# Audio Time-Compression

ELEC-E5620

Priyanshi Pal and Hercule Le Guirec

19 April 2024

# Contents

# 1 Introduction

## 1.1 Background

Time Scale modification (TSM) is a technique which is often implemented widely used in various applications like music editing, sound design and post production, and similar methods can also be applied for videos as well. Many years ago, before the prevalence of digital devices, TSM was primarily achieved using analog methods such as tape manipulation. For varying speech, tapes were also modified by cut-and-splice methods to change the playback speed. With the advent of Digital signal processing, various time domain and frequency domain techniques were proposed to do the same, especially with the goal of maintaining the pitch for various playback speeds. Some popular methods include include Wavelet-based methods, Harmonic/Percussive Separation (HPS), and time-domain techniques like Overlap-Add (OLA) and Waveform-Similarity based Overlap-Add methods.

The wavelet based time-scale modification techniques use wavelet transforms to decompose the audio signal into different frequency bands. HPS focuses on separation of harmonic (pitched) and percussive components from audio signal. Using he separation, HPS can facilitate independent time-scale modification of these components, which can be beneficial for creative audio manipulation. Prior to discussing the time domain based techniques which are the focus of this report, there are some frequency based methods which are quite effective and popular. In particular, Phase Vocoder is widely used, which analyses the spectral content of the signal using Short-Time Frequency transform. The crux of the method lies in the manipulation of the phase information while preserving its the magnitude spectrum, the vocoder can stretch or compress the signal in time without changing its pitch, allowing for time-scale modification.

In this report, the time-scale methods OLA and WSOLA are implemented in python. These methods will be discussed in detail in the later sections.

## 1.2 Scope

Our main objective is to reduce the time scale of a sample without changing the pitch of it. The motivation of that could be to simplify the process of creating remixes in music production for instance. Moreover, TSM could be use for synchronization in movie production where TSM aids in dialogue and sound effects synchronization with visual elements, especially when scene duration changes during post-production. Another more specific potential algorithmic use could be that it can be used in a similar manner like Dynamic Time warping (DTW) algorithms. For example, if you have audio signals of certain phonemes being articulated by different speakers at varying speeds, to analyse them better, you can use TSM algorithms to ensure in similar time scale so that they are more appropriate for measuring similarity between the given temporal sequences. However, it must be mentioned that TSM is ideal for real-time audio processing, offering low latency and the ability to handle streaming audio, making it suitable for live broadcasts and interactive applications where maintaining pitch and quality is crucial. In contrast, DTW is better suited for offline scenarios, excelling in precise alignment and handling variable speeds.

**Working principle of TSM**

TSM work in three steps: Signal Decomposition, Frame Relocation and Signal Reconstruction. This three steps are fundamental to perform a TSM, as depicted in Figure 1 in [DM16].

# 2 Implementation

The implementation of these methods were done using Python and more especially Jupyter. The project code is on the github repository For our methods, the frame size is kept as 50ms and we compute the analysis hop size according to the relations below. Since windowing and relocating the subsequent frames is an important aspect of OLA and WSOLA, we considered relocation with no windowing as a simple compression method as something worth comparing the two methods to.

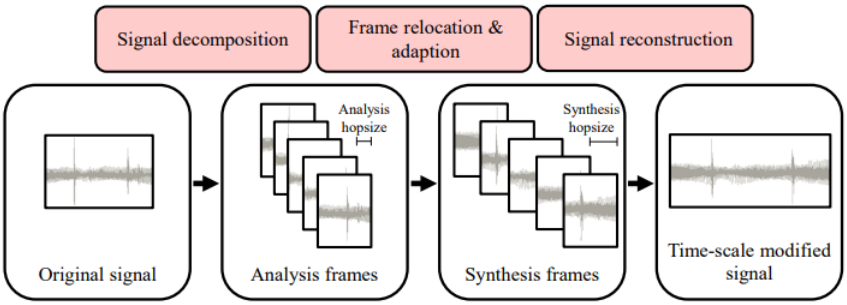



Figure 1: Principle of TSM

## 2.1 TSM-OLA

To implement TSM based on overlap-add (TSM-OLA) we need to define main parameters such as:

1. The stretching factor define by: alpha = Hs/Ha
2. The synthesis hopsize Hs = N/2 or N/4 depending of our own preferences (With N the number of samples the audio signal contain).
3. The analysis hopsize Ha which can be compute using the stretching formula.

