By Jinho Kim
A&S Undergraduate Honors Thesis 2013
Advised By Professor Sergio Alvarez
Computer Science Department, Boston College

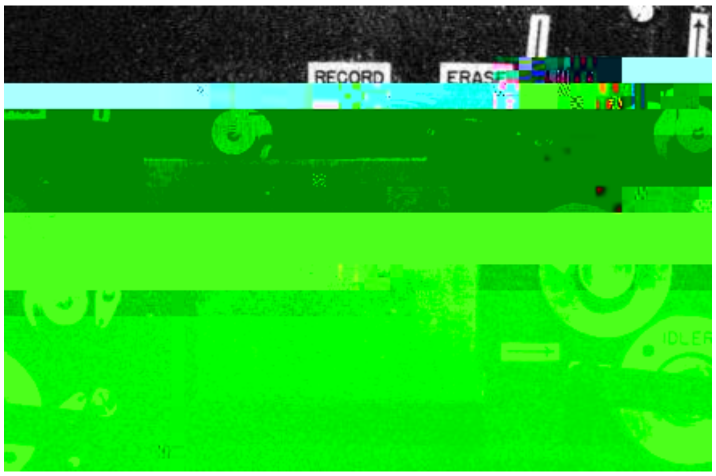
Abstract

Musical notes are acoustic stimuli with specific properties that trigger a psychological perception of pitch. Pitch is directly associated with the fundamental frequency of a sound wave, which is typically the lowest frequency of a periodic waveform. Shifting the perceived pitch of a sound wave is most easily done by changing the playback speed, but this method warps some of the characteristics and changes the time scale. This thesis aims to accurately shift the pitch of musical notes while preserving its other characteristics, and it implements this in real time on an Android device. There are various methods of detecting and shifting pitch, but in the interests of simplicity, accuracy, and speed, a three step process is used. First, the fundamental pitch of a stable periodic section of the signal is found using the Yin pitch detection algorithm. Secondly, pitch marks that represent the local peak of energy are found, each spaced out by roughly one period (inverse of the fundamental frequency). Lastly, these marks are used in the Pitch Synchronous Overlap and Add (PSOLA) algorithm to generate a new signal with the desired fundamental frequency and similar acoustical characteristics to the original signal.

Table of Contents

1. Introduction

1.1 A Brief History of Pitch Shifting



As the capacity of memory improved, digital devices became available to become powerful tools in sound processing. The Eventide H910 was released in 1975 as the first commercial harmonizer[3]. This was the first device that could digitally shift pitch whilst preserving time length, and it was instantly put to use in recording studios. For example, this device was first used to downward shift a sped up version of the sitcom *I Love Lucy* in order to fit in more commercials.

Figure 2

Pitch shifting is now a commonly used sound effect that is included in most digital sound processing libraries. As pitch shifting algorithms and their implementations have improved, this effect is now achievable in real time, and it has seen widespread use in the music

converts a normal voice into a robotic sounding one by pitch shifting the original signal to a perfect pitch, as well as artificially altering some of the vocal characteristics.

1.2 Main Goals

The purpose of this thesis was to successfully study and implement pitch shifting while meeting specific constraints:

1) Preservation of the time scale   the shifted playback audio should have the same duration as the recorded audio.
2) Preservation of the original acoustical characteristics   the shifted audio should sound the same as the original audio except for the pitch shift.
3) Processing in real time   the processing rate should be faster than the recording rate.

The implementation was done in Android for demonstrative purposes, and a variety of algorithms were explored (see section 2) to find the an appropriate method in order to achieve the above mentioned constraints.

1.3 Main Challenges

As there are so many algorithms available, it was difficult to choose which ones to use for this project. After much research, I decided to pitch shift with PSOLA (section 2.5), pitch detect with YIN (section 2.5.1), and pitch mark with the two-phase algorithm (section 2.5.2).

I first implemented PSOLA using MATLAB for proof of concept. Once pitch shifting was achieved and the sound quality was deemed acceptable, I shifted my focus towards Android.

[redacted]

Figure 3

1.4.2 Frequency and Period – Pitch of a Sound

The rate at which the vibrations oscillate is known as the frequency and is typically measured in Hertz (Hz), which is the number of cycles completed per second. The inverse of frequency is the period (wavelength), which is the length of time it takes to finish one cycle, as shown in **Figure 4** below.

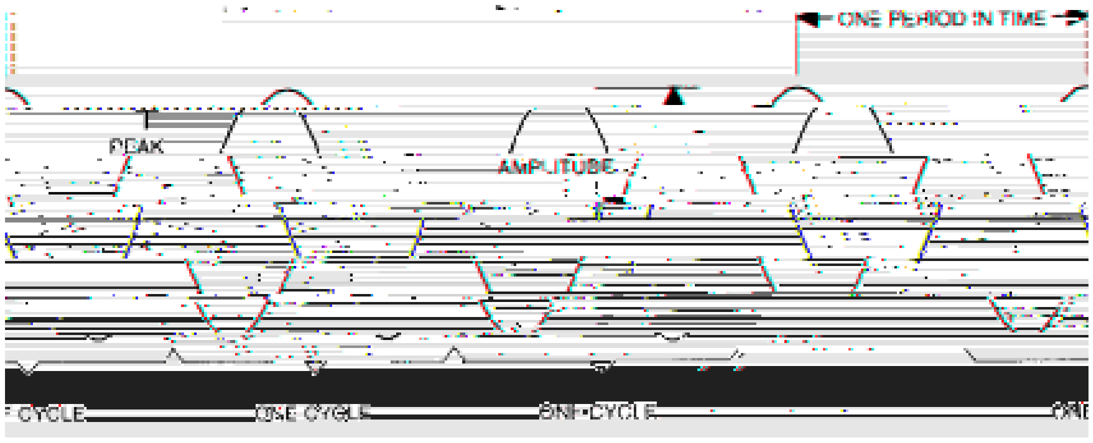


Figure 4

frequency of a sound wave, so a high pitched squeak of a mouse would be characterized by rapid vibrations in the air, while a low pitched call of a whale would be characterized by slower vibrations in water. Different animals have varying ranges of sensitivity to pitch. For example, h ears are able to detect sounds from a frequency range of roughly 64Hz to 23kHz, while dogs can hear from a frequency range of 67Hz to 45kHz precisely why dog whistles cannot be heard by humans[6].

