TIME WARP

A PROFESSIONAL TIME STRETCHING APP

Reza Ghanavi (SID: 480187864)

Final Proposal for Digital Audio Systems, DESC9115, 2018
Graduate Program in Audio and Acoustics
Sydney School of Architecture, Design and Planning, The University of Sydney

Abstract:

Time Warp is a fantastic plugin that converts your digital audio workstation (DAW) to a powerful time stretcher system. The function offers a reliable algorithm based on *Filter Bank* (sum of sinusoids in frequency domain), the technique that can implement high-fidelity time scaling on a variety of sound sources such as synchronizing the audio with video content and tempo adjustment of the music.

Problem description:

Time-scaling has been one of the highest in demand technologies in the last few decades. Most of the classical commercial applications of time stretching implement time domain processing based on synchronized overlap-add systems such as **SOLA** and **OLA**. The performance of these simple applications is poor when a user needs to apply a large time scaling factor on a sound material. Apart from warbling (a type of frequency modulation), these algorithms can also produce doubling and skipping on percussive signals (Driedger, J. et al., 2014)

In contrast, the framework of **Time Warp** application is a superior version of frequency based time-scale modification (Phase Vocoder) algorithm that allows sound designers to expand their revolutionary ideas in terms of sound effect creation without any concern about the intruding of the prevalent artifacts or scaling limitations.

The main purpose of this proposal report is to provide an in-depth technical review about the current **Time Warp** plug-in application. The review will also try to outline the extent of improvements that this system has created in comparison to the previous DSP based on time domain overlap and add system.

Technical Specification:

Time Warp can be used with any mono (.wav) file. The main principle of the DSP is based on phase vocoder algorithm for time stretching. The algorithm scales the duration of the input signal by shifting the phase of individual segments of signal while preserving the magnitude and instantaneous frequencies *(Zölzer, U. 2008).* The output signal can be scaled by the ratio of Rs (Synthesis hop size) to Ra (Analysis hop size) which define a new phase difference value (Psi) as follows.

$$\Delta\boldsymbol{\psi}(\boldsymbol{k}) = \frac{\boldsymbol{R_s}}{\boldsymbol{R_a}}\Delta\boldsymbol{\varphi}(\boldsymbol{k})$$

By implementing the filter bank approach, it is possible to reconstruct the output signal through the integration of the real part (cosines) of phases and calculating the instantaneous frequency for individual bins (Zölzer, U. 2008).

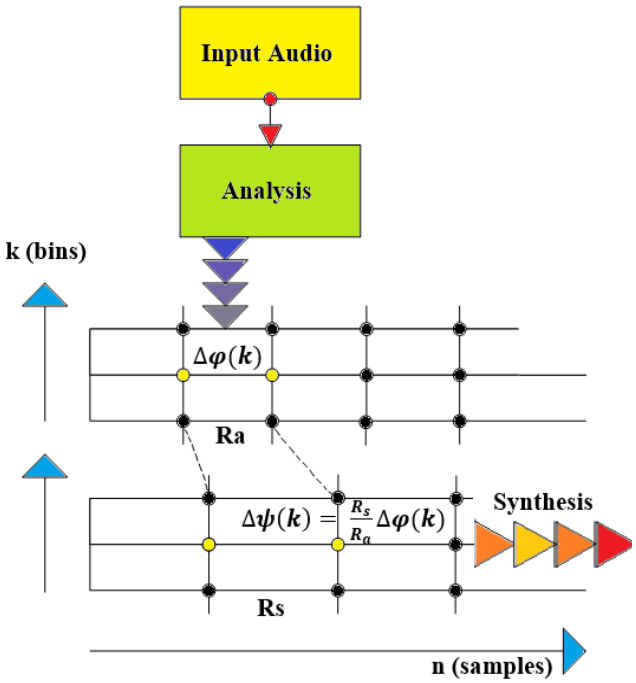

***Figure 1**. The upper time-frequency grid depicts analyzed blocks with hop size **Ra** and phase difference between unwrapped phases $\Delta\boldsymbol{\varphi}(\boldsymbol{k})$. The synthesis processing has started from the resized time-frequency grid with phase difference $\Delta\boldsymbol{\psi}(\boldsymbol{k})$ and hop size **Rs** (lower graph).*