*Outline of the Method:* The procedure consist to create synthesis frame out of analysis according to the previous properties defined. For instance if we have alpha equal to 0.5 and we have N = 4 samples. Then we will obtain Hs = 1 and Ha = 2. Hence, we will take an analysis frame every Ha and relocate them based on the synthesis frame Hs second, that way we obtain the following figure 2 by [DM16]

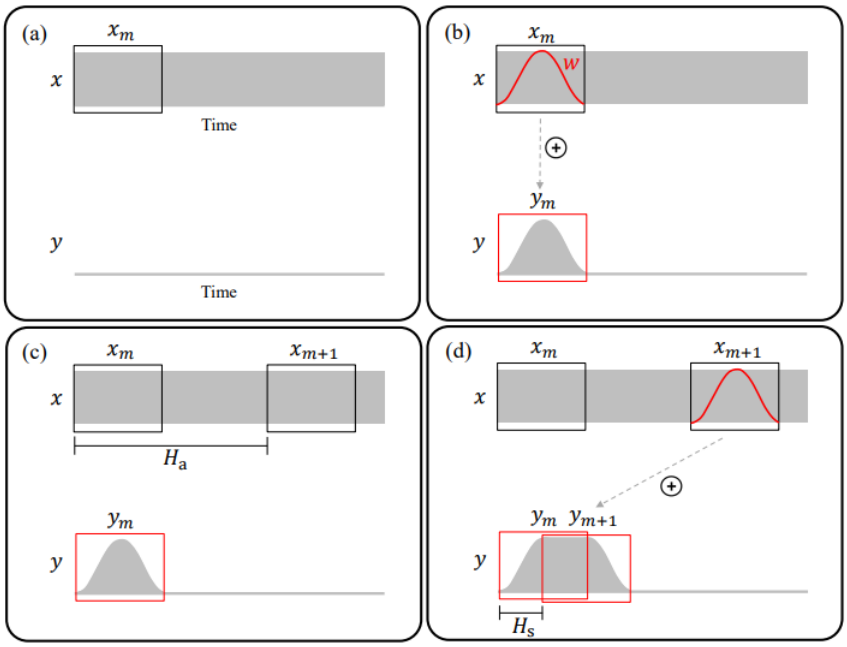



Figure 2: Principle of TSM-OLA

In (a) a frame Xm is defined and the output y relatively. In (b) we apply a window function which is always Hann

windows with the property:

$$\sum \omega(r - n\frac{N}{2}) = 1$$

Then in (c) the frame Xm+1 at a distance Ha from Xm is defined. Finally in (d) synthesis frame are applied according to the overlapp-add procedure with the specified hopsize Hs.

## 2.2 TSM-WSOLA

TSM based on waveform similiraty overlap-add (TSM-WSOLA) is pretty similar to the TSM-OLA except that there is cross-correlation involving. This cross-corellation is define by:

$$c(q, p, \Delta) = \sum q(r)p(r + \Delta)$$

With this new paramater "c" we can compute:

$$\Delta_{m+1} = \underset{\Delta \in [-\Delta_{\max}:\Delta_{\max}]}{\text{argmax}} \; c(\tilde{x}_m, x^+_{m+1}, \Delta) \; .$$

With that Delta, we can define adjusted frames described by the Figure 3 from [DM16]