1.4.3 Harmonics and Complex Waves – Tone Quality of a Sound

The previously shown sound waves were perfect sine waves A perfect sine wave is a pure tone of a single frequency and sound uncharacteristically dull[5]. Most musical tones are complex and have multiple sine waves of different frequencies in them. The lowest frequency in the tone is known as the *fundamen22(s h-)2(w a-)3(the(oun is h-)2(w a-)3(the)4*

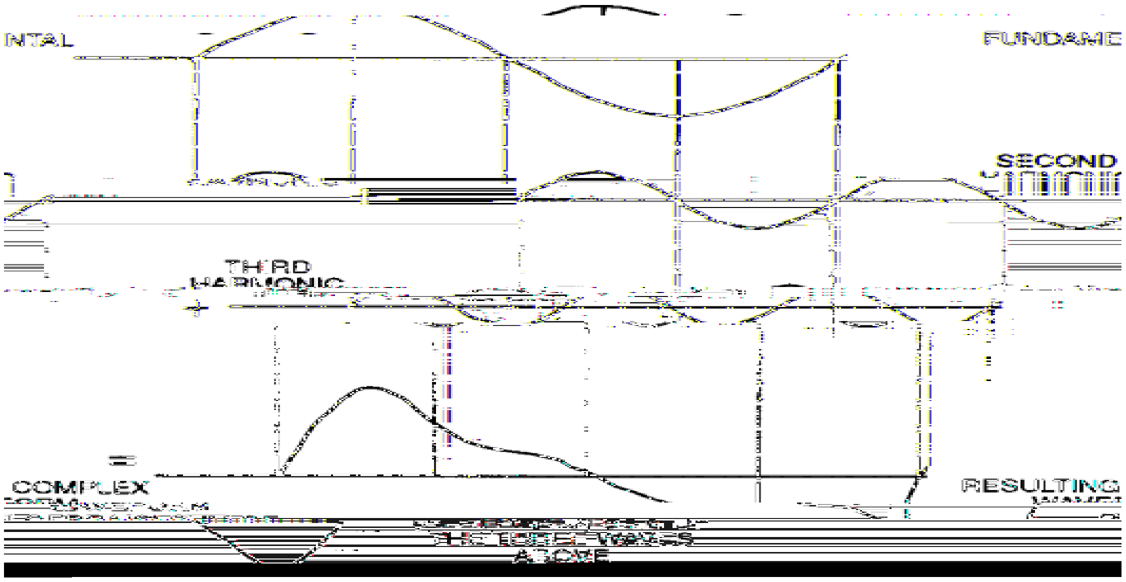


Figure 5

is land as(o1 90.024 93TBJuTTJETBTvJ93-4(22.7 659ID 2>B6h3lB50 /P 591edou1Ta2n44(93-18 Tmn1T Tmf Tcfd.fn111.r-61

2. Pitch Shifting Algorithms

2.1 Overview

The Phase Vocoder was one of the earliest methods for pitch shifting with time preservation, and it was proposed in 1966 by Flanagan and Golden[8]. This algorithm was first implemented 10 years later by Portnoff in 1976 using the Fast Fourier Transform[9]. Digital Sound Processing has exploded in growth since then, and a number of pitch shifting algorithms have been invented and improved.

2.2 Time Domain versus Frequency Domain

Pitch shifting algorithms are typically separated into two groups those that operate in time domain, and those that operate in the frequency domain. Sound processing in the time domain is done using the familiar representation of the change of amplitude over time. The frequency domain is a different representation, where the x axis is frequency and the y axis is the amplitude. This representation can show the strengths of the individual frequency components present in the signal, which would be difficult to see or detect in the time domain. **Figure 6** shown below depicts a single complex signal in the time domain. This signal is composed of the union of 2 different frequencies, which are easily separated in the frequency domain, as shown by the 2 distinct lines in the right graph[10].

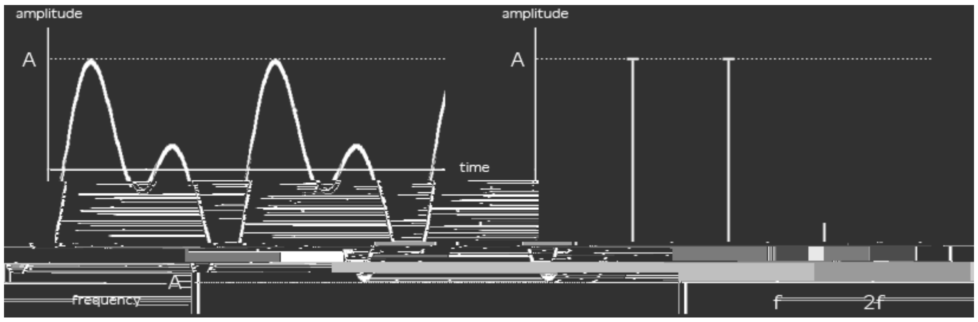


Figure 6

2.3 Pitch Shifting in the Frequency Domain

Sound manipulation in the frequency domain is usually done by breaking up the signal into small overlapping segments (windows) and then processing each segment separately. This is necessary because the frequency of a signal can change over time, which would result

Sound manip21/9oa]sthe frequ45oadsund mIJE44(eom350.334nt ssrR4(sult)-)-9(wg47(qupa(H (sz1 0 4(s=p)10(v)20av)25(sf)3a

