Automatic technique in frequency domain for near-lossless time-scale modification of audio

Jordi Bonada
Audiovisual Institute, Pompeu Fabra University
Rambla 31, 08002 Barcelona, Spain
jordi.bonada@iua.upf.es
http://www.iua.upf.es

Abstract

Time-scale modification of sounds has been a topic of interest in computer music since its very beginning. Many different techniques both in time and frequency domain have been proposed to solve the problem. Some frequency domain techniques yield high-quality results and can work with large modification factors. However, they present some artifacts, like phasiness, loss of attack sharpness and loss of stereo image. In this paper we propose a new frequency domain approach for an automatic near-lossless time stretching of audio.

1. Introduction

Time-scaling an audio signal means changing the length of the sound without affecting other perceptual features, such as pitch or timbre. The system presented in this paper strives for obtaining a near-lossless time-scaled audio modification without any other perceptual change, just like if the music was performed faster or slower.

The system has been implemented taking into consideration two important issues: easiness of use and speed of computation. The system should be very easy to control and to make use of, and should have no complex parameter selection at all. Concerning the processing speed, it should be able to apply the effect in real-time, without any DSP or hardware add-on.

2. Frame based frequency domain technique

The technique used in this paper is a frame based frequency domain technique. It's based on the well-known phase-vocoder [GL84, D86, ML95]

2.1 General diagram

In figure 1 we can see the general diagram, where the input is the audio signal and the output is the time-scaled version of the input. First or all the input sound is *windowed* and goes through the *FFT module* to get the analysis frame (AF_n), with the spectrum amplitude and phase envelopes. Then the *time-scale module* generates the synthesis frame (SF_m) that goes to the *inverse FFT* (IFFT). And finally the *Windowing&Overlap-Add block* divides the sound segment by the analysis window and multiplies it by the overlap-add window, to get the output sound.

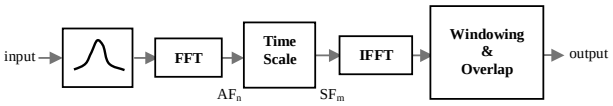

Figure 1 General diagram. It is clear that both analysis and synthesis processes must use the same window size and type.

2.2 Constant frame rate

It is important to remark that the frame rate used is the same both in analysis and synthesis modules, as opposed to the most broadly used change of frame rate in synthesis to achieve the time-scale. In the next figure we can see what happens in our system in the cases of a time-scale factor that makes the audio length larger than the original duration (TS > 1), and the opposite (TS < 1).

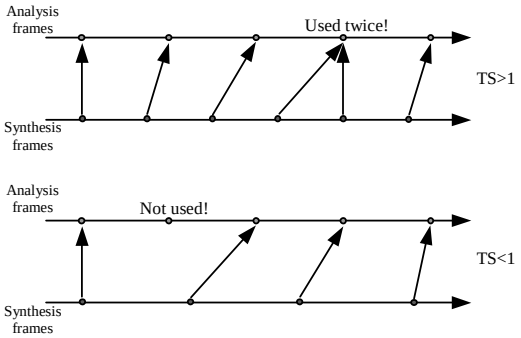

Figure 2 Analysis and synthesis frames. The horitzontal axis corresponds to the time of the center of the frame in the input audio signal. Therefore, when TS>1, the time increments in the input audio signal will be shorter in synthesis than in analysis frames, but the actual frame rate will be exactly the same. Each synthesis frame points to the nearest analysis frame looking at the right.

As it is shown in figure 2, sometimes one analysis frame is used twice (or more), and sometimes it is never used! This will not add any artifacts if the frames are small enough and the sound characteristics don't change very

fast. In the case of a percussive attack, for example, a repetition or an omission could be noticeable by the listener even if the frames are really small (about 10 ms). Therefore some wise knowledge of the sound is needed to decide where to repeat or omit frames, as it will be later exposed.

3. The Time-Scale module

3.1 Description

The inputs to the time-scale module are the analysis frames (AF_n) containing the spectrum amplitude and phase envelopes. A peak detection algorithm followed by a peak continuation block is applied to the current and previous (Z^{-1}) amplitude envelopes. The peaks that are going to be continued are used as inputs to the spectrum phase generation module. The spectral amplitude envelope is unchanged by the time-scale module. Only the phase is changed.

