Time Scaling

September 29, 2014

Contents

1 Problem Statement **2**

2 Sample Rate Change **3**

3 Time Stretching Algorithms (Time Domain) **3**

3.1 Overlap and Add (OLA) 3

3.2 Synchronous Overlap and Add (SOLA) 4

3.2.1 Time-domain cross-correlation 5

3.3 Pitch-Synchronous Overlap and Add (PSOLA) 5

3.4 Pitch Detection 7

3.4.1 Zero-crossing rate 7

3.4.2 Pitch Detection with Auto-correlation 9

4 Real-Time Implementation **9**

1 Problem Statement

In presenting the news, there are often times when the newscaster must increase the amount of information given ensuring that the program meets certain time constraints. On the radio this is quite prevalent when the djs have already begun to play a song and must rapidly provide information before the lyrics or main part of the music begins. For some viewers of the television programs or listeners of the radio this speech is too fast for it to be comprehensible. This is even more of a problem when the listener does not share the same mother tongue and needs a slower pace in order to understand the presenters.

If the broadcasts are recorded and played back for a listener, a simple remedy for this quick pace is to simply slow down the playback of the audio. Unfortunately this comes at a cost of severely distorting the pitch content of the sound. The slow downed playback has a much deeper sound while the sped up version has a much higher sounding pitch. An example of this is if you were to place your finger on a record to slow down the play speed noticing that the pitch becomes much lower. Figure 1 illustrates this using a sound segment that is shortened and elongated to show the change in the spectral information. On the left side of the figure the time information is displayed where, with the same sample range, the information is either compressed or elongated. Likewise the frequency information or spectra has also changed due to the compression and expansion of the original signal.

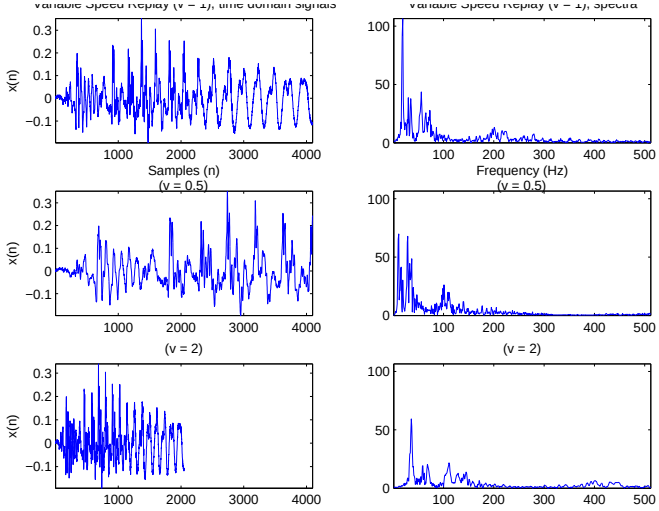


Figure 1: Time representation (left) showing variable speeds and the effect on the signals corresponding spectra(right).

Time scaling or time stretching is the process of slowing down or speeding up an audio signal without changing the pitch. The applications of time-scaling are numerous with an example given above as well as reading text for the blind, and learning a foreign language. Time scaling is performed by dividing the signal into fixed overlapping frames. These overlapping frames are then shifted according to the overall goal (speeding up or slowing down) and combined to give a reconstructed output.

The goal of this project is to construct a time scaling algorithm. The algorithm will at first be simply implemented using basic techniques with each assignment adding an increased difficulty. Toward

the end of the class this algorithm will be implemented in real time and used in conjunction with the other groups Wim De Vilder filter.

2 Sample Rate Change

As alluded to in section 1 a naive approach to time scaling would be to simply speed up or slow down the audio. This can be accomplished with what is referred to as sample rate conversion. While we will not focus on sample rate conversion it is important to understand the overall effect on the frequency information when a signal is stretched in the time domain. Therefore we look to resample a signal and observe the effect this has on the pitch of the speech.

In MATLAB sample rate change can be performed with the

```
>> Y = resample(X,P,Q)
```

command.

- Use the resample command in MATLAB to change the sample rate, try both increasing and decreasing the sample rate.
- View the spectra of the same from all the signals. How did this change with resampling the signal?

