ICALIP 2008 - Proc. of the IEEE International Conference on Audio, Language and Image Processing, Shanghai, China; pp. 29 - 35.

Low Latency Audio Pitch Shifting in the Time Domain

Nicolas Juillerat, Simon Schubiger-Banz
Native Systems Group, Institute of Computer Systems, ETH Zurich, Switzerland.
{nicolas.juillerat|simon.schubiger}@inf.ethz.ch

Stefan Müller Arisona
Media Arts & Technology, University of Santa Barbara, USA.
sma@mat.ucsb.edu

Abstract

This paper presents a novel pitch shifting algorithm that achieves latency significantly lower than existing techniques. As such, it provides a solution for various situations such as live performances in which pitch shifting within less than 10ms of latency is desired. Based on IIR filters, oscillators and ring modulations, the proposed algorithm completely differs from previous approaches. This paper presents this novel algorithm in detail and shows how it was implemented. It further compares it to other pitch shifting techniques in terms of latency.

1. Introduction

Audio pitch shifting (also known as frequency scaling) consists in changing the pitch of an audio signal without affecting its speed. This is an old problem on which extensive research has been done. Various techniques have been proposed such as the phase vocoder [15, 27], time-domain overlap-add [7], and sinus+transient+noise modeling [20, 21]. Each technique has been further improved over the years to increase the quality of the result and to deal with various artifacts [9, 19, 23, 26].

Less research has been targeted to latency though. Latency is a major issue in live systems in which an instrument or voice is recorded, pitch shifted and played back in real-time. While latencies as high as *50ms* are acceptable for melodic instruments such as pipes or strings, previous work has reported that latencies as low as *10ms* are required for instruments with sharp attacks or drums [10]. With higher latencies, the delay prevents the player from synchronizing with the rest of the performance.

This paper describes a new pitch shifting algorithm that allows latencies below *10ms* to be achieved. The algorithm works entirely in the time domain, yet it does *not* use the block-based approach of existing time domain algorithms such as synchronous overlap-add (SOLA) [7, 26], but uses a *subband approach* instead. A property of the presented algorithm is that is does not require any pre-buffering of the signal and basically works sample by sample. This differs from both existing time domain block-based approaches and frequency domain approaches such as phase vocoders, that both have to pre-buffer the signal into frames before processing it.

The remainder of this paper is structured as follows: Section 2 gives an overview of related work. Section 3 briefly introduces *frequency shifting*, a transformation that differs from pitch shifting but that is central to the implementation of the presented new algorithm. The idea of the algorithm is presented in section 4 and detailed implementation issues are discussed in section 5. Section 6 summarizes experimental results and section 7 highlights possible future works.

2. Current approaches

While many pitch shifting algorithms and several variations of them have been proposed, latency has been given little attention, although research exists in this area concerning digital audio effects in general [1, 4, 14].

The latency of algorithms based on the short time Fourier transform (STFT) such as the phase vocoder directly depends on the size of the Fourier transform. With typical values ranging from *2048* to *8192* at *44.1 kHz*, latencies between *46* and *186ms* are achieved [1]. For simple audio signals such as speech, a Fourier transform size as low as *1024* can be used, yielding *23ms* of latency. Smaller values result in too much deterioration of the signal and are considered of insufficient quality in practice [28].

The latency of time domain block-based algorithms such as SOLA approaches is determined by the size of the blocks as well as various other factors that depend on the actual approach and implementation. Resulting

latencies are similar to those of STFT-based approaches [7, 29].

Regarding advanced techniques based on multiresolution or signal decomposition [6, 20, 21], these techniques usually rely on some STFT or STFT-like implementation and therefore have the same limitations in term of latency. Their latency is expected to be even higher in some variants because of additional steps in the process.

In addition to the latency of the algorithm itself, other issues related to hardware and software further increase the latency of a complete system. These can be limited to a few milliseconds with suitable architectures [5], but are worth mentioning.

