Spring 5-2017

Mathematical Methods for Voice Transformation

Nathan Lilla

MATHEMATICAL METHODS FOR VOICE TRANSFORMATION

A Thesis

Submitted to the School of Graduate Studies and Research

in Partial Fulfillment of the

Requirements for the Degree

Master of Science

Nathan Lilla

Indiana University of Pennsylvania

May 2017

Indiana University of Pennsylvania
School of Graduate Studies and Research
Department of Mathematics

We hereby approve the thesis of

Nathan Lilla

Candidate for the degree of Master of Science

_____ _____
John C. Chrispell, Ph.D.
Associate Professor of Mathematics, Advisor

_____ _____
Frederick A. Adkins, Ph.D.
Professor of Mathematics

_____ _____
Yu-Ju Kuo, Ph.D.
Associate Professor of Mathematics

ACCEPTED

_____ _____
Randy L. Martin, Ph.D.
Dean
School of Graduate Studies and Research

Title: Mathematical Methods for Voice Transformation

Author: Nathan Lilla

Thesis Chair: Dr. John C. Chrispell

Thesis Committee Members: Dr. Frederick A. Adkins
Dr. Yu-Ju Kuo

Each individual's voice has unique characteristics that can be distinguished from someone else. When two people vocalize the same pattern of speech, we can easily identify that there are two different speakers. To establish a working framework in converting a voice, a sample phrase of a desired voice will be compared with the same sample phrase of the source voice. Constructing a basis for each sample will allow for frequency analysis between the two recordings. From this comparison, a mapping will be identified from the given input to the the desired output. Using two individual voices: whenever the desired phrase is recorded, the data undergoes a transformation as key frequencies are mapped to the desired output and results in the phrase spoken with the desired sound. When determining the key characteristics of each audio signal fast Fourier transforms will be used. Once a mapping is identified, future input signals will be mapped to the desired output frequency using both a FFT and an IFFT (inverse fast Fourier transform) to produce the desired modulation of the given input.

ACKNOWLEDGMENTS

Dedicated to my mother, Mary.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

INTRODUCTION

Background

In popular television shows and movies, characters sometimes distort their voice as not to give away their identity. This is often a theme with superheroes. Superman was one of the few who could distort his voice to sound like other people by using precise movements to change his vocal chords. He used this ability once when he was disguised as Batman. Since we don't have the unique abilities that Superman does, is there a way we could change our own voice using technology.

There have already been developments in the manipulation and modification of sound with technology. One development is that guitar amps can create distortion of a guitar's sound. The distortion comes from manipulating the amplitude of sound waves. Some amps have effects that add new frequencies to change the desired output. There are guitar pedals available that can produce effects such as phaser, wah, reverb, delay, tremolo, etc. TV shows and movies often use a filter over someone's voice to make a villain sound more menacing. Auto tuners such as those developed by the company Antares can be used for someone singing to help them sing in the correct key. There are a few phone apps available for someone to change their voice after a recording. These apps generally have a few default options that a desired output can be. The question we wish to study is whether is method of converting audio can be applied to convert someone's vocal frequencies. This thesis explores a starting point in identifying a method to change a person's voice to someone else's or an instrument into another.

Previous Research

The article *Voice conversion through vector quantization* [1] set a lot of the ground work that many other researches have applied in the technique of voice conversion. The method discussed in the article begins by defining codebooks based on both source and target

speakers. A codebook is a library of specifications of a person's unique vocal attributes such as spectrum parameters, power values, and pitch frequencies [2, 7, 15, 17, 21]. Once both codebooks are constructed, a correspondence is established using dynamic time warping which matches up the words spoken in time between the two speakers. Then a third audio is converted using a mapping between the codebooks based on spectrum parameters and pitch frequencies.

Other research has taken the idea of generating codebooks and applying other methods such as Gaussian mixture models [5, 22, 23] as well as maximum likelihood transformations [14, 16, 19, 20, 25] which can change a parameter to produce a desired result. In the case of a Gaussian mixture model, the time-aligned acoustic features of a source speaker and target speaker are determined. These acoustic features are written as a probability distribution which is composed of a set of basis functions that follow normal distributions with parameters given as mean vectors and covariance matrices [6,10,11]. For this thesis, the goal is to instead examine a different set of basis functions. These basis functions will be ones that are created using a Fourier transform. They are helpful in analyzing the frequencies of an audio signal.

To briefly outline what will be studied, chapter 2 will discuss how the Fourier transform is performed and what applications it can be used for. It will be fundamental to understand what Fourier coefficients are as the 3rd chapter will describe methods on finding relationships between these coefficients from two different speakers. Chapters 4 and 5 will summarize results found and what should be considered for future development.