3 Time Stretching Algorithms (Time Domain)

After noticing the problems that are introduced when the sample rate is changed in the signal we look to a way to preserve the pitch information while still adjusting the playback speed. This is done by implementing a time stretching algorithm on the collected data.

3.1 Overlap and Add (OLA)

A very basic algorithm for time scaling is accomplished by first dividing the signal into overlapping blocks of a fixed length N as shown in figure 2. The original blocks are separated with a time shift of S_a samples. The blocks are then repositioned with a time shift of $S_s = \alpha S_a$. The overlapping block are now weighted by a fade-in and fade-out function and summed sample-by-sample. Finally the new blocks are concatenated in order to produce a time stretched signal.

- Load a speech file into MATLAB and separate it into frames with an overlap of 50%.
- Reposition the block with a time shift that either increases or decrease the speed. **NOTE** Be careful with how far you shift the signal, if the new time scale is larger than the frame size ($S_S \geq N$) there will be no overlap which will create discontinuities in the output signal.
- Listen to the output and observe the spectra. How does it sound? What happened with the spectra?

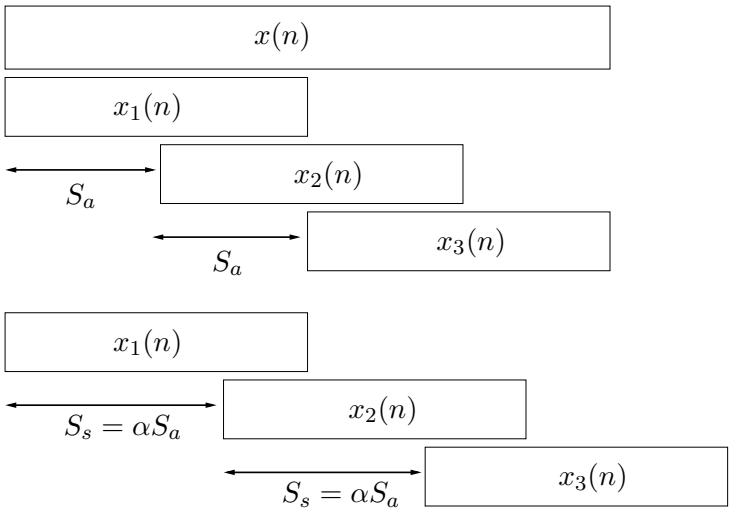


Figure 2: Time Stretching : Overlap Add

3.2 Synchronous Overlap and Add (SOLA)

Synchronous Overlap and Add synthesis is very similar to that of the general OLA procedure that was presented previously. The main difference between the two is that SOLA relies on correlation techniques to improve on the time-stretching algorithm. When the blocks are shifted by the time factor α similarities in the area of the overlap intervals are searched for a discrete-time lag of maximum similarity. This point of maximum similarity of the overlapping blocks are then weighted by a fade-in fade-out function and again summed sample-by-sample. A depiction of this is given in figure 3.