Recently, the *sliding phase vocoder* has been proposed as a low latency phase vocoder implementation [3]. It shares a property with the presented algorithm in that it also processes the audio signal sample by sample. However, it works in the frequency domain and its latency is hence dependent on the DFT size. A consequence is that, unlike the presented algorithm, reducing the latency comes at the expense of reducing the quality as well. While the authors suggest that "useful information" is already produced after supplying an amount of samples corresponding to one third of the DFT size only (which would be significantly better than traditional phase vocoders in terms of latency), no accurate latency measure is provided. Furthermore, their algorithm does not run in real-time without parallel hardware.

It is worth to mention the *phase-locked vocoder*, proposed as a high-quality phase-vocoder implementation of pitch shifting [15, 16]. Although it does not address latency issues, it shares some similarities with the presented algorithm. Namely the use of frequency shifting as a part of the pitch shifting process, except that the phase-locked vocoder performs everything in the frequency domain (using the STFT). As such its latency is bounded by the size of the Fourier transforms. The proposed algorithm works entirely in the time-domain and therefore does not have this restriction.

3. Frequency shifting

Frequency shifting does the *addition* of a constant to all frequencies. This differs from pitch shifting which performs a scaling of the frequencies, or the *multiplication* of all frequencies by a constant. Indeed, the application of frequency shifting on a musical signal completely breaks the harmonic relationships, and the result sounds metallic and highly unnatural. Consequently, frequency shifting has always been reported as completely unsuitable for pitch shifting, although it has uses in other areas [13, 22]. Despite of its apparent irrelevance for pitch shifting, frequency shifting is the main component of the algorithm presented in this paper.

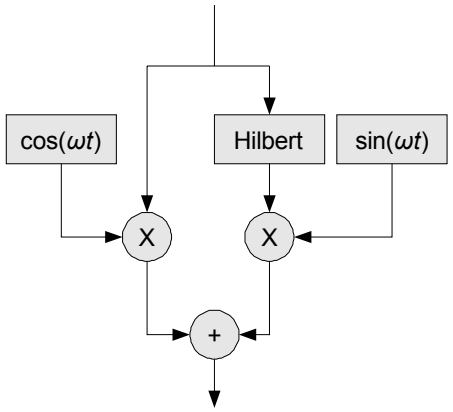

Figure 1. Frequency shifting in the time domain by ω radian per samples using the Hilbert transform.

Frequency shifting can be implemented in the time domain using a Hilbert transform, two oscillators 90 degrees out of phase, and two ring modulators [13], as illustrated in Figure 1. In practice, a true Hilbert transform is not necessary, and a pair of IIR all-pass filters can be used instead [22].

Although frequency shifting does something completely different from pitch shifting, it does it almost perfectly. No single artifact is known: neither transients nor steady sounds are affected in any way apart from the actual shifting of the frequencies. Although frequency shifting alters the phase of every component of the sound, this modification is unrelated to the well-known phasiness problem of the phase vocoder [23, 27]. With the phase vocoder, the phasiness is mainly due to the apparition of various additional aliased frequencies in the Fourier transform, that do not cancel each other properly after they are scaled due to a loss of vertical phase coherence [15, 23]. With frequency shifting, the phase modifications are comparable to those of IIR filters such as low-pass or high-pass filters. Unlike the phasiness of the phase vocoder, they are almost imperceptible. Indeed, pairs of complementary frequency shifts have been successfully used for radio transmissions [13].

4. Overview of the algorithm

The overall structure of the pitch shifting algorithm presented in this paper is illustrated in Figure 2.

The first part is a large filter-bank made of IIR filters. Although other filter types could be used, IIR filters allow low latency processing. Ideally, the bands are equally spaced on a logarithmic frequency scale, and cover the entire range of audible frequencies. More details will be given in sections 5 and 6.

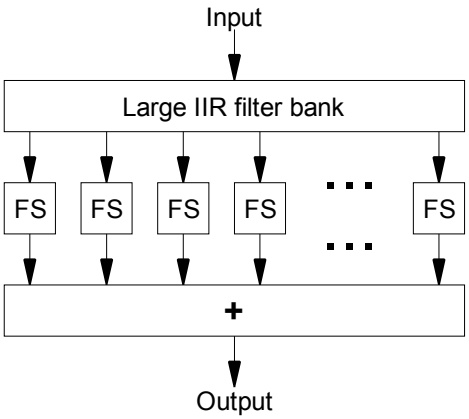

Figure 2. Structure of the algorithm. "FS" denote frequency shifting operations.

