is that it cannot explain the high pitch resolution of the ear at low frequencies. It is believed that both processes operate in parallel, with one or the other dominant depending on the frequency content and type of sound. The pitch of pure tones depends not only on frequency, but also on other parameters such as the intensity, which is generally expressed as Sound Pressure Level (SPL) [9]. For example, if a 200 Hz tone is presented at 80 dB(SPL) and 40 dB(SPL) alternately, the louder tone is judged as having slightly lower pitch than the softer one. However, at 6 kHz this effect is reversed i.e. a 6 kHz 80 dB(SPL) tone produces a slightly higher pitch than a 6 kHz 40 dB(SPL) tone. Hence, although frequency is the major cue in pitch perception, it is also influenced to a small extent by sound intensity.

2.2 Musical Scales

For thousands of years people have expressed themselves through music as well as speech. Many totally different languages evolved with different civilisations. In a similar way, many different musical scales came into being. However, closer inspection of seemingly different scales reveals characteristics common to almost all scales, suggesting perhaps that music is a natural form of expression. The oldest and most popular musical instrument is the human voice. The Greeks used the *octave* (a frequency ratio of two), whereby a low voice was accompanied by a second voice, one octave above it. In musical nomenclature, a ratio of two pure frequencies or *tones* is called an *interval*. Certain specific intervals are given names, e.g. the ratio 3/2 is called a *fifth* and the ratio 4/3 is called a *fourth*. Helmholtz [10] notes that '*…unpractised singers, when they wish to join in the chorus to a song that does not suit the compass of their voice, often take a fifth to it*'. He cites this naturally occurring phenomenon as '*proof that the uncultivated ear regards repetition in the fifth as natural*.' Helmholtz also notes that '*such an accompaniment in the fifth and fourth is said to have been systematically developed in the early part of the middle ages*.' This systematic development formed the basis of much modern music theory. An important observation of this development was the splitting of the octave into two equivalent sections called *tetrachords* (meaning *four strings* in Greek). The tetrachord is defined as a series of four notes having an interval of a fourth between the first and last, i.e. the ratio of the highest to the lowest frequency notes in the tetrachord is 4/3. For the modern major scale, the two tetrachords are as shown in Figure 1.

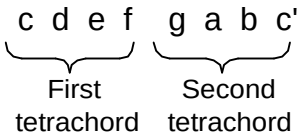

Figure 1: Major Scale Tetrachords

Each note in the second tetrachord is one fifth (ratio of 3/2) above the corresponding note in the first tetrachord. This division of the octave into two equivalent sections is common to almost all scales from different civilisations and historical periods. The choice of interim notes within (and also possibly between) the two tetrachords, however, varies and many different scales based on different such choices exist.

Much of the early work on musical scales is credited to the Greek philosopher and mathematician, Pythagoras (c.569 – c.475 BC). He used a monochord (a hollow box with a single string stretched between two supports near the ends as shown in Figure 2. A third support, called a

bridge, could be moved to any point in between, dividing the two string segments into any desired ratio (l_1/l_2).