For calculating the output samples in resynthesizing process, the synthesis phase growth derivative factor $\boldsymbol{d\psi(k) = \Delta\varphi(k)/R_a}$ should be integrated to the value of complex time-frequency synthesis phase value as follow.

$$\tilde{\boldsymbol{\psi}}(\boldsymbol{n+1,k}) = \tilde{\boldsymbol{\psi}}(\boldsymbol{n,k}) + \boldsymbol{d\psi(k)}$$

The DSP divides the input audio signal into a series of blocks named grains and each individual grain is analyzed by a FFT filter that calculates the amplitude and phase of the signal's frequency content while shifting by the size of **Ra**. Applying the time scale factor and interpolating the correspondent data can result in **Rs** (synthesis hop) with a new phase shift while keeping the frequency contents unchanged (Zölzer, U., 2008). Finally, the time scaled signal **y(n)** can be reconstructed with summation of weighted ***cosines*** resulting from a linear interpolation between two successive ***amplitude (A)*** values and similarly two psi phase ($\tilde{\boldsymbol{\psi}}$) values on the introduced grid with a hop size of ***one*** (Zölzer, U. 2008).

$$\boldsymbol{y(n) = \sum_{k=0}^{N/2} A(n,k)\, cos\, \tilde{\psi}(n,k)}$$

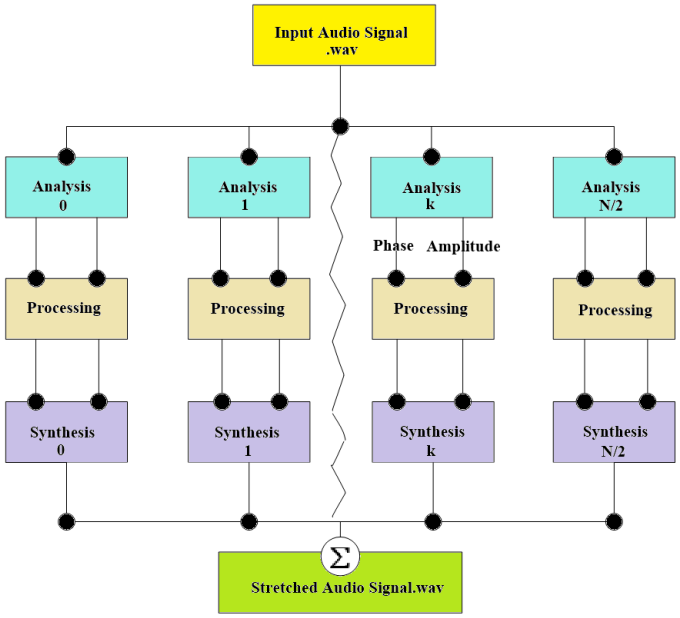

***Figure 2.** Time Warp DSP block diagram*

Implementation:

A graphical user-friendly interface has been offered for the **Time Warp** plugin application. The **GUI** allows users to apply time stretching on the opened mono input signal ***(.wav)*** and adjust the scaling factor between ***20*** to ***800*** percent. Furthermore, the output quality can be selected from the ***Precision*** drop-down menu. The precision option offers three different quality choices ***(High, Medium and Low)*** that can adjust the default length of **Ra** *(256 samples for high fidelity analysis)* to 512 or 1024 samples for lower qualities respectively.

Although the lower quality presets can produce a little ***phasiness*** artifact by reducing the phase coherence between multiple bins, increasing the ***Hop Size*** can reduce the number of analysis and shortening the total time of processing by a factor of ***2*** in each step (Laroche, J. et al., 1999) This can save evaluation time when there is no need for a high-quality output. For example, a DJ can save ample time when adjusting the tempo of a track.

Figure 3*. proposed GUI interface for Time Warp App (drown in photoshop by author).*

Evaluation:

To evaluate the application, four different audio files (.wav) have been selected and stretched by a factor *(High quality preset)*. The type and style of selected audio files have been chosen to address some prevalent time stretching scenarios involving human voice, music with percussive components and music with harmonic components.