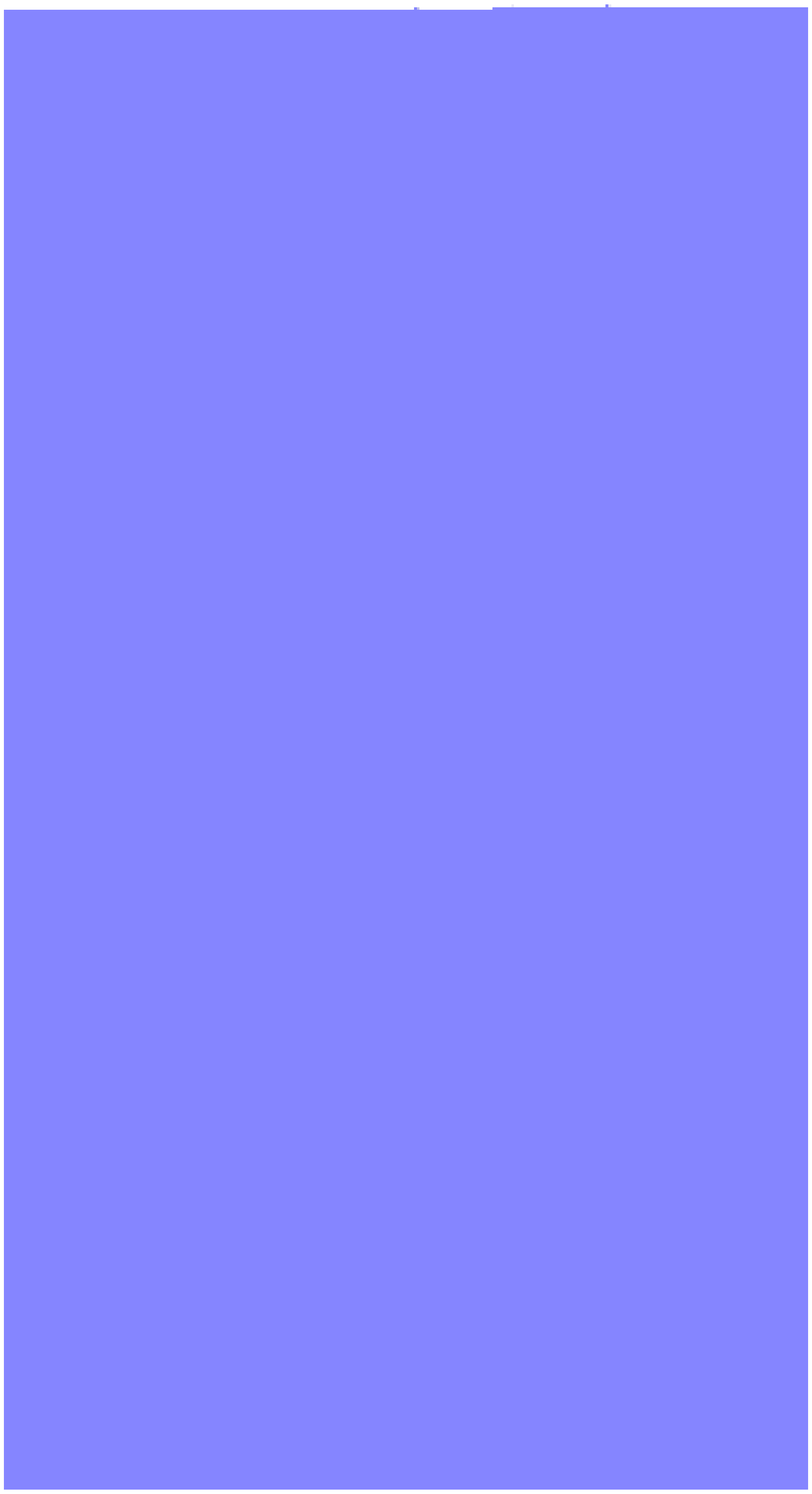


Figure 7

2.3.1 Phase Vocoder[12]

As mentioned previously, the Phase Vocoder algorithm is a classic algorithm in pitch shifting, as it was one of the first methods to be able to pitch shift while preserving the time length of a signal.

As phase modification is a major part of this algorithm, it is important to know what phase is. The phase of a wave at any given point is the measure of the progression of the cycle. This progression is measured in degrees, with 360 degrees being a complete cycle. The cycle starts from the baseline (0^o), increases to the peak (90^o), goes back down to the baseline (180^o), sinks to the valley (270^o), and then finally returns back to the baseline (360^o or 0^o). **Figure 8** below shows the various stages of phase, w-7(a.p7D281 198.2c,af( )] idh)

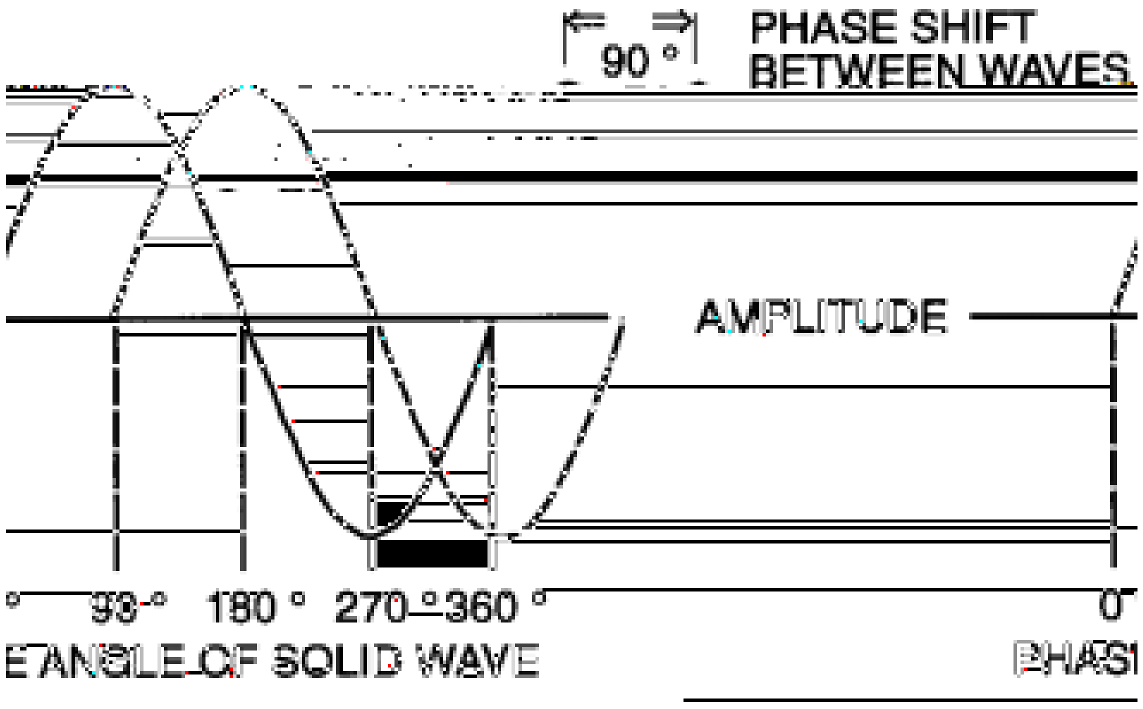


Figure 8

The basic steps of the algorithm are as follows:

1) Time-shift overlapping blocks (windows) to alter the length of the signal while preserving the pitch. As shown in **Figure 8**, shifting the blocks forward in time elongates the length, and shifting them backwards in time shortens the length.
2) Take the FFT of each overlapping part of the blocks and modify the phase of each bin to ensure phase continuity.
3) Resynthesize blocks using the inverse FFT
4) Resample blocks to restore the original time length while changing the frequency.