Chapter 2

FOURIER SPACE

Understanding the Fourier Transform

A discrete Fourier transform can be thought of as a function that decomposes an audio signal into a set of coefficients for a range of basis functions [18]. The basis functions summed together will give an approximation of the original function. We can consider this as the analogy of a bunch of intertwined cables. Picture that there is a large pile of Christmas lights all tangled together on the floor. We have no idea how many different sets there are or of which lengths. Now suppose there is this exciting new machine that you can throw the pile of lights in, it unties them all for you, and then spits them out one by one in order of size. Wouldn't that be best seller during Christmas! The machine essentially is a Fourier transform, where we input our audio signal, and out comes the frequency's coefficients to place on the basis functions that make up the signal. Figure 2.1 shows how three basis functions can be combined to make a new function.

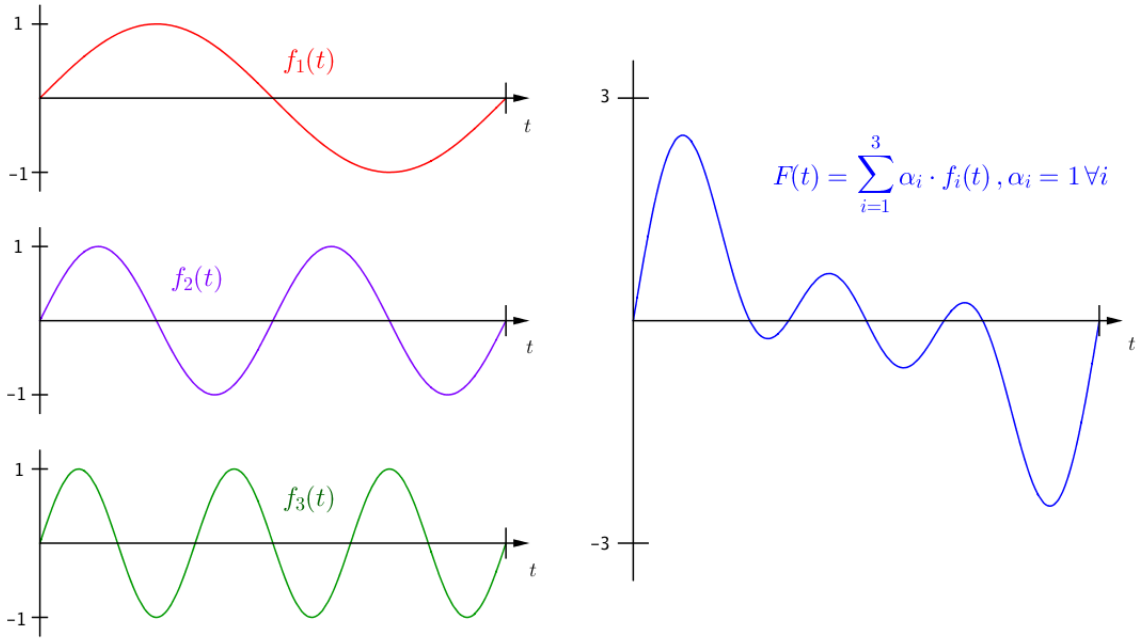

Figure 2.1: The single wave is a combination of three waves of different frequency.

This size of an audio signal is determined by how many points make up the wave. If the sample rate was 1000, then each second would be composed of 1000 points. Based on the size of the audio signal, the Fourier transform will produce that many coefficients for the basis functions which are composed of sine and cosine functions [8]. The discrete Fourier transform is constructed by the following summation:

$$X_k = \sum_{n=0}^{N-1} x_n \cdot e^{-i2\pi kn/N} \tag{2.1}$$

- X_k = Amount of frequency k (a complex number where the real part is the amplitude and the imaginary part is the phase)
- N = number of time samples (length of the audio signal)
- x_n = value of the time signal at time n.
- k = current frequency under consideration

Please note that we use the summation for the discrete case where as the integral is used for the general form of a Fourier transform [3]. Essentially what the transform is doing is finding the coefficients a_k and b_k for a sum of cosine series and sine series:

$$F(x) = a_0 + \sum_{k=1}^{\infty} [a_k \cos(kx) + b_k \sin(kx)] \tag{2.2}$$

In the case of the discrete Fourier transform, a_k will be a real number and b_k will be an imaginary number. Lets delve deeper into this equation to understand what is happening. First, recall Euler's formula:

$$e^{ix} = \cos(x) + i \sin(x)$$

We can now rewrite equation 2.1 as:

$$X_k = \sum_{n=0}^{N-1} x_n \left[\cos(2\pi kn/N) - i \sin(2\pi kn/N)\right]$$

Please note that the negative exponent in Figure 2.1 is addressed here as cosine is an even function and sine is an odd function. The sum will produce something in the form of:

$$X_k = a_k + b_k i$$

The values of X_k can be plotted in an imaginary coordinate plane as in Figure 2.2. From Figure 2.2, it can be seen how the amplitude is measured as the magnitude of the vector. The value of θ represents the phase shift of a cosine function.