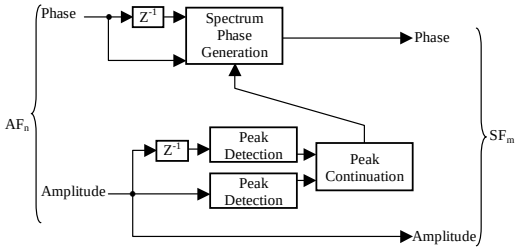

Figure 3 The time-scale module.

3.2 Peak detection and continuation

The peak detection algorithm used is very simple. It just locates relative maxima (around a band, $A_{peak}(i)>A_{peak}(m)$ for $|m-i|\leq B/2$, where B is the required bandwith of the peak) of the amplitude envelope and applies a parabolic interpolation. The continuation algorithm just chooses the closest peak in frequency, and resolves peak conflicts.

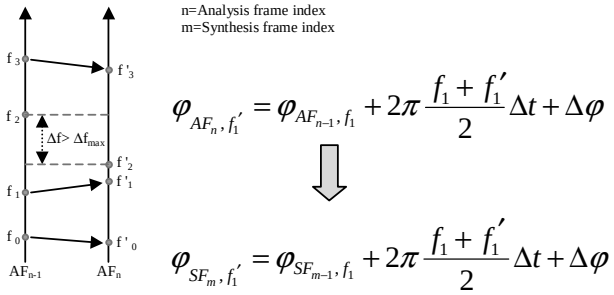

Figure 4 Peak continuation and phase generation. The peak of frequency f_2 in AF_{n-1} is not continued in the next frame AF_n because the difference $\Delta f = f'_2 - f_2$ is greater than the defined maximum deviation frequency Δf_{max}, and f'_3 is closer to f_3 than to f_2

3.3 Spectrum Phase generation

The phase of the peak is calculated supposing that frequency varies linearly between two consecutives frames and that exists some phase deviation (fig. 4). Since the frame rate is the same for analysis and synthesis, then the phase variation between two consecutives frames can be supposed to be also the same. As pointed in [LD97], the peaks subdivide the spectrum into regions around each peak where the phase is related to the peak's phase. The phase around each peak is obtained applying the delta phase function of the original spectrum phase envelope.

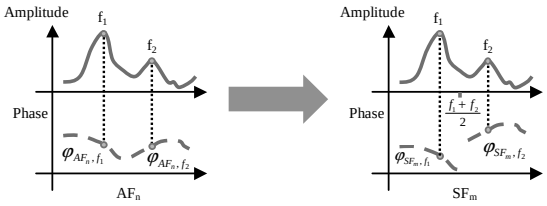

Figure 5 Original delta phase function around each peak.

4. Fast changes

We cannot time-scale fast changes (specially attacks) in order to keep the perceptual features of audio. For example, the percussive attacks should be kept as they are. The solution proposed is to detect the fast changes and keep their original length. This means that a greater amount of time-scale should be applied around a fast changing region, so to keep the average time-scale modification factor.

4.1 Detection of fast changes

The detection of fast changes (*usually attacks*) is done processing several observations: bank filter energies, Mel cepstrum coefficients, and its deltas. The bank of filters is composed of 42 bands between 40 and 20353.6 Hz, following a Mel scale. The window used is a Hamming of about 23.2ms. Several simple rules are applied combining these inputs, but the main idea is to find points of maximum increasing slope.

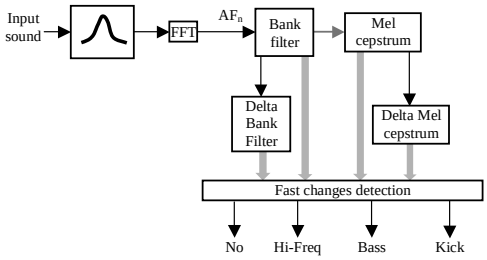

Figure 6 Fast changes detection.

4.2 Fast changing region

The audio around the fast changing time should not be time-scaled. Besides, it's important to keep the original phase as much as possible in a fast changing region, to preserve the perceptual features of the input sound.