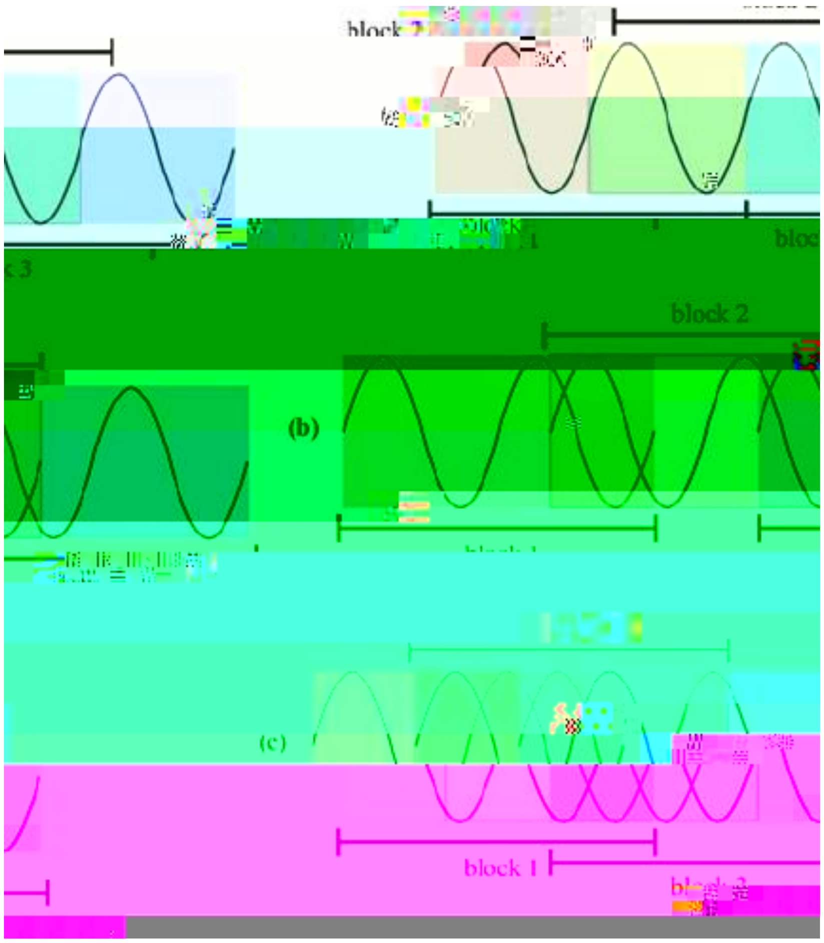


Figure 9

[12]

peak in magnitude at the time lag that corresponds to the pitch period of the fundamental