The second part consists of one time domain frequency shifter (FS) per frequency band. Each frequency shifter is set to a different amount, such that the *center* of the band is scaled by the desired ratio. To get a frequency scaling of k for instance, a frequency band centered at frequency f must be shifted by the amount $fk - f$. This corresponds to a frequency scaling by a factor k, but only for the center frequencies of the bands. Yet, if the bands are sufficiently narrow, the errors for the frequencies that are not centered in a band might eventually be small enough so that they are not audible.

Figure 3 illustrates how multiple frequency shifting operations approximate a frequency scaling process. While individual, small frequency bands are shifted, the entire spectrum is globally scaled.

Note that the phase-locked vocoder [15, 16] also uses frequency shifting, but it performs it in the frequency domain using the STFT. Unlike the presented approach, this introduces a significantly larger latency. Furthermore, reducing the Fourier transform size in the phase-locked vocoder could only reduce the latency at the expense of severe degradations of the quality. The proposed time domain approach does not have this limitation.

The aim of the next section is to develop these ideas and discuss suitable constraints for the design of the filter bank used by the presented algorithm.

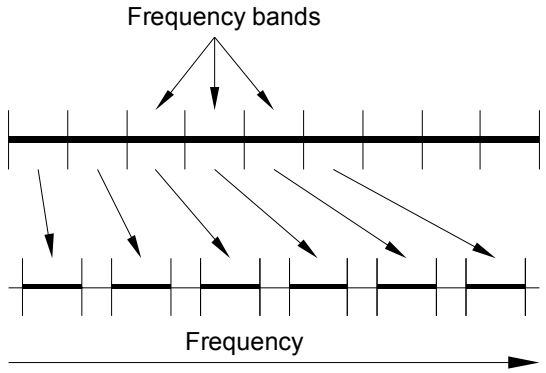

Figure 3. Shifting small frequency bands by different amounts approximates a frequency scaling operation.

5. Tuning the algorithm

This section investigates various constraints for tuning the algorithm (in particular for the design of the IIR filters). Because pitch shifting cannot be done exactly from the mathematical point of view, approximate *bounds* are inferred using either previous results on psychoacoustics or features of existing pitch shifting algorithms.

5.1. Filter bank size

First, the number of bands n that is required to get an accurate frequency scaling is investigated. If the bands are equally spaced on the logarithmic scale between a minimum frequency f_{min} and a maximum frequency f_{max}, the maximum scaling error E between the generated and desired scaled frequencies occurs on frequencies lying on the boundary of a band. Their distance with the center frequencies is half the band width. The maximum error is hence given by:

$$E = e^{(ln(f_{max}) - ln(f_{min}))/2n} \quad (1)$$

This error can yield to audible artifacts that will be referred to as *detuning*. To get competitive results, E should be at most equal to the scaling error of other techniques such as the phase vocoder. The error E of the phase vocoder when a frequency is estimated half a bin apart[1] is given by

$$E = \frac{f + f_s/2N}{f} \quad (2)$$

[1] This is an approximation only. Exact errors depend on aliasing and are very complex to infer correctly on arbitrary signals [18].

where f_s is the sampling rate and N the DFT size. Typical values are $f_s=44.1\ kHz$ and $N=1024$. Assuming a frequency $f=440$ and an audible frequency range from $f_{min} = 20$ to $f_{max} = 20000$, solving (1) and (2) for n gives a number of bands $n=72.29$. This gives an order of magnitude on the minimal number of bands to use for low-quality. For perfect results, as much as $n=1500$ bands should be taken, which equals the number of different frequencies the human ear is able to discriminate [25]. Fortunately, most sounds consist of many harmonics. As these harmonics have no direct relation with the filter-bank center frequencies[2], the errors (1) will randomly fluctuate among different harmonics of a given sound. Because previous research shows that the perceived pitch of a sound depends on all its harmonics, the effects of the error fluctuations in each harmonic will tend to *cancel each other* [24]. In practice, good results are already obtained with about 200 bands. In all cases, detuning is expected to be mostly audible on tones with very few harmonics such as pipes.