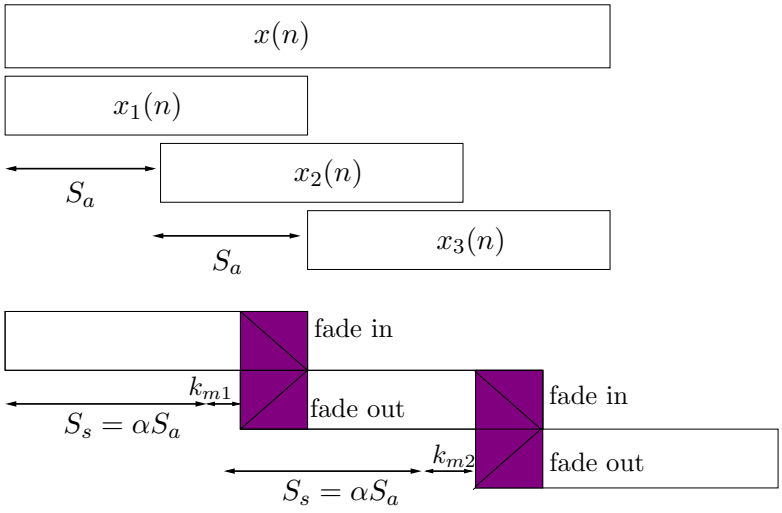


Figure 3: Time Stretching : Synchronous Overlap and Add

3.2.1 Time-domain cross-correlation

The cross-correlation is a way to determine the similarities of two waveforms over a time-lag. It is used extensively in signal processing to find smaller wave forms in a longer sample which leads to pattern recognition. We will use this cross-correlation information in order to find the place with maximum similarity between the overlap intervals of the time shifted signal. The cross-correlation is found by

$$r_{xL1}r_{xL2} = \frac{1}{L}\sum_{n=0}^{L-m-1} x_{L1}(n)x_{L2}(n+m), 0 \leq m < L \tag{1}$$

where $x_{L1}(n)$ and $x_{L2}(n)$ are the segments of $x_1(n)$ and $x_2(n)$ in the overlap interval of length L. We now use the index that corresponds to the maximum correlation as a way to overlap the signals as shown in figures 4,5.

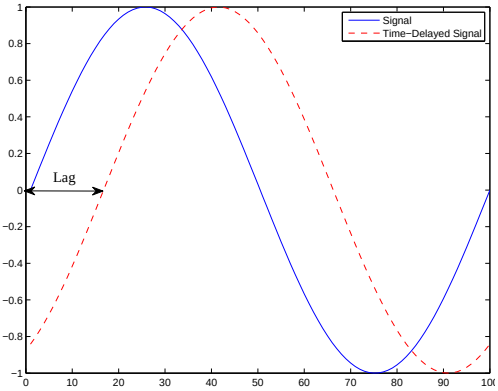


Figure 4: Orignal signal and a time-delayed version of itself.

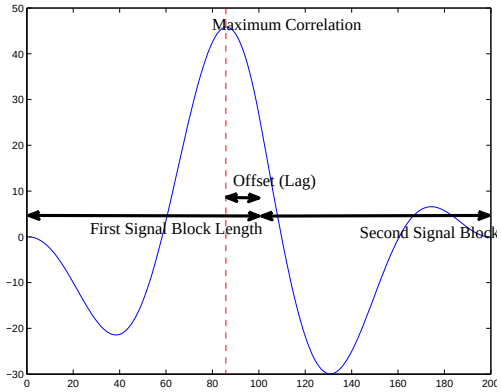


Figure 5: Cross-correlation between orignal signal and time-delayed version of iteself.

Goal:

- Design and evaluate a correlation routine between frames segments of a speech signal.
- Implement the SOLA algorithm using the correlation function as well as the previously designed OLA method.

3.3 Pitch-Synchronous Overlap and Add (PSOLA)

Pitch-synchronous Overlap Add uses the hypothesis that the input sound is characterized by a pitch. It exploits the knowledge of the pitch to correctly synchronize time segments avoiding pitch discontinuities. The PSOLA algorithm is essentially divided into two steps: the first phase analyzes the segments of input sound and extracts the pitch information, and the second phase synthesis a time stretched version by overlap and adding time segments extracted by the analysis phase.
Analysis algorithm :

1. Determine the pitch period of the input signal and of the time instants (pitch marks) t_i.

2. Extract segments centered at each pitch mark t_i by using a Hanning window with length $L_i = 2P(t_i)$. This two pitch period ensures that a fade-in and fade-out can take place.

Synthesis algorithm :

1. Choose analysis segment that minimizes the time distance $|\alpha t_i - t_k|$.
2. Overlap and add the selected segments. Notice that this will results in some input segments being repeated $\alpha > 1$ and some segments being discarded $\alpha < 1$.
3. Determine the next time instant t_{k+1} where the next synthesis segment will be centered.

The PSOLA algorithm is depicted in figure 7.

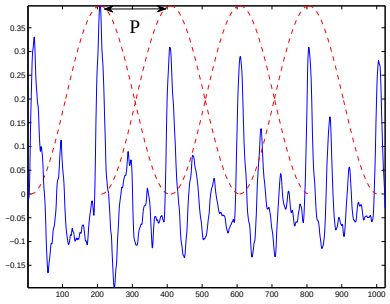


Figure 6: PSOLA : Pitch analysis and block windows.