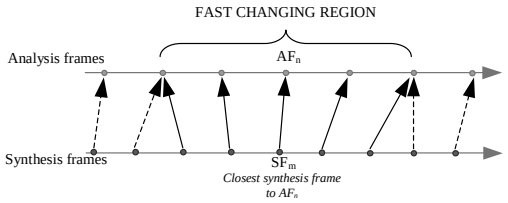

Figure 7 Fast changing region. Notice that no frame is omitted or repeated along the fast changing region. Therefore, the original timing in the fast changing region is preserved.

We can use the original spectrum phase envelope above a specified frequency cut (*FC, 2500Hz has proven to be good selection*) along the fast changing region, with no perceived artifacts and preserving the perceptual behaviour. For frequencies lower than the frequency cut, we need to continue the phase (section 3.3) of the quasi-stable sinusoidal peaks. But we can use the original phase in the non stable peaks (*this way we just assure the phase envelope in the synthesis frame being more similar to the original one, and therefore guarantee a more similar timbre*).

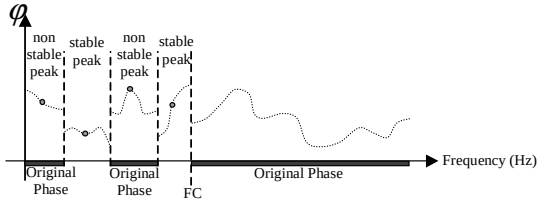

Figure 8 Phase envelope in the fast change region.

There are different types of fast changes, where each type defines a different frequency cut (FC):

- *hi-freq*: only fast changes in the high frequencies (cymbal, crash…). FC high
- *bass*: fast changes in low frequencies (kick drum, tom…) FC low
- *kick*: big changes at all frequencies. FC zero (*the full original phase spectrum is used*)

5. Parallel windowing

5.1 Frequency vs. time resolution

It is good to have long windows in order to achieve a high frequency resolution, but also to have short windows so to achieve a high time resolution. If an important low frequency sound is present in the audio signal, then a long window is really needed, because the low frequency peaks will be very close one to the other, and with a short window the peaks will not be detected properly. Otherwise, if we use this long window, then the time-scale process will add some reverberance and smoothness to the sound. This leads us to think about parallel windowing.

5.2 Multiple windows in parallel

We propose to use several channels (band pass). Each channel is the result of an FFT with a specific window size, window type and zero padding. For low frequencies the window should be longer than for high frequencies. The peak detection is applied to each of the channels. The peak continuation takes care of the desired channel frequency cuts, so it can connect peaks of different channels. The Time-scale module fills the spectrum of all

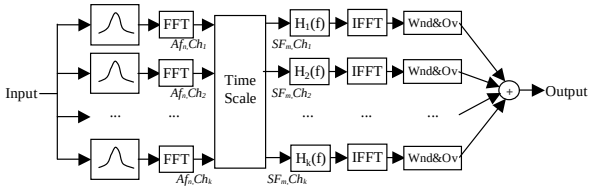

Figure 9 Multiple parallel windowing. There are *K* different channels, each one with a specific window size, window type and zero padding factor.

the channels (*amplitude and phase envelopes*) and applies a set of parallel filters $H_n(f)$ that must add up to a constant (*all pass filter*). Time varying channel frequency cuts are required because if a peak is very close to a frequency cut, then breaking it into different channels could produce some artifacts. This way we can guarantee the amplitude and phase envelopes around the peak to be the desired ones.

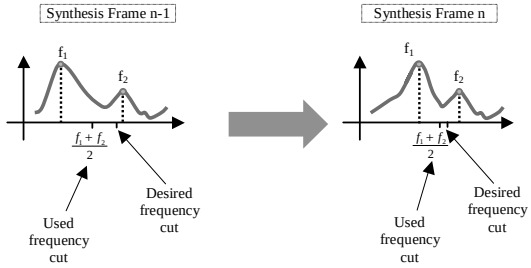

Figure 10 Variable phase frequency cut. The used frequency cut is calculated as the middle point between the two closest peaks to the frequency cut that come out of the peak continuation.