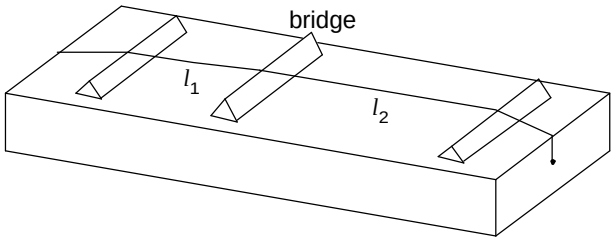

Figure 2: Monochord

He studied the sounds produced by striking one segment and noticed that the frequency of the sound was inversely proportional to the length of the vibrating segment. He also noticed that by striking both segments simultaneously, the sensation produced by the resulting sound varied between pleasant and unpleasant depending on the position of the bridge. The sound was pleasant when the ratio of the segment lengths, (l_1/l_2), was equal to the ratio of two small integers. This means that the ratio of the two frequencies is also equal to the ratio of two small integers (equal to the reciprocal of the string length ratio, i.e. l_2/l_1). This condition is called *consonance* or *harmony*. He also noticed that the unpleasant sounds were associated with frequency ratios equal to the ratio of large integers, a condition called *dissonance* or *disharmony*. Deciding whether a sound is pleasant or not is a subjective issue and there is no clearly defined transition between consonance and dissonance. Pythagoras chose a set of frequency ratios (intervals) of the form $3^n/2^m$ where n and m are integers. For example, the *Pythagorean whole tone* is equal to $9/8 = 3^2/2^3$ and the so-called *Pythagorean diatonic semitone* is equal to $256/243 = 2^8/3^5$. Note that the choice $3^n/2^m$ covers any combination of multiple fifths and fourths. The so-called *major scale* is defined by the intervals: tone, tone, semitone, tone, tone, tone, semitone. Using the Pythagorean whole tone and the Pythagorean diatonic semitone, the Pythagorean C major scale results, as shown in Table 1.

c	d	e	f	g	a	b	c′
1	$\frac{3^2}{2^3}$	$\frac{3^4}{2^6}$	$\frac{4}{3}$	$\frac{3}{2}$	$\frac{3^3}{2^4}$	$\frac{3^5}{2^7}$	2

Table 1: The Pythagorean C Major Scale

In Table 1, we express the intervals or ratios in a form which emphasises the fact that the second tetrachord notes (g, a, b, c′) are a fifth above their first tetrachord counterparts (c, d, e, f), i.e. g = (3/2)c, a = (3/2)d etc. Other musical scales developed in various parts of the world. In many of these early scales, the use of intervals of less than a tone was avoided, which resulted in a variety of scales based on intervals of a tone and a tone and a half. These scales generally had five intervals per octave and as such are often referred to as *pentatonic* or *five-tone* scales. An example of such a pentatonic scale is the major scale with the fourth and seventh notes missing. Much Arabian music is based on dividing the octave into 16 or 17 unequal intervals. Hindu music has 22 divisions per octave. In the sixteenth century an Italian mathematician named Zarlino, modified the Pythagorean scale by replacing the notes based on large integer ratios by nearby ones based on smaller integer ratios as shown in Table 2. These modifications formed the basis of the so-called *just diatonic scale*.

c	d	e	f	g	a	b	c′
1	$\frac{9}{8}$	$\frac{5}{4}$	$\frac{4}{3}$	$\frac{3}{2}$	$\frac{5}{3}$	$\frac{15}{8}$	2

Table 2: The Just Diatonic (C Major) Scale

Note that the intervals between consecutive notes of the just diatonic scale are equal to 9/8 (called a *just major tone*); 10/9 (called a *just minor tone*) and 16/15 (called a *just semitone*). The ratio of a just minor tone to a just semitone is equal to $10/9 \div 16/15 = 25/24$. The *chromatic scale* is derived from the just diatonic scale by including the notes which are spaced by the interval 25/24 above and below (ratio = 24/25) the existing notes. A problem with the chromatic scale however, is that for example C# and Db are close but not quite equal. The same applies to D# and Eb etc. The *equal-tempered scale* (often called the *tempered scale*) is based on a simplification of the chromatic scale whereby C# is set equal to Db and D# = Eb etc. This is achieved by dividing the octave into 12 equal intervals. If the frequency ratio associated with any one of these intervals is equal to some value, r, then r must satisfy the equation $r^{12} = 2$, therefore, $r = 2^{(1/12)} = 1.059463$. This interval is called the *tempered semitone*. The Modern piano is constructed and tuned to give seven octaves of the tempered scale. Orchestral instruments are also constructed and tuned to the tempered scale, although most allow less than seven octaves, e.g. the concert flute gives three tempered octaves. There are two major reasons for constructing and tuning instruments to the tempered scale. Firstly, sharps and flats are combined in a single note. Secondly, (and more importantly for our application) the equal interval between all consecutive notes means that a piece of music can be played in any key, i.e. it can be transposed up or down in frequency by a desired number of tempered semitones without affecting the consonance of the chords (or harmonies) in the piece.