$$R[k]=\sum_{m=0}^{N-k-1} x[m]x[m+k]$$

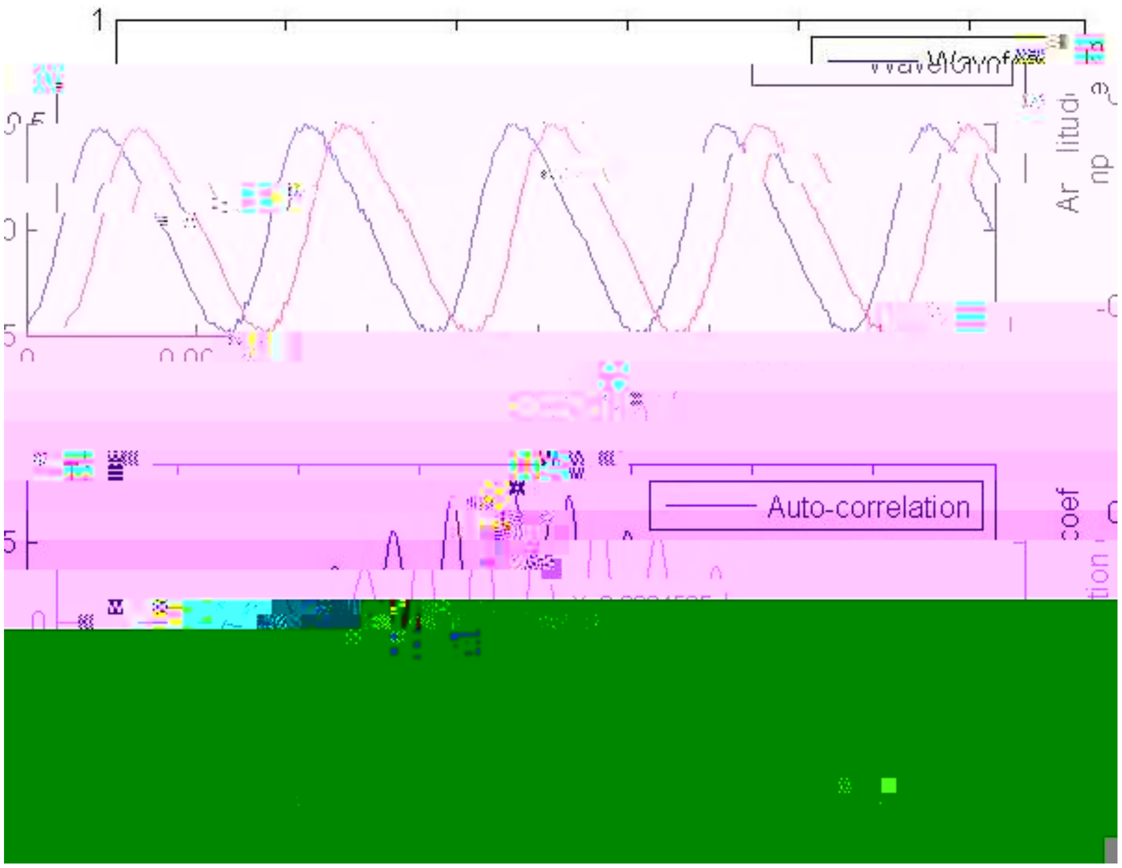


Figure 10

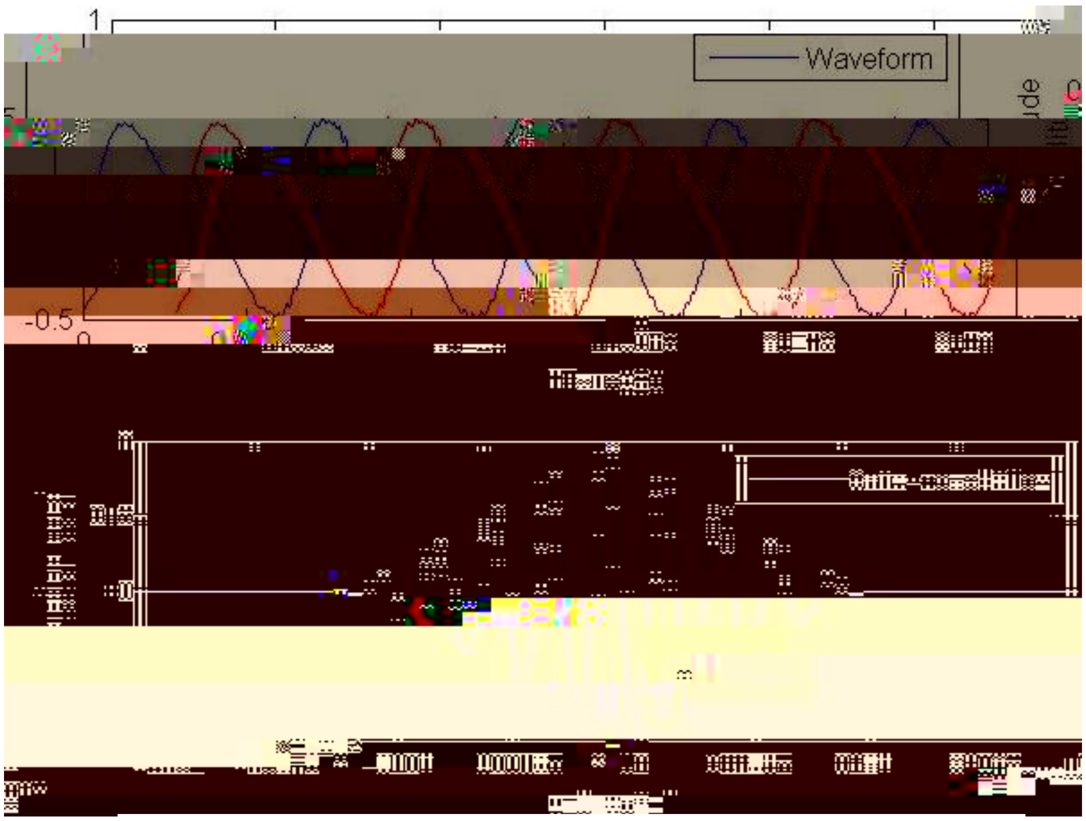


Figure 11

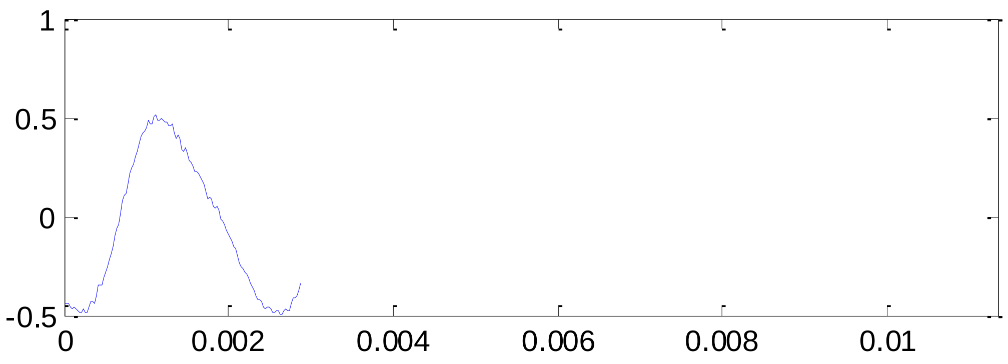


Figure 12

Basic autocorrelation by itself is not reliable or accurate enough, however, as it is sake of simplicity, only peaks will be discussed, as the same algorithm can be used on a negated signal to find the valleys. The first step of this phase of the algorithm is to find the global maximum of the waveform and make this the first pitch mark. This mark is circled below in

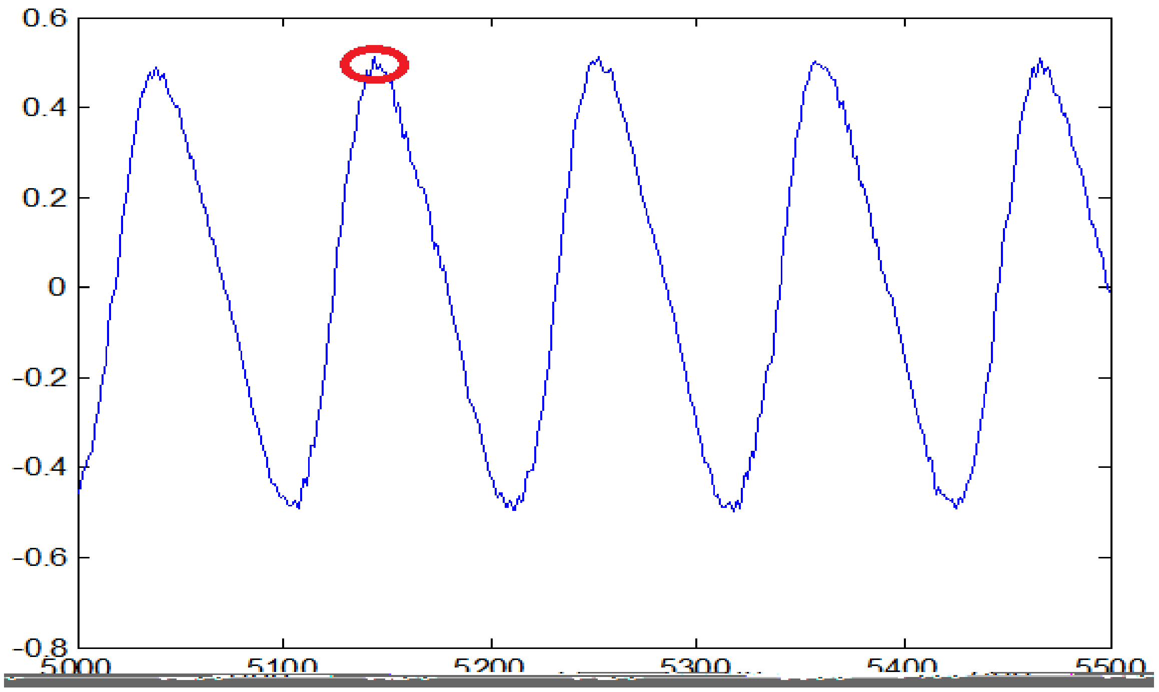


Figure 13

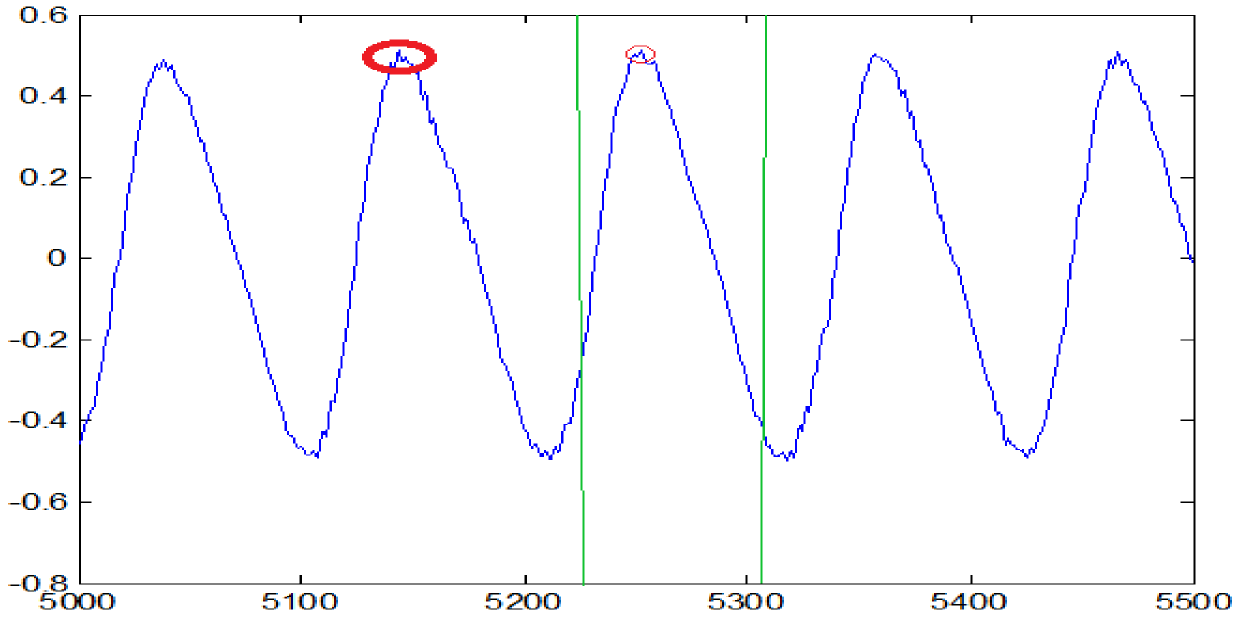


Figure 14

This process is repeated until it reaches the end of a signal or an unperiodic section of the signal, where the pitch detection was unable to find a fundamental frequency. This process can also be repeated towards the left of the global maximum to fully m