5.2. Crossover

Due to the imperfect roll-off of any filter, there is a small crossover region between every two consecutive bands of the filter bank. While the design of graphic equalizers usually permits relatively large crossover as long as perfect reconstruction is achieved, the implementation of the presented pitch shifting algorithm has to minimize the crossover region, eventually at the expense of perfect reconstruction.

Indeed, a frequency falling in a crossover region will fall in two different bands. Hence it will be shifted by two different amounts, resulting in two closely spaced frequencies, and hence to an audible tremolo (slow amplitude modulation) [13]. Tremolos are easily perceived by the human ear, as low as *-30 dB* [9]. Two different bounds for the filter bank design can be inferred depending on the nature of the sounds to process. If perfect result is desired for pure tones, every crossover must be below *-30 dB* to fully prevent audible modulations. On the other hand, if sounds with a large number of harmonics are assumed, only a few of them will fall into a crossover region. As the harmonics are unrelated to the band's center frequencies, the amount of modulation approximately corresponds to the *average* crossover level of the filter bank. This gives a much weaker constraint on the filter design.

A simple and cheap way of reducing crossover is to renounce to perfect reconstruction, and to allow notches between the bands. This is acceptable because frequency notches can be relatively large before they significantly degrade the quality of the result [12, 13]. Existing block-based time domain implementations of pitch shifting can give approximate lower bounds on acceptable amount of notches. The plain overlap-add approach for instance, does not take care of the signal characteristics, and tends to remove as much as *half* of the signal through frequency notches, in a way similar to a comb filtering operation [13]. Higher quality SOLA approaches provide improvements (with perfect reconstruction of signals made of a single sinusoid), but still fall into the same situation in the worst case [26]. On real and polyphonic signals, their behavior is somewhere in the middle. These results suggest that notches are quite well tolerated, but removing more than about a quarter of the frequencies is not desired.

5.3. Resonance

Due to the large size of the filter bank used by the presented algorithm, individual bands have a very narrow frequency width w. Hence they are expected to have a long impulse response duration of at least $1/w$, regardless of the filters that are used [13]. Because the algorithm uses equally spaced bands on a logarithmic frequency scale, the response is the longest on the lowest frequency bands and the shortest on the highest frequency bands. The resulting sound is hence expected to exhibit some kind of "down-chirp"-like resonance, if any. More precisely, the minimal impulse response duration for the lowest frequency band is[3]:

$$\frac{1}{f_{min} \cdot e^{(ln(f_{max})-ln(f_{min}))/n} - f_{min}} \qquad (3)$$

With $n=100$ bands and a frequency range between $f_{min}=20$ and $f_{max}=20000$, (3) already gives an impulse response of about 0.7 seconds for the lowest band, which is not acceptable. Previous results on psychoacoustics suggest that only response durations below 0.1 seconds for a single sound [10], and below 0.4 seconds for polyphonic audio [25] are acceptable. Rather than decreasing the number of bands n to reduce resonance, an alternate possibility is to force the frequency width of the lower bands to some fixed minimum width, and to use a logarithmic scale only for the higher bands. Hence, the resonance can be limited on low frequencies at the expense of increased detuning, but high frequencies are not affected. In all cases,

[2] The band's center frequencies are not placed on an harmonic scale.

[3] When IIR filters are used, the *actual* impulse response is infinite, but it usually drops sufficiently fast so that the *perceived* impulse response duration is in the same order of magnitude as the minimal one.

both artifacts (resonance and detuning) are expected to be dominant on low frequency sounds.

6. Results

6.1. Implementation

The presented algorithm has been implemented in Java, as part of the *PitchTech* framework [5]. Music excerpts processed with it as well as other pitch shifting techniques are available on-line [31] (the implementation of the presented algorithm is referred to as the "Rollers" algorithm in the on-line pages).