This means, for example, that a piano accompaniment can be played in a key to suit a person's singing voice. Our objective for the music key transposition application is to develop an algorithm to enable the frequencies of a piece of recorded music to be scaled to any other key on the tempered scale within one octave of the original key.

2.3 Musical Instrument Digital Interface (MIDI)

Musical Instrument Digital Interface (MIDI) was developed in the early 80's as a communication protocol to allow electronic musical instruments (synthesisers) to interact with each other. In its original form, MIDI allowed two synthesisers to be connected together such that a note played on one could sound on the other also (as if it had been played on both). The MIDI information consists of data bytes which control such things as when to start and stop playing a note, how loud to play it etc. The format of the MIDI information was well suited to storage and generation on a computer and computer manufacturers quickly developed MIDI plug-in boards for their products. Nowadays, a computer with such a MIDI interface can control a chain of synthesisers. Up to sixteen synthesisers can simultaneously play different tracks (or parts) in this way. MIDI files can also be played on a standard multimedia PC through the use of a software synthesiser [11]. The quality of the reproduction depends on the sound card and synthesiser program. Many programs are available which can produce good quality renditions of MIDI files and also convert them to other audio file standards such as .WAV files [12]. Within a MIDI file, the data relating to note frequency and duration are stored separately so that it is a trivial matter for a MIDI file player to play the piece slower or faster without changing the key. Similarly, the piece can be played in a different key without changing the original tempo, or if required, both the tempo and the key can be independently set to any desired values. One system which uses this feature of MIDI as a music study aid is the Maestro system [13]. Another useful feature of MIDI is that one or more of the sixteen tracks in a MIDI file can be switched off (muted). For example, the lead instrument could be muted leaving only the accompaniment which the practising musician can play (or sing) along with at their desired tempo and in their desired key. A shortcoming of MIDI is that, being synthesiser input data, it cannot reproduce a human voice sound. Also, no synthesiser to date comes close to rendering a piece of instrumental music with the 'feeling' of an accomplished musician. Narrowing this gap remains a major challenge. Nonetheless, with the increasing availability of low cost multimedia PC's, software synthesisers and MIDI data files [14], aspiring musicians can enhance their practice and performance with MIDI.

3. TIME & FREQUENCY SCALING

If a signal is sampled using sample frequency, f_s samples per second, and then played back using a different conversion frequency, f_p, the duration of the signal will be scaled by the factor, f_s/f_p and the frequency content of the signal will be scaled by the factor f_p/f_s. Time-Scale Modification (TSM) of an audio signal consists of modifying its duration without affecting its perceived frequency content. Similarly, Frequency-Scale Modification (FSM) consists of modifying its frequency content without affecting its duration. In 1985 Roucos [8] presented the Synchronised Overlap-and-Add (SOLA) algorithm for speech TSM. With this approach, overlapping segments (or frames) of the input signal are first extracted, a frame being typically several pitch periods in duration. By decreasing the overlap between successive frames, time-scale expansion is realised. Similarly, by increasing the overlap, time-scale compression is realised. In the original SOLA algorithm [8], the segment alignment was optimised by computing a normalised cross-correlation measure, R, for a range of possible alignment offsets and then choosing the offset for which R is a maximum, indicating maximal similarity between overlapping segments. High quality combined with moderate computational load has made the SOLA algorithm the choice for many speech and audio TSM systems. We present an alternative TSM algorithm which offers a significant reduction in computational load without loss of quality. If a signal is time-scaled by some factor, T_S, and then played at T_S times its original sample rate, the net effect is to scale the frequency content by the factor, T_S, without affecting the original duration, i.e. FSM. We use this approach to realise frequency-scale modification.

3.1 Adaptive Overlap-Add (AOLA)

Referring to Figure. 3, the solid trace of plot (a) represents a rectangular windowed segment of the input signal. The window length, w, is chosen such that it will accommodate at least two cycles of the fundamental frequency of harmonic music. For non-harmonic music, the choice is not critical and w can be left equal to the value chosen to satisfy the above harmonic condition. We used $w \approx 46$ ms. Assuming we wish to scale the duration of this segment by some desired expansion factor, de, the steps involved in the algorithm are as follows:

1. The windowed input segment (a) is duplicated and the duplicate aligned with (a) as shown in plot (b). The alignment criterion is based on aligning the two largest peaks or troughs.