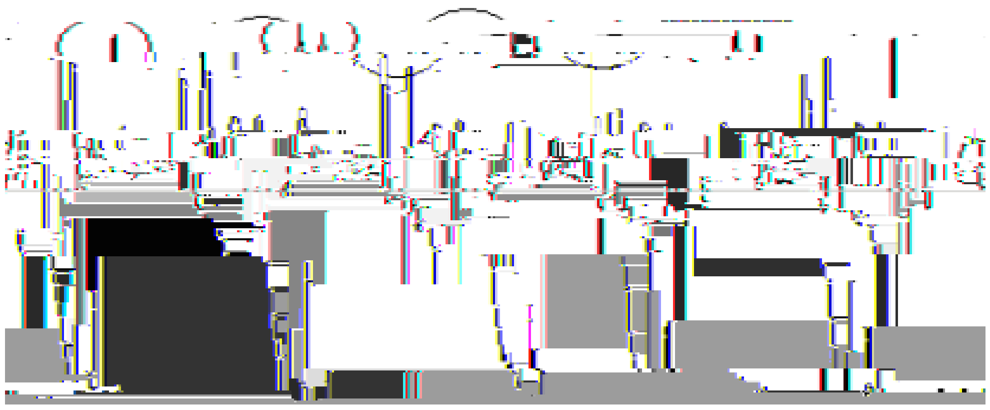


Figure 15

The second phase of this algorithm attempts to fix this error by first finding multiple candidates within each search region and then using dynamic programming to find the best sequence of pitch mark candidates.

Each pitch mark candidate is assigned a state probability that is associated with its relative magnitude and is given by the following equation:

$$\bar{s}_i(j) = \frac{h_i(j) - h_{min}}{h_{max} - h_{min}}$$

Where j ranges from 1 to n (the number of pitch mark candidates), $h_i(j)$ is the height of candidate j in region i, and h_{max} and h_{min} are the max and min of the signal.

Each search region (except the last one) is also assigned an nxn matrix to represent the transition probabilities of the n candidates in a given search region to the n candidates in the next. The transition probability is associated with the similarity of the distance between pitch marks and the detected fundamental period found in the previous pitch detection process and is given by the following equation:

$$\bar{t}_i(j_1, j_2) = \frac{1}{1 + \beta \left| f - \frac{f_s}{d} \right|}$$

Where f is the detected fundamental frequency, f_s is the sampling rate, d is the distance (in sample points) between candidates j_1 and j_2, and is a fine tuning parameter (I left this at 1).

These probabilities could then be used to assign accumulated probabilities from the first pitch mark (typically the global maximum). The first pitch mark would have an accumulated probability of 1. The rest of the accumulated probabilities of pitch mark candidates are calculated dynamically; the candidate in the previous search region that gives the highest sum of accumulated probability and transition probability to that particular candidate in the current search region is found, and the accumulated probability is calculated by adding that sum to the state probability of the current candidate. The previous candidate used is saved for each new candidate calculated, so that the optimal path can be backtracked at the end. In **Figure 16** below, the number of candidates per region is limited to two, and the optimal path of pitch marks can include any combination of white and black circles (representing different peaks)[16].

Figure 16

The effectiveness of this two-phase algorithm was tested by comparing it (using 3 candidates per search region) to the simple one-phase algorithm that only looked for one max peak. I tested these two algorithms with 4 different types of frequencies; low (100Hz), medium (200Hz), high (400Hz), and moving (ranging from 100-400Hz). The accuracy of these algorithms was measured by the deviation of consecutive pitch mark differences from expected periods. The variance of the differences between consecutive pitch marks was found by calculating the average squared difference between the expected period (the detected fundamental period in the previous step) and the detected period (the difference

2.5.3 Synthesis Pitch Marks

Once the pitch marks have been found, the synthesis pitch marks need to be placed. These synthesis pitch marks are spaced relative to the distances between the analysis pitch marks, but synthesis marks can be affected by a shift factor. If the shift factor *B* is given by (desired frequency) / (original frequency), the distance between consecutive synthesis marks should be the (current period) / *B*. If the desired frequency is higher than the original, then the period will be smaller, and thus the synthesis marks will be closer together than the analysis marks. This would eventually lead to a reconstructed signal with a higher frequency in the next step, with the opposite occurring if the desired frequency is lower. The first synthesis pitch mark starts at the first analysis pitch mark, but each synthesis mark thereafter is (current period) / *B* samples away. In **Figure 19** below, the analysis step finds 4 pitch marks, but the synthesis step results in 5 synthesis marks, which leads to a higher pitch[17].