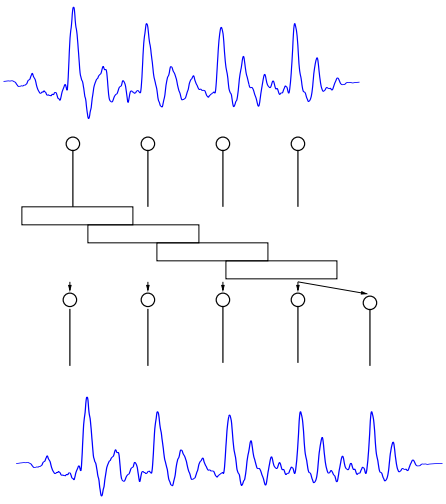


Figure 7: Depiction of the PSOLA algorithm.

Goal: Implement the PSOLA algorithm using the pitch detection methods previously discussed. **Pay close attention to unvoiced segments!**.

3.4 Pitch Detection

Pitch is an attribute that is associated with the frequency of a sound. Depending on the frequency of the signal it is classified to a certain pitch. While the two are not equivalent the usage of pitch information will play a critical role in improving on the previously discussed time-stretching algorithms.

3.4.1 Zero-crossing rate

Using the zero-crossing rate is rudimentary pitch detection algorithm. It works well in the absence of noise and is discussed here for its simplicity and computation. The zero-crossing rate determines how man times the waveform crosses the zero-axis in a certain time. Figure 8 shows a 100 Hz sine wave on a measurement interval at $20ms$. There are 4 zero-crossing throughout the sample frame.

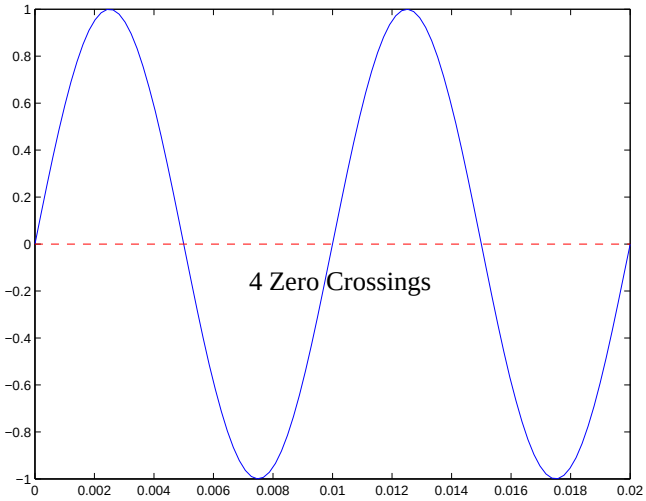


Figure 8: Zero-crossings for a $f_s = 100Hz$ sine wave.

We define a function sign{} that returns a +1 or 0 depending upon whether the signal is greater than zero or not. The zero-crossing rate (ZCR) may then be given as

$$ZCR = \frac{1}{N}\sum_{n=0}^{N-1} |\text{sign}\{s(n)\} - \text{sign}\{s(n-1)\}| \tag{2}$$

where the $\frac{1}{N}$ provides the normalization to find the crossing rate.

In order to calculate the fundamental frequency f_f of the waveform in the frame we use the following formula

$$f_f = \frac{\text{ZCR} \times f_s}{2}. \tag{3}$$

The ZCR approach works well for pure speech tones as well as some speech segments. However if we look at a more complex wave form (figure 9) of a speech signal the ZCR given may not accurately reflect the true crossing rate.