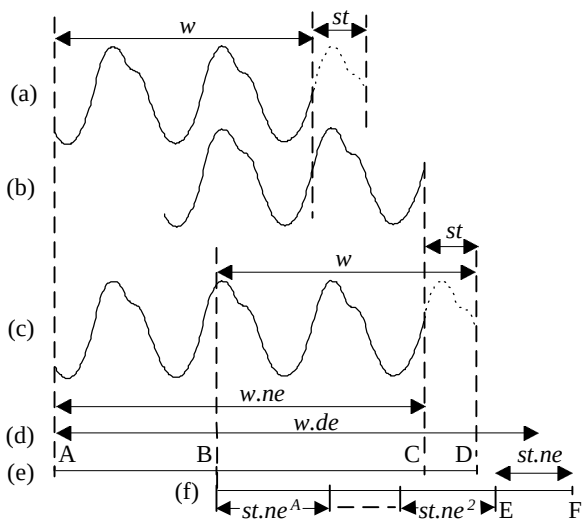

Figure 3: Adaptive Overlap-Add

2. A synthetic segment, (c), is produced by fading gradually from (a) to (b) in the overlapping region. The natural expansion factor, *ne*, is given by the ratio of the lengths of (c) and (a) as indicated.
3. The rectangular window is stepped forward in time by $st = |CD| = w.(1\text{-}ne)/(1\text{-}de)$ and the new step-size segment of the original concatenated with (c) as indicated (dotted) such that the next segment to be expanded is the length-*w* portion of (c) above BD, see plot (e). Repeat from step 1 until the end of the signal being scaled is reached.

***Rationale**:* Plot (d) represents the desired length to which we wish to time-scale (a), i.e. *w.de*. The segment of (c) above AB has been time-scaled by the desired factor, *de*, and is output from the time-scaling window. For each step of the window we repeat the peak search and update *ne*.
Assuming *ne* to be approximately equal to its last value, segment BD of (c) expands in the same way as (a) to length $|BF| \approx w.ne$. Step size portion CD of (c) expands to length $|EF| \approx |CD|.ne$, but we require it to be expanded by factor *de*. To achieve this we must apply our natural expansion factor *A* times where $ne^A = de$. If segment CD is to have *A* applications of natural expansion factor, *ne*, before leaving the expansion window, then from plot (f), the step size, *st*, must satisfy the following equation

$$st.ne + st.ne^2 + \cdots + st.ne^A \approx w.ne \qquad (1)$$

$$\Rightarrow \quad st \approx w.\frac{1-ne}{1-ne^A} = w.\frac{1-ne}{1-de} \qquad (2)$$

As *ne* is continuously varying, (1) (and (2)) is an approximation. In fact, each of the *ne* and *st* terms in (1) are slightly different. By updating *ne* and hence *st* for every step of the window, the algorithm accurately adapts to the local signal characteristics.
For time-scale compression the approach is similar. In this case the peaks or troughs are aligned as before but the sections of (c) to the left and right of the central overlapping section are discarded leaving a naturally compressed segment. If the input segment has a natural compression factor, *nc*, and the desired compression factor is *dc*, then equation (2) becomes

$$st = w.\frac{1-nc}{1-dc} \qquad (3)$$

For simultaneous time- and frequency-scaling by factors, T_S and F_S respectively, the procedure is a simple modification of the above frequency-scaling procedure. The frequency-scaling by factor F_S is realised by playing the signal at F_S times its original recording rate. This however, scales the duration by factor $1/F_S$ which must be compensated for by the AOLA TSM. That is, the necessary TSM factor is T_S times F_S.