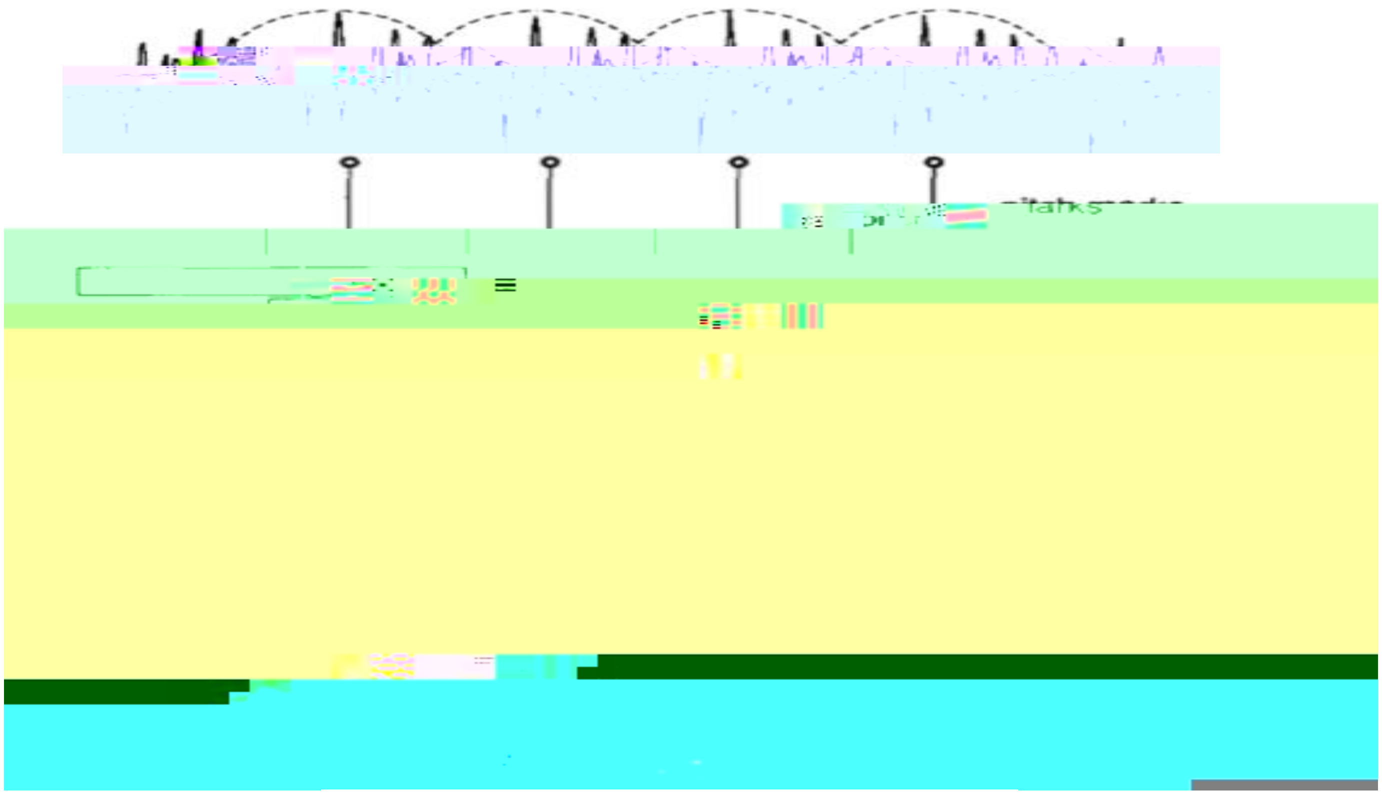


Figure 19

2.5.4 Overlap and Add

Once the synthesis pitch marks are placed, the algorithm continues with the final step the reconstruction of the shifted signal. The shifted signal starts off empty, and the algorithm starts on the initial synthesis mark and then does the following:

1) Find the closest pitch mark p to current synthesis mark s_i
2) Obtain a windowed segment centered on p with the width being 2 * pitch period
3) Multiply this window by a hanning window of the same size to taper off ends
4) Overlap and Add this window to the shifted signal, centering it at current synthesis mark s_i.
5) Repeat for the next synthesis mark s_{i+1}.

The number of periods in the resulting shifted signal will correspond to the number of synthesis marks, as opposed to the number of analysis marks. If these numbers differ, the pitch will have successfully shifted.

3. Android Implementation

This pitch shifting project was implemented in Java on my personal phone, a 2-year-old Android device (HTC Droid Incredible) with a single core 1GHz processer. Although it has fairly weak performance compared to the quad-core smartphones of today, my phone was able to handle the sound processing in real time well enough.

3.1 Android Data Flow

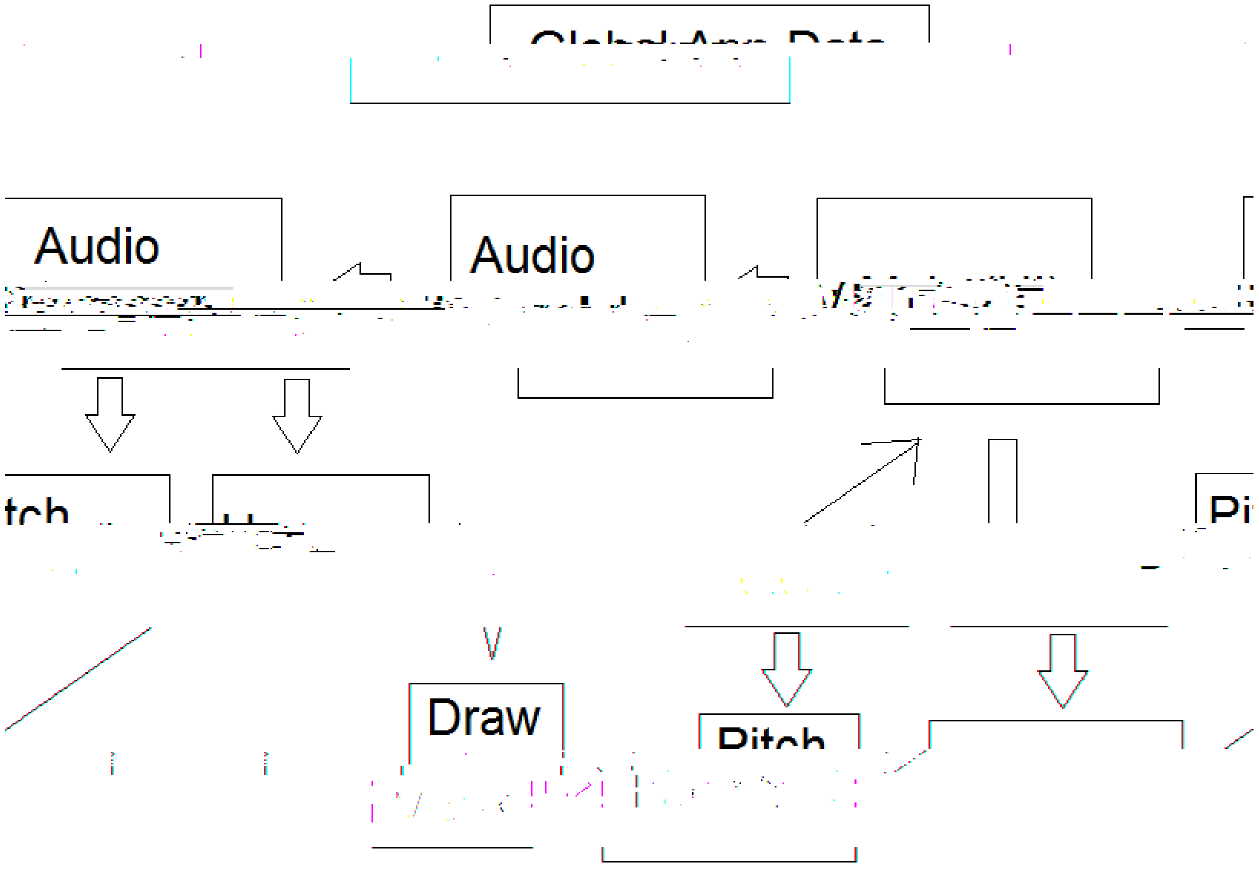


Figure 20

The flow mostly revolved around the Main, Audio Recorder, Audio M1Qe MasouD9.54lin,aso24 564.1 T

PSOLA algorithm. The amount to shift is determined by either the user or the Harmonizer class (experimental). This shifted signal data is then sent back to the Main thread, which then passes it to the Draw thread to display 1 of 4 options:

1) Waveform

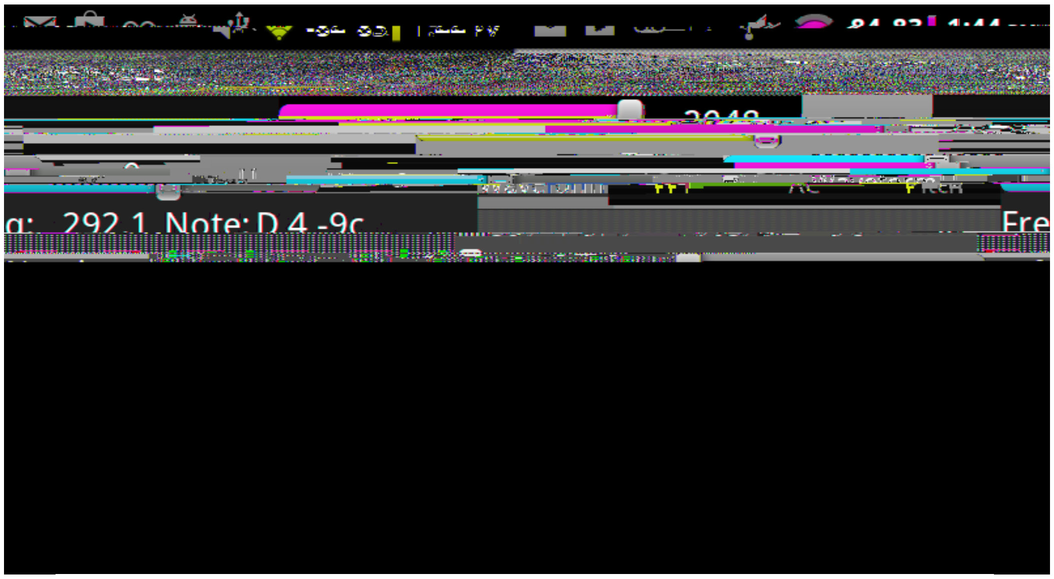


Figure 21

2) FFT

Figure 22

3) YIN

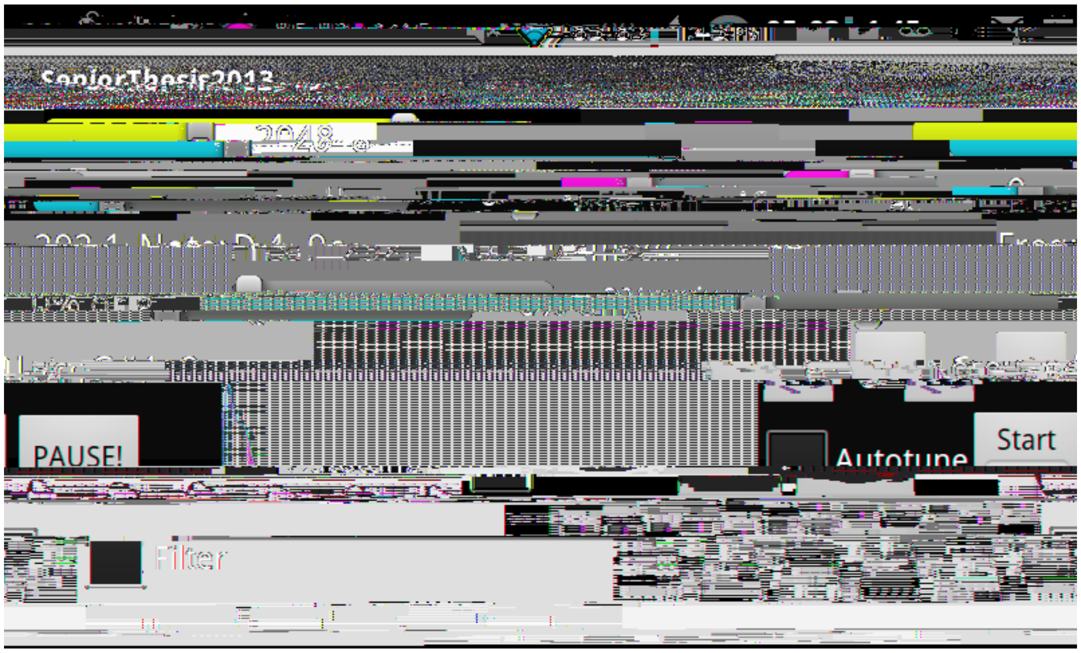


Figure 23

4) Pitch

Figure 24

5. Acknowledgements

I would like to thank Professor Alvarez for all the advice and encouragement given throughout the past few years pertaining to my thesis, my education, and my personal growth. This thesis would lack both breadth and depth if it were not for him. Professor Alvarez is not someone you would want to disappoint.

I would also like to thank Professor Signorile for answering all my random questions about Android design. The application flow improved significantly with his help, and real time processing became possible as a result.

6. References

[1] "Audio time-scale/pitch modification - Wikipedia, the free encyclopedia." Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/Audio_time-scale/pitch_modification (accessed May 16, 2013).

[2] - *Transactions of the IRE Professional Group on Audio, vol.2, no.1, pp. 7-12*, Jan 1954

[3] "1975 Eventide H910 Harmonizer-Mix Inducts Eventide H910 Harmonizer Into 2007 TECnology Hall of Fame.*" Mix Magazine | Pro Audio, Live Sound, Music Recording and Live Post for Audio Pros.* http://mixonline.com/TECnology-Hall-of-Fame/1975-eventide-harmonizer (accessed May 16, 2013).

[4] PRASHANT. "Acoustics." Padante. padante.com/acoustics/ (accessed May 16, 2013).

[5] Goldline Inc.. "Basics of Sound." Audio Engineering Society.

[illegible]

[12] Sawyer, Scott, Habib Estephan, and Daniel Wanninger. "Real-Time Pitch Shifting on an FPGA - II. Design Overview - Frequency-Domain Pitch Shifting." Villanova University. www56.homepage.villanova.edu/scott.sawyer/fpga/II_freq_domain.htm (accessed May 16, 2013).

[13] -Robust Pitch Detection using Auto-correlation Function with Enha *J. King Saud University, Vol. 22, Comp. & Info. Sci., pp. 13-28*,Riyadh (1431H./2010)

[14] *Journal of the Acoustical Society of America, vol. 111, pp. 1917–1930,* 2002.

[15] C.-Y. Lin and J.- -phase pitch marking method for TD-PSOLA *Proceedings of Interspeech/ICSLP pp. 1189–1192*, Jeju Island (Korea), October 2004.

[16] Chen, J.-H. and Kao, Y.- - *Computational Linguistics and Chinese Language Processing, Vol. 6, No. 5, pp. 1-12*, February 2001.

[17] Zölzer, U. e. a., *DAFX Digital Audio Effects,* John Wiley & Sons, eds., 2002