Human voice, Speech & Vocals:

The quality of the proposed DSP in comparison to *overlap and add systems* for stretching human voice is significantly better. The objective and subjective evaluations have shown artifact free results for a variety of input audio files. This app can be fully functional for adjusting the audio streams in video clips such as synchronizing the audio material for slow motion movies (Laroche, J. et al., 1999)

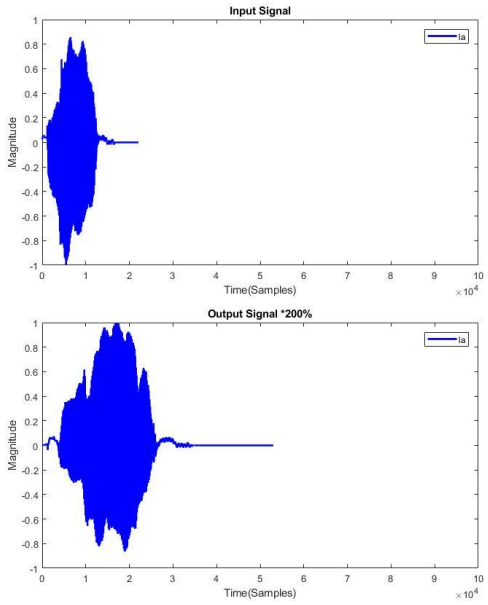

Figure 4. ***la.wav*** wave form is *stretched by a factor of* ***200%.***

Preserving the tone and formant of stretched speech is a key aspect for using this application as a tool in other language learning and speech therapy software applications.

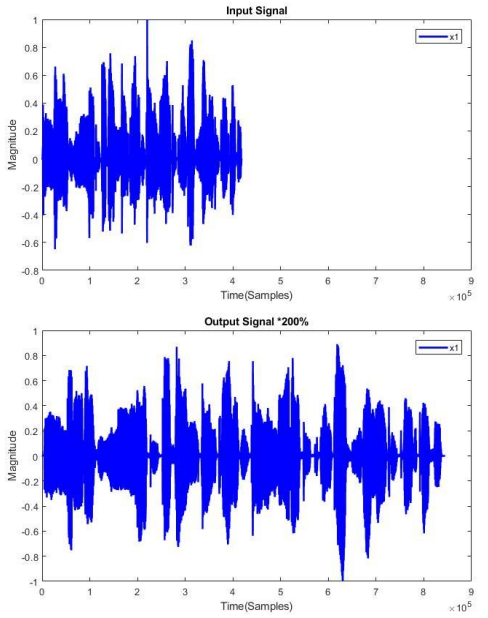

Figure 5. ***x1.wav (vocal)*** *waveform stretched by a factor of* ***200%.***

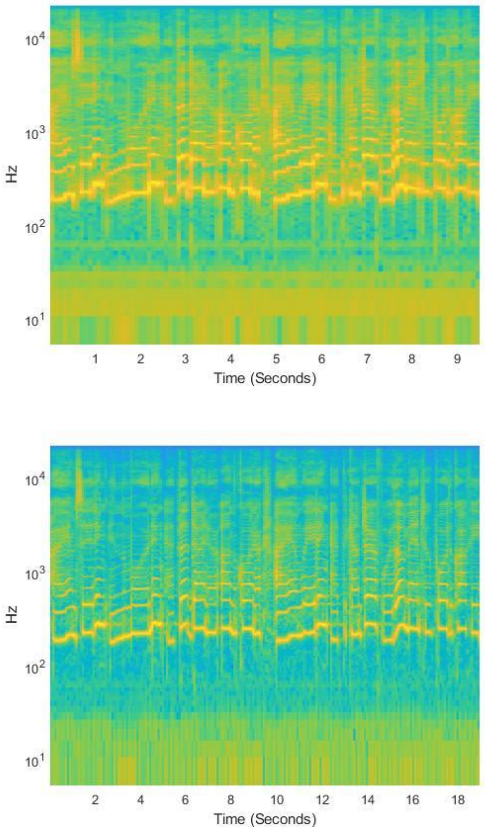

Figure 6. ***x1.wav (vocal)*** *spectrum (top) and the stretched spectrum by 200% (down) depicts an artifact free output in comparison to input signal.*