The IIR filter-bank is implemented using Butterworth band-pass filters of fourth order with crossovers at *-12 dB*. Notches between the bands remove about *35%* of the signal at a ratio *k=1*. The Hilbert transform used for frequency shifting is implemented using a pair of IIR all-pass filters. The number of bands and the minimum bandwidth are configurable by the user, with default values of *100* bands and a minimum bandwidth of *50 Hz*. These choices empirically give a reasonable tradeoff between quality and execution speed. In particular, Butterworth filters have many trivial coefficients (0, 1, -1), allowing highly optimized implementations. This implementation runs in real-time on a *2.4 GHz* Pentium with up to *200* bands.

6.2. Latency

The latency of the algorithm was measured using methodologies discussed in a previous paper [5] and the result is around *5ms*. The exact latency is actually frequency-dependent, with the highest frequencies coming within less than *1ms*. Note that one or two milliseconds are added in practice due to hardware issues [5]. Previous research states that latencies below *20ms* are usually not perceptible for melody and speech, and below *10ms* for drums [10]. While the best phase vocoders can hardly perform with a latency below *20ms* without severe degradations of the quality due to the DFT size [1, 5], the presented algorithm breaks both of these two barriers.

Note that if one tries forcing other algorithms down to similar latencies by reducing the size of blocks or Fourier transforms, the result usually gets unusable. The phase vocoder for instance would require a 220 point DFT (or less) at 44.1 kHz to achieve a latency below *5ms*. Yet this DFT size does not even permit accurate separation of the frequencies of a major chord based on middle C [11].

6.3. Artifacts

Regarding artifacts in the transformed signal, the parameters chosen in the discussed implementation are far from the optimal ones discussed in section 5, and sometimes lower than the suggested acceptable ranges of values. Therefore various audible artifacts are expected in the result. Preliminary results suggest that the presented algorithm is better in term of quality than both the "crude" phase vocoder and a simple time-domain overlap-add for various sources, but does not reach the quality of the most advanced approaches using phase-locking, multiresolution analysis or transient detection. With regards to the results of the previous section, this suggests that the proposed algorithm is already a competitive choice when low latency constraints are imposed. Yet various further improvements remain possible (see section 7). Music excerpts processed with the presented algorithm as well as various other techniques are available on-line [31]!!

All pitch shifting algorithms are particularly known for their artifacts. The phase vocoder is well known for its *phasiness* and *transient smearing* artifacts for instance, although advanced and hybrid techniques have significantly improved on the problem [6, 20, 21]. Block-based time domain approaches are known for various problems such as tempo modulation or transient duplications.

Table 1 summarizes all the artifacts that have been reported in the literature concerning the phase vocoder [9, 15, 18, 23] and block-based time domain approaches [7, 13, 26]. The last column gives artifacts of the presented algorithm, as predicted by section 5.

Table 1. Comparison of artifacts between phase vocoder techniques (PV), time-domain techniques (TD) and the presented algorithm (Rollers).

Artifact	**TD**	**PV**	**Rollers**
Phasiness	Yes	No	No
Transient duplications	No	Yes	No
Detuning	Yes	No	Yes
Phase cancellations	No	Yes	No
Tempo modulation	No	Yes	No
Tremolo	Yes	No	Yes
Frequency notches	No	No	Yes
Resonance	No	No	Yes
Stereo field loss	Yes	Yes	No

Note that Table 1 omits artifacts that are the consequence of others for clarity. In practice, phase cancellations for instance can result indirectly in tremolos, detuning and frequency notches.

An interesting result form this comparison is the fact that phasiness, transient duplications and tempo modulation are absent from the presented algorithm. From the musical point of view, this is desirable because it means that it perfectly preserves the attacks of instruments, which is not the case with the two other approaches. Only advanced approaches [7, 8, 13, 21] can get close to it in this aspect. From the theoretical point of view, this shows that the time-frequency tradeoff can yield artifacts other than phasiness, transient duplications or tempo modulation. With the proposed algorithm, it yields a compromise between detuning and resonance. These two artifacts are mostly affecting the low frequencies.