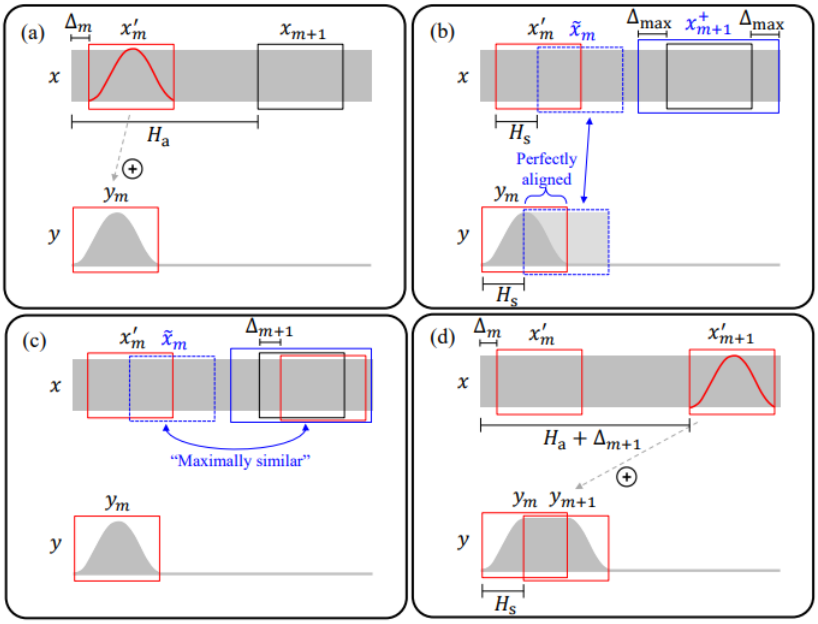



Figure 3: Principle of TSM-WSOLA

*Outline of the Method:* As you can see WSOLA principle involves several steps: (a) Beginning with the input audio signal x and its adjusted analysis frame xm, already windowed and incorporated into the output signal y. for (b) and (c) we select a frame from the extended frame region xm+1 (solid blue box) that closely resembles the natural progression (dashed blue box) of the adjusted analysis frame xm. Finally, for (d) we window and copy the adjusted analysis frame xm+1 to the output signal y.

# 3 Results

After describing how our implmented TSM algorithms works let's now talk about the result we obtained. For the results we have three differents test samples: one percussive, one non-percussive (e.g. with a pitch) and the combination of both. To see real difference between TSM algorithms we must use pitched sample to compare them that why we compute the spectrogram of a guitar sample. Indeed it contain pitch and transient (when notes are hit).

Let us see how the wavform of a Simple compression method (as referred earlier) looks:

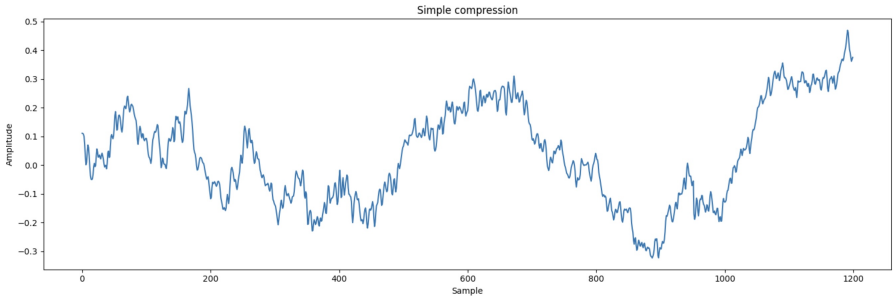



Figure 4: Guitar with Simple compression - OLA with no windowing

## 3.1 TSM-OLA

In this figure, although we notice some smoothing and the signal generally being less noisy, there are still some artifacts and the TSM-OLA doesn't sound that well especially for non-percussive samples. See the following result in the Figure 5.

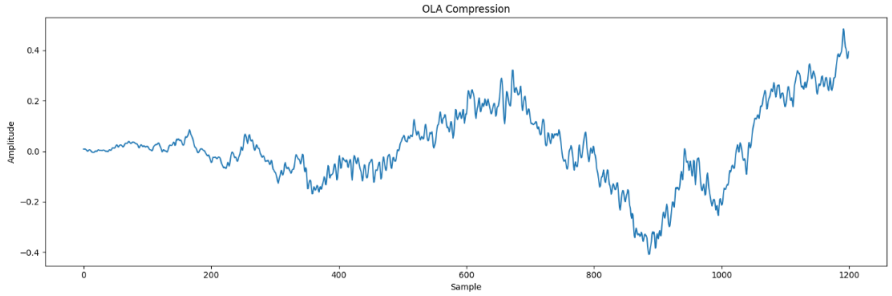



Figure 5: Guitar with TSM-OLA

## 3.2 TSM-WSOLA

Using TSM-WSOLA we observe that results are slightly better (see Figure 6) but still have some artifact. Sometimes the artifacts may even include transient doubling also called shuttering. As the frames are shifted and added to the output signal, sometimes the transient is effectively replicated.

## 3.3 Comparison of OLA and WSOLA

To compare efficiently the two methods we've implemented we will use the spectrograms of them. For reference, let us see the spectrogram with simple compression for reference.