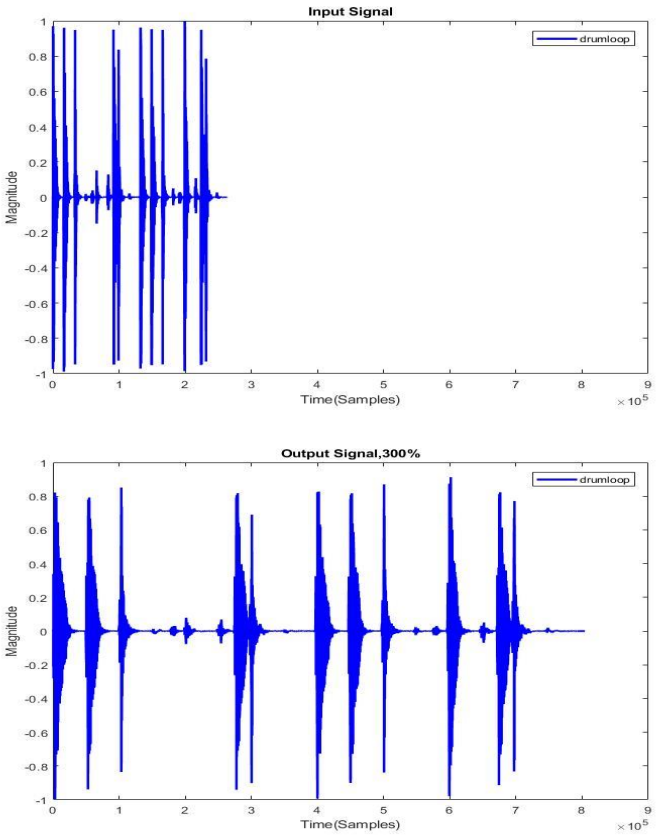

Figure 7. ***Drumloop.wav*** *time_magnitude response. The percussive peaks and beats in the scaled signal (300%) are preserved.*

Music, Percussive and Harmonic components:

The DSP can be a great tool for adjusting the tempo for percussive music. An accurate vertical and horizontal phase coherence in synthesis channels between the frequencies in each bin have developed the DSP for capturing precise musical snapshots (Laroche, J. et al., 1999) As seen in the following graphs, the temporal rhythm and sharp peaks are preserved accurately.

Using the phase vocoder algorithm is an advantage for improving phase discontinuity that has been a dominant artifact for many other applications based on *overlap and add* algorithms. **Time Warp** is able to preserve the frequency content of harmonic musical signals. Hearing test of the first five measures of Beethoven's Symphony No.5 have shown a high similarity between the timbre of original and stretched harmonic signals which have been shown in the graphs for better visual evaluation.