4. SOLA vs AOLA Comparison

4.1 Computational Load Comparison

In the original SOLA algorithm [8], the segment alignment was optimised for the *m*-th input frame by computing a normalised cross-correlation measure $R_m(k)$ for a range of possible alignment offsets, *k*, and then choosing the offset, K_m, for which $R_m(k)$ is a maximum, indicating maximal similarity between overlapping segments. The normalised cross-correlation measure $R_m(k)$ can be expressed as :

$$R_m(k) = \frac{\sum_{j=0}^{L-1} y(mS_s + k + j)x(mS_a + j)}{\sqrt{\sum_{j=0}^{L-1} x^2(mS_a + j)\sum_{j=0}^{L-1} y^2(mS_s + k + j)}} \qquad (4)$$

In (4), $x(n)$ represents the sampled signal to be time-scaled and $y(n)$ the time-scaled signal. S_a represents the analysis interframe interval or step-size and S_s, the synthesis step-size. If the time-scale factor is a, then $S_s = aS_a$. *L* represents the length of the overlapping portions of $x(n)$ and $y(n)$. Clearly the value of *L* varies with *k*. This variable overlap is one reason why the normalization denominator term in (4) is needed.

The normalization also prevents loud portions of the signal from producing deceptively large cross-correlation measures relative to those produced during quiet portions. In the original system of Roucos, $R_m(k)$ was computed for offset values, k, in the range 20 to 130 samples. The computational load associated with the SOLA algorithm is due mainly to the multiple computations of the $R_m(k)$ term. A useful simplificationoften used to reduce this computational load is to use sum-of-magnitude terms in the denominator, i.e. to use a simplified normalized cross-correlation measure $R'_m(k)$ given by:

$$R'_m(k) = \frac{\sum_{j=0}^{L-1} y(mS_s + k + j)x(mS_a + j)}{\sum_{j=0}^{L-1}|x(mS_a + j)|\sum_{j=0}^{L-1}|y(mS_s + k + j)|} \quad (5)$$

Also, in practice, an FFT-based fast convolution process [15] is used in computing the numerator of (5) reducing the number of multiply operations. For the purpose of calculating an approximate computational load estimate, we make two simplifying assumptions, namely, that the analysis frame step-size has a default value of $N/2$ and that the search range of alignment offset, k, is from $k = 0$ to $k = N/2$, where N is the number of samples in the analysis frame. Based on these assumptions it can be shown [16] that the computational load associated with SOLA time-scale expansion ($a > 1$) is as shown in Table 3 and that the load associated with SOLA time-scale compression ($a < 1$) is as shown in Table 4.

N	SOLA	AOLA	A/S (%)
Mult.s	$aN(\log_2 N+3/2)$	$aN/2$	*5.3*
Add.s	$aN((3/2)\log_2 N+3/4)-2)$	$aN/4$	*2.0*
Comp.s	$aN/2$	$a2N$	*400*

Table 3: Computational load comparison ($a > 1$)

N	SOLA	AOLA	A/S (%)
Mult.s	$aN(\log_2 N+5/2)$	$aN/2$	*4.8*
Add.s	$aN((3/2)\log_2 N+13/4)-1)$	$aN/4$	*1.6*
Comp.s	$aN/2$	$a2N$	*400*

Table 4: Computational load comparison ($a < 1$)

To compute a rough estimate of the AOLA computational load we assume the window length, w, corresponds to the SOLA frame length, N, (sample periods) and the number of multiply and add operations needed to realise the overlap-add cross-fade are the same as for SOLA. The number of comparisons needed to find the two largest peaks or troughs within the N-sample frame is equal to $2N$ and this is further weighted by the TSM factor, a. The key saving is achieved by replacing the normalized cross-correlation multiply and add operations with an additional $3N/2$ comparisons. Tables 3 and 4 also show the approximate relative computational burden of the AOLA method compared to the SOLA method. The comparative measures are shown as functions of the segment length, N and the TSM factor a. The right-hand column shows the ratio of the AOLA to SOLA computational load for a frame length, N = 2048 sample periods.

5. Results

5.1 AOLA vs SOLA Sound Quality Comparison

We applied the AOLA algorithm to a selection of music signals for TSM factors in the range 0.5 to 2.0 (corresponding to FSM factors in the range –1 to +1 octave). We also used a commercially available SOLA-based TSM system to realise equivalent time-scale and frequency-scale modifications. The sound quality of the AOLA outputs were deemed equal to the SOLA-based system outputs for both radio- and CD-quality audio.