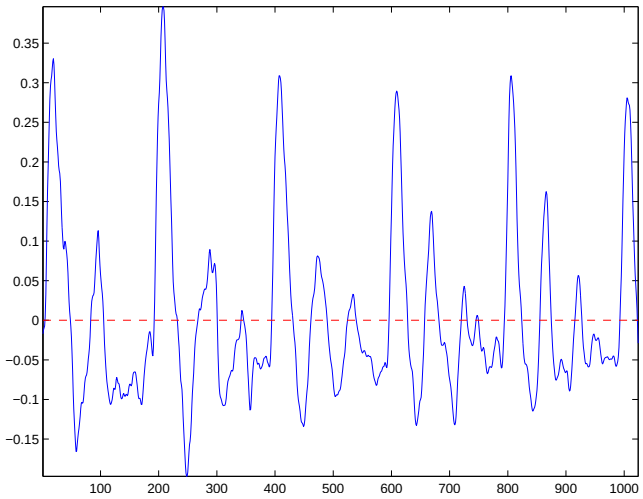


Figure 9: Zero crossings for a complex waveform.

Therefore we can modify the zero-crossing rate with the use of a threshold. A threshold is proposed in the form of

$$\sigma = \gamma \frac{1}{N} \sum_{n=0}^{N-1} |s(n)| \tag{4}$$

where $\gamma \approx 1.2$. We now define a new signal $s_p = s(n) - \sigma$ and instead of counting the ZCR we count only the negative to positive transitions ($\mathrm{ZCR_p}$). This gives a positive rate transition which corresponds to a half-period fundamental frequency of

$$f_{f_p} = \mathrm{ZCR_p} \times f_s. \tag{5}$$

Likewise a negative displacement to the ZCR is given as $s_n = s(n) + \sigma$. Similarly to the positive displacement only the positive to negative transitions are counted resulting in another half-period fundamental frequency of f_{f_n}. Finally the true fundamental frequency is given as the mean of the two frequencies or

$$f_f = \frac{f_{f_p} + f_{f_n}}{2}. \tag{6}$$

In the presence of noise these techniques become even more difficult as there is often severe jitter around the zero crossing point. Another concept of a threshold-crossing rate (TCR) can therefore be introduced that takes into account the amount of noise that is present in the system.

Goal:

- Implement a zero-crossing algorithm for a signal.
- Observe the ZCR during speech and silent periods.
- Adjust γ in (4) and observe the effects on the f_f.
- Add noise to the signal and try to implement a TCR in order to avoid false-positives in the ZCR.

3.4.2 Pitch Detection with Auto-correlation

Another, more robust, way of performing pitch detection is to utilize the auto-correlation of a signal. The auto-correlation is similar to the cross-correlation introduced in 3.2.1 with the difference being that the cross-correlation is performed with the same signal. In order to accurately determine the pitch we take windows of the signal that are at least as twice as long as the longest period we wish to detect.

As the shift in the auto-correlation function begins to reach the fundamental frequency we will see a maximum in the auto-correlation function. This maximum can therefore be thought of as the pitch period. Therefore by using the auto-correlation of the signal we are able to extract the pitch period of the signal. The auto-correlation is found by

$$r_{xL1}r_{xL1} = \frac{1}{L}\sum_{n=0}^{L-m-1} x_{L1}(n)x_{L1}(n+m), 0 \leq m < L \tag{7}$$

where $x_{L1}(n)$ is the segment of $x_1(n)$ with interval of length L.

Goal:

- Use previously developed correlation function to perform the auto-correlation of the frames of a signal.
- Compare the pitch-detection of the auto-correlation compared to that of ZCR.

4 Real-Time Implementation

For the previous implementations of the time-stretching algorithms we have used recorded signals where the statistics have been known during the whole processing period. In real-time implementations we have signal statistics that are unknown and changing throughout time. Therefore if we try to time-stretch the signal by making it faster, we do not have the ability to grab the future frames which makes speeding up the signal impossible. However in order to slow down the signal we are simply, in some cases, repeating parts of the known signal, therefore time-stretching used to decrease the speed of the speaker is possible.

MATLAB comes with the availability to do real-time implementations by way of the built in package simulink. Use the simulink package in order to implement your time-stretching algorithm in real-time.

After you have a working time-stretching algorithm in the simulink environment you will merge this with the other groups Wim De Vilder filter. Therefore it is recommended that before the initial design process in simulink the two groups discuss with each other what input and output parameters the other group needs in order to construct a working model.