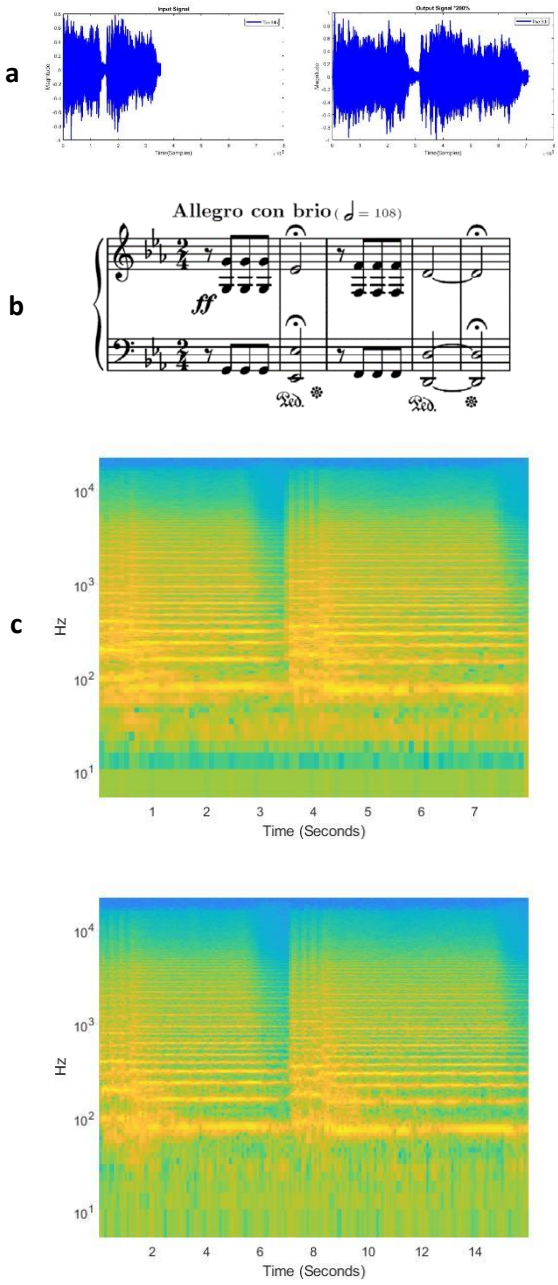

Figure 8. ***The 5th.wav.*** *in this picture the quality of* ***Time Warp*** *in preserving musical harmonic contents has been illustrated. (a): wave forms of input and output (scale 200%) signals. (b): Beethoven 5th symphony first five measures (Driedger, J. et al. 2014) (c): input and output spectrograms.*

Conclusion:

In the development of **Time Warp** application, a new technology based on frequency bank approach (Zölzer, U. 2008) has been implemented. As described and illustrated technically and graphically, this tool apps produces less artifacts in comparison to similar apps, especially when used for speech and harmonic music. With an attractive user-friendly interface, GUI has been designed for easy access to time and quality adjustments. Users can benefit from this application in different fields of functionalities such as designing sound for movies and manipulating the musical and vocal tempo.

References:

[1] Driedger, J., & Müller, M. (2014, September). TSM Toolbox: MATLAB Implementations of Time-Scale Modification Algorithms. In DAFx (pp. 249-256).

[2] Laroche, J., & Dolson, M. (1999). Improved phase vocoder time-scale modification of audio. IEEE Transactions on Speech and Audio processing, 7(3), 323-332.

[3] Lazzarini, V., Timoney, J., & Lysaght, T. (2005). Time-stretching using the instantaneous frequency distribution and partial tracking.

[4] Smith, S. (2002). The Scientist and engineer's guide to digital signal processing. California: California Technical Publishing.

[5] Zölzer, U. (2008). Digital Audio Signal Processing. Chichester, West Sussex: J.Wiley & Sons.

[6] Zölzer, U. (2011). DAFX - digital Audio Effects. Chichester, West Sussex: J.Wiley & Sons.

Appendix: (Time Warp package)

Function and scripts:

- Time_Stretch_Bank.m (codes related to the *Precision options***,** adopting *the time-scaling domain* to the function and *scripts* have been developed by *the author*. Remainder of function codes are derived from *DAFX book, 2nd ed., chapter 7.)*
- Time_Stretch_Bank_Script.m
- Plot_Script.m

Example stretched Audio files:

- Drumloop_200%
- flute_400%
- La_200%
- The 5th_200%
- x1_50%
- x1_200%
- x2_300%

Raw Audio Files:

- Drumloop
- flute
- la
- The 5th
- x1
- x2

(source of media: DAFX)

Developed Matlab codes:

```
[DAFx_in, FS]  =
audioread('la.wav'); % Input from
wave file
Scale = 200;
% funnction parameter setting
Precision   =   "High";
% function parameter setting
DAFx_out =
Time_Stretch_Bank(DAFx_in,Scale,FS
,Precision);

% Function call
audiowrite('tstretch_bank.wav',DAF
x_out, FS );
% Saves Output as wave file
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if     Precision == "High";  % set
precision factor to "High"
        pf=1;
elseif Precision == "Medium";% set
precision factor to "Medium"
        pf=2;
elseif Precision == "Low";   % set
precision factor to "Low"
        pf=4;
else
       disp('error, set precision
factor: High , Medium ,Low') ;
        end
Scale = Scale/100; % Time scale
factor "alpha"
n1            = 256*pf; % analysis
step increment [samples]
n2  =round( n1*Scale);  %
synthesis step increment [samples]
.........................

.........................
```