Also note that the presented algorithm is insensitive to the input signal characteristics. As a consequence, the stereo field is preserved and no explicit processing is required[4].

7. Future works

The implementation of the presented algorithm was done on a standard computer. Hence processing speed imposed the use of very cheap Butterworth filters in order to run in real-time. Yet the structure of the algorithm (see section 4) is essentially parallel: each band can be extracted from the input signal and frequency shifted independently; only the final summation must be serialized, yet this only represents a minor part of the whole process. A future working direction is therefore to implement the algorithm on parallel hardware, such as general-purpose graphic processing units such as NVIDIA's CUDA [30]. On such hardware, it is expected that much better filters (Elliptic, Chebychev, and of much higher order) could be used without sacrificing real-time and low latency, but yielding significantly improved quality.

Quality improvements with the help of parallel hardware also seek for better quality assessments and comparisons with other approaches. A future working direction would be to use audio metrics for each of the artifacts depicted in Table 1 in order to better quantify them.

The phase-locked vocoder was mentioned as an algorithm that also makes use of frequency shifting, but entirely works in the frequency domain. This algorithm performs frequency estimations in order to improve the result [15, 17]. The same could be done in the time-domain and hence be used to improve the presented algorithm as well by accurately estimating the dominant frequency of each band. Unfortunately, frequency estimations introduce latency [13], which makes this problem challenging with respect to the initial goal. A possibility to investigate is to try using frequency estimations for the high frequencies only (they introduce less latency), and to estimate low frequencies based on probabilistic models: a given low frequency value is a better candidate for instance if many estimated high frequencies are multiple of it, as most instruments are made up of harmonics.

It seems that the use of frequency shifting in the *time domain* to express pitch shifting (or frequency *scaling*) has not been investigated before (although it has been used in the *frequency domain*). While the primary goal was low latency, the use of frequency shifting in the time domain may have use in other areas. Its use in other audio effects could be considered as a future work.

[4] Regarding the stereo field in Table 1, "yes" means that it is lost unless handled *explicitly*.

8. Conclusion

This paper introduced a novel low latency pitch shifting algorithm using a subband approach based on IIR filters and time domain frequency shifting. Issues regarding its implementation were detailed.

From the theoretical point of view, a novel approach to pitch shifting in the time domain was proposed, whose implementation radically differs from previous approaches. It was shown that this new approach yields different artifacts than existing algorithms.

From the practical point of view, the presented algorithm was shown to achieve latencies below *10ms* in a live system, where previous approaches can hardly go below *20ms*.

9. References

[1] E. Lee, T. Karrer, J. Borchers, *An Analysis of Startup and Dynamic Latency in Phase Vocoder-Based Time-Stretching Algorithms*, International Computer Music Conference, vol. II, pp. 73 – 80, 2007.

[2] W. A. Sethares, *Rhythm and Transforms*, Springer, 1st edition, July 31, 2007.

[3] R. Bradford, R. Dobson, J. Ffitch, *The Sliding Phase Vocoder*, International Computer Music Conference, vol. II, pp. 449 – 452, 2007.

[4] Jens Gulden, *JJack - Using the JACK Audio Connection Kit with Java*, Linux Audio Conference, pp. 49 – 55, 2007.

[5] N. Juillerat, S. Müller Arisona, S. Schubiger-Banz, *Real-time, Low Latency Audio Processing in Java*, International Computer Music Conference, vol. II, pp. 99 – 102, 2007.

[6] T. Karrer, E. Lee, J. Borchers, PhaVoRIT: *A Phase Vocoder for Real-Time Interactive Time-Stretching*, International Computer Music Conference, pp. 708 – 715, 2006.

[7] D. Dorran, *Audio Time-Scale Modification*, PhD Thesis, Dublin Institute of Technology, 2005.

[8] E. Ravelli, M. Sandler and J. P. Bello, *Fast Implementation for Non-Linear Time-Scaling of Stereo Signals*, Digital Audio Effects Conference, 2005.

[9] D. Dorran, E. Coyle, R. Lawlor, *An Efficient Phasiness Reduction Technique For Moderate Audio Time-Scale Modification*, Digital Audio Effects Conference, 2004.