5.2 AOLA vs MIDI Comparison

We recorded a professional pianist playing a short piece in its intended key (D) and at its intended tempo (120 beats per minute (bpm)). We also recorded it being played in D at tempos 60, 90, 150 and 180 bpm and at 120 bpm in keys Bb (-4 semitones), C (-2 semitones), E (+2 semitones) and G (+5 semitones). We applied the AOLA algorithm to the 120 bpm, D recording to realise equivalent time- and frequency-scaled versions of the original. The AOLA outputs corresponding to 90 and 150 bpm and keys C and E were deemed almost equal to the corresponding originals while the outputs corresponding to 60 and 180 bpm and keys Bb and G were deemed good but inferior to the corresponding originals.

6. Discussion

In the study of musicianship, the ability to independently control the tempo and/or key of a recording can encourage and enhance a practice session. Musical Instrument Digital Interface (MIDI) allows such control for instrumental music, provided the recording is available in MIDI file format and that the music student has access to a suitable synthesizer or multimedia PC. In many cases the recording is in some digital audio format other than MIDI, such as a CD or WAV file. Converting from a digital audio format such as WAV to MIDI is very difficult. For example, [17] attempts WAV to MIDI

conversion for instrumental recordings, but requires thirty minutes to convert one second of the input WAV file on a 133 MHz Pentium with 32 Mbytes of RAM. It also requires 200 Mbytes of hard disk space and only works on mono digital audio at sample rates up to 22050 samples per second. We have presented an algorithm which allows efficient time-scale (tempo) and frequency-scale (key) modification of a digital audio signal (instrumental and/or voice). This algorithm uses a fixed length rectangular stepping window and a simple peak alignment criterion to track the local natural scaling factor and adapt the window step size. The desired TSM factor is realised by the appropriate number of applications of the constantly varying local natural scaling factor. The local natural scaling factor estimate is updated at sub-pitch period intervals giving accurate pitch tracking and high quality in the output scaled signal.

7. Conclusion & Further Work

In the future, the home hi-fi unit will interface to the home PC and MIDI files will be widely available. However, if an aspiring musician wants to accompany a favorite CD track at their desired tempo and key, a real-time high quality time-scale / frequency-scale modification algorithm will be needed. We have presented the basis of such an algorithm. In the future we plan to investigate further refinements to the algorithm such as applying it on a subband basis for improved quality. We also plan to investigate its real-time implementation.

8. References

[1] http://www.entropic.com/products/etsm/etsm.html
[2] http://www.goldwave.com/
[3] http://www.syntrillium.com/cep/index.html
[4] http://www.janetdavismusic.com/trnskr.html
[5] http://www.ronimusic.com/muscdpl.html
[6] http://www.worldwidewoodshed.com/
[7] Pages 2 and 3 of http://www.justjazz.com/manuscripts/transcribe.pdf
[8] Roucos, S. and Wilgus, A. M.: 'High-quality time-scale modification for speech'. IEEE proceedings on acoustics, speech and signal processing, March 1985.
[9] David M. Howard and James Angus: *Acoustics and Psychoacoustics*. Focal press. 1998
[10] Hermann Helmholtz: *On the Sensations of Tone*. Original German Edition 1885. Dover pub. (translation), 1954.
[11] http://www.yamaha-xg.com/english/xg/s-synth/s-synth.html
[12] http://www.eden.com/~mitchell/index.html
[13] http://www.everythingmusic.com/
[14] http://www.midifarm.com/
[15] Burrus, C. S. and Parks, T. W.: *DFT/FFT and Convolution Algorithms and Implementation*. John Wiley & Sons, 1985
[16] B. Lawlor and A. D. Fagan: 'A Novel Efficient Algorithm for Audio Time-Scale Modification', Irish Signals & Systems Conference, NUI Galway, 1999.
[17] http://www.hitsquad.com/smm/programs/AMA_Win95/