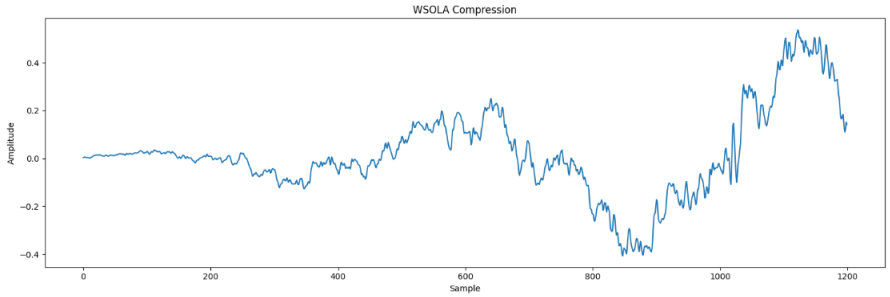
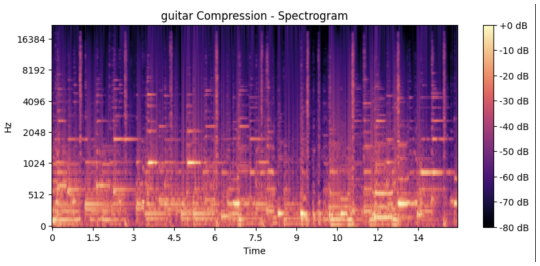


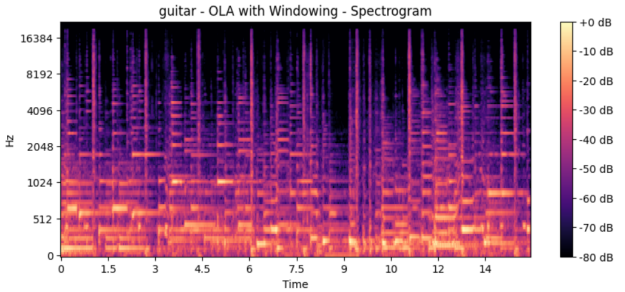

Figure 6: Guitar with TSM-WSOLA

The frequencies in the Figure 7 are clearly very smeared across time and in general very noisy.

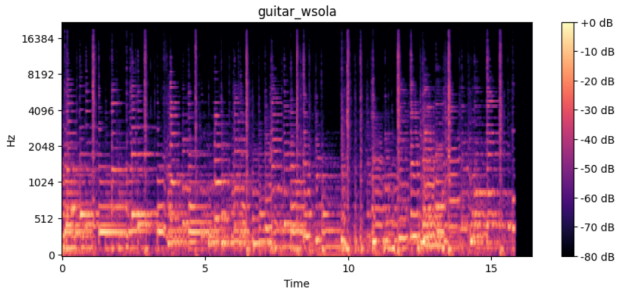
Now, let us compare the spctrogram of OLA. As you can see in the Figure 8 the spectrogram still contains some noise and it's difficult to see where the pitch it exactly located, however the frequencies are less smeared. In Figure 9, we can observe pitch more clearly, compared to the other methods. These changes in the spectrograms shows that pitch is more preserved for the TSM-WSOLA than the TSM-OLA.




(a) Guitar with Simple Compression




(b) Guitar with TSM-OLA




(c) Guitar with TSM-WSOLA

Figure 7: Comparison of Guitar Sound Processing Techniques

## 3.4 Conclusion

To conclude, upon implementing TSM-OLA, we observed certain artifacts that adversely affect the quality of non-percussive samples. Thus, TSM-OLA proves to be a cost-effective algorithm primarily suitable for percussive samples. Subsequently, our implementation of TSM-WSOLA yielded improved results in terms of pitch preservation. To further improve results, HPS and Phase Vocoder can be implemented. You can find more details on this approach in this paper by Damskägg and Välimäki. [DV17]

### 3.5 Individual Contributions

**Priyanshi Pal**

1. Implemented TSM-OLA and TSM-WSOLA in python and responsible for final submissions, as outlined in the paper [DM16]
2. Report writing

**Hercule Le Guirec**

1. Writing Reports
2. Tried and helped to implement Simple TSM methods, and tried to implement others TSM methods in MATLAB using the following papers: [DM16] [DV17]

## References


[Fit10] Derry Fitzgerald. "Harmonic/percussive separation using median filtering". In: (2010).

[DM16] Jonathan Driedger and Meinard Müller. "A review of time-scale modification of music signals". In: *Applied Sciences* 6.2 (2016), p. 57.

[DV17] Eero-Pekka Damskägg and Vesa Välimäki. "Audio time stretching using fuzzy classification of spectral bins". In: *Applied Sciences* 7.12 (2017), p. 1293.


## Appendix A Codes and Presentation document

- Time Scale Modification jupyter notebook code and wav files used in the notebook in the Repo: https://github.com/pal-priyanshi/TSM

- Presentation document and Sound files:
https://drive.google.com/drive/folders/1gv4L06NoeXb-W$_C$e9ma1v2 − 7JUVjBNXI?usp = sharing