A useful example with three bands could be the next one, with a frame rate of 86 frames/sec

	Band 1	Band 2	Band 3
Frequency cut (*Hz*)	700	2400	22050
Window size (*in frames*)	8	4	3
Window type	KB2.0	KB3.0	KB3.0

6. Stereo time-scale

If we want to keep the stereo image, then it is necessary to preserve the phase and amplitude relation between left and right channels. Since we don't change the spectrum amplitude envelope of the analysis frame in the system, then the amplitude relation between channels will be already preserved if we just use always the same frame

times for left and right channels. This means that fast changes synchronization is needed to ensure that the left and right channel frames have the same time tag. Therefore, only the phase relation needs to be carefully treated. In the next figure we can see the diagram showing the stereo time-scale system. Notice that the number of FFT and IFFT operations is multiplied by two, and as a consequence the same happens to the processing time.

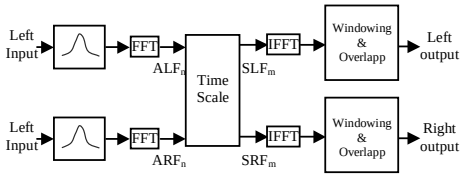

Figure 11 Stereo time-scale

6.1 Channel phase relation

As it was said before, the time-scale module should preserve the phase relation between channels. The phase around the peak is obtained with the original delta phase function (section 3.3). Therefore, by straight-forward algebra it is very clear that if we keep the phase difference of the peaks from different channels, than all the spectrum bins around the peaks will also keep the phase relation, and the stereo image will be preserved.

7. Time varying time-scale

The amount of time-scaling can be a time varying function without affecting the quality of the processed audio. The only significant change in the system is that the time increments of the synthesis frames into the input signal is not constant. The rest of the system remains exactly the same. This opens a lot of new and interesting perspectives:

- The system could be easily adapted and used for alignment and synchronization of two sound sources.
- The amount of time-scale can be used in a wise way to inspire emotions. For example, to increase the climax or the suspense of a musical piece, by slowing or increasing the tempo during certain fragments.
- An interesting application could be to control the time-scale factor the same way as the orchestra conductor does and play in real-time a previously recorded background with a live performance.

8. Results and conclusions

The system has been fully implemented. Several audio sources have been tested in order to prove the robustness and quality of the time-scale modification. The tests consisted in hearing and comparing carefully the input and the output audio signals. Different styles of music were used as input into the system. The processed audio yields a high quality (near loss-less) in a broad range of time-scaling factors (70% to 150%). The detection and treatment of fast changes (*attack*) works quite well in most types of audio inputs (musical or not). The attack sharpness is really preserved in the processed signal, as well as the stereo image. Besides, it's possible to apply a time varying time-scale factor with no loss of quality.

Weaknesses of the system:

- The detection and characterization of fast changes needs to be improved.
- Very low pitch harmonic sounds require preserving the phase alignment, specially if the waveform is quite impulsive.
- It's slow. About 50% real-time on a AMD Athlon CPU running at 700Mhz (44Khz, 16 bit)

9. Future work

There are still many things to be done. First of all, the current system needs a better detection and characterization of fast changes. It's also important to optimise the implementation and the system to get a real-time time-scaling of stereo sound files (44Khz, 16 bit) on the last PC's in the market without any DSP or hardware add-on. Another area of interest is the object filtering: detect harmonic audio objects, preserve the phase alignment of harmonic objects (*this is important for the perceived timbre, specially for low frequency sounds*), preserve original vibratos and tremolos. It would be also interesting to transform the system into a pitch shifter, using most of the current implemented processes. Finally, the system should be adapted to the voice, so to be able to keep the original length of plosives and short consonants, use a wise dynamic time-scale factor, etc.

References

[GL84] D.W. Griffin, J.S.Lim. "Signal estimation from modified short-time fourier transform". *IEEE Trans. Acoust., Speech, Signal Processing*, ASSP-32(2):236-243, Apr 1984.

[D86] M. Dolson. "The phase vocoder: A tutorial". *Computer Music Journal*, 10(4):14-27, 1986.

[ML95] E. Moulines, J. Laroche. "Non parametric techniques for pitch-scale and time-scale modification of speech". *Speech Communication*, 16:175-205, Feb 1995.

[LD97] J. Laroche, M. Dolson. "About this phasiness business". *Proceedings of International Computer Music Conference*, 1997.