[10] N. Lago, *The Quest for Low Latency*, International Computer Music Conference, pp. 33 – 36, 2004.

[11] R. G. Lyons, *Understanding Digital Signal Processing*, Prentice Hall PTR, April 2004.

[12] S. M. Petrov, *Perception of a Spectrally Deprived Speech Signal*, Human Physiology, Volume 29, Number 1, January 2003.

[13] U. Zölzer, *DAFX - Digital Audio Effects*, John Wiley & Sons, 2002.

[14] Kees van den Doel, Dinesh K. Pai, *JASS: A Java Audio Synthesis System For Programmers*, Proc. of the International Conference on Auditory Display, Espoo, Finland, 2001.

[15] J. Laroche, M. Dolson, *New Phase-Vocoder Techniques for Real-Time Pitch-Shifting, Chorusing, Harmonizing and Other Exotic Audio Modifications*, Journal of the Audio Engineering Society, Vol. 47, No. 11, 1999.

[16] J. Laroche, M. Dolson, *New Phase-Vocoder Techniques for Pitch-Shifting, Chorusing and Other Exotic Effects*, IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, pp. 91 – 94, 1999.

[17] J. Laroche, M. Dolson, *Improved Phase Vocoder Time-Scale Modification of Audio*, IEEE Transactions on Speech and Audio Processing, vol. 7, no. 3, pp. 323 – 332, May 1999.

[18] M. S. Puckette, J. C. Brown, *Accuracy of Frequency Estimates Using the Phase Vocoder*, IEEE Transactions on Speech and Audio Processing, vol. 6, no. 2, pp. 166 – 176, March 1998.

[19] S. N. Levinne, T. S. Verma, J. O. Smith, *Multiresolution Sinusoidal Modeling for Wideband Audio with Modifications*, IEEE International Conference on Acoustics, Speech and Signal Processing, Seattle, US, pp. 3585 – 3588, 1998.

[20] T. S. Verma, T. H. Y. Meng, *Time Scale Modification Using a Sines+Transients+Noise Signal Model*, Proceedings of the Digital Audio Effects Workshop, Barcelona, pp. 49 – 52, 1998.

[21] S. N. Levine, J. O. Smith, *A Sines+Transients+Noise Audio Representation for Data Compression and Time/Pitch Scale Modifications*, 105th Audio Engineering Society Convention, San Francisco, 1998.

[22] S. Wardle, *A Hilbert-Transformer Frequency Shifter for Audio*, Proceedings of the Digital Audio Effects Workshop, Barcelona, pp. 25 – 29, 1998.

[23] J. Laroche, M. Dolson, *Phase-Vocoder: About this phasiness business*, IEEE Proc. Workshop Appl. of Signal Processing to Audio and Acoustics, 1997.

[24] R. Meddis, L. O'Mard, *A unitary model of pitch perception*, The Journal of the Acoustical Society of America, Volume 102, Issue 3, pp. 1811 – 1820, 1997.

[25] Chris A. Lanciani, *Auditory Perception and the MPEG Audio Standard*, PhD in Electrical Engineering, Georgia Institude of Technology, 1995.

[26] J. Laroche, *Autocorrelation Method For High-Quality Time/Pitch-Scaling*, IEEE Proc. Workshop Appl. of Signal Processing to Audio and Acoustics, pp. 131 – 134, 1993.

[27] J. L. Flanagan, R. M. Golden, *Phase Vocoder*, Bell Syst. Tech. J., vol. 45, pp. 1493 – 1509, Nov. 1966.

[28] S. Bernsee, *Pitch Shifting Using the Fourier Transform*, http://www.dspdimension.com/admin/pitch-shifting-using-the-ft/

[29] O. Parviainen, *SoundTouch Audio Processing Library*, http://www.surina.net/soundtouch/

[30] NVIDIA Corp., *CUDA Zone Home Page*, http://www.nvidia.com/object/cuda_home.html

[31] Companion page, http://www.pitchtech.ch/Confs/ICALIP2008/Rollers.html