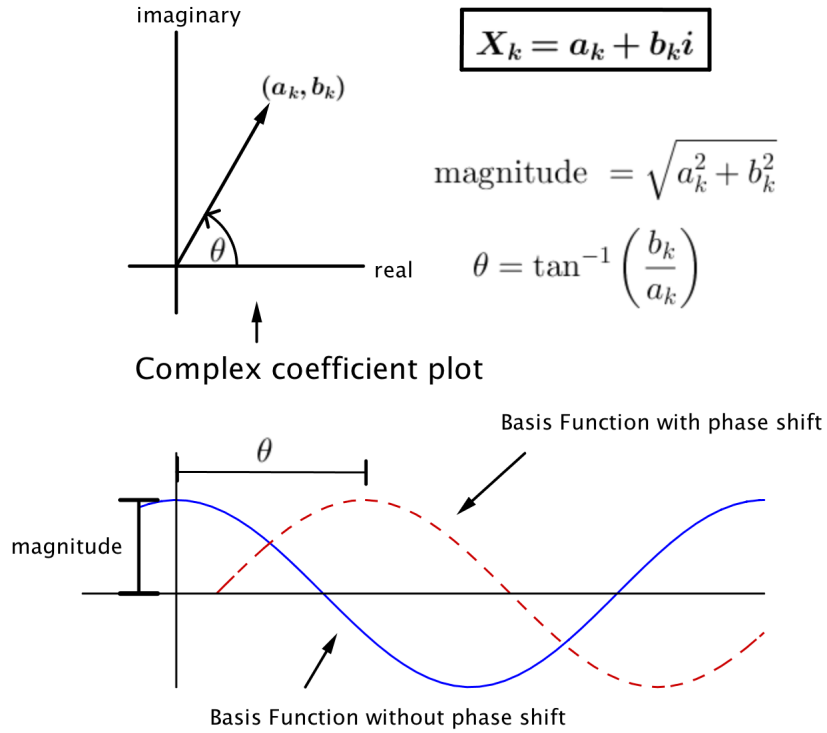

Figure 2.2: Magnitude and phase shift of the complex coefficient.

As an example, consider finding the Fourier transform of a simple wave. Given a sine wave with frequency 1 Hz and amplitude of 1 over 1 second. Let the sampling frequency be 8 Hz. To keep things simple, take the same number of discrete points as our sampling

frequency. Figure 2.3 shows the The points we have for x_n are:

x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7
0	$\frac{\sqrt{2}}{2}$	1	$\frac{\sqrt{2}}{2}$	0	$-\frac{\sqrt{2}}{2}$	-1	$-\frac{\sqrt{2}}{2}$

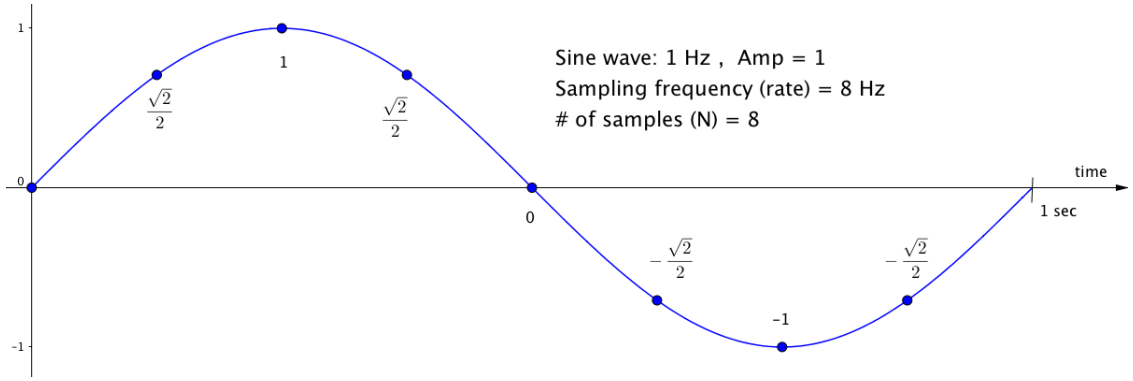

Figure 2.3: Sine wave of 8 discrete points with period 1 second.

To find the values for X_k. First consider $k = 0$. All we get is $\sum_{n=0}^{N-1} x_n$ because $\cos(0) = 1$ and $\sin(0) = 0$. The sum of the x_i values is zero so $X_1 = 0$. Now for frequency $k = 1$. We get the following sum:

$$\begin{aligned}
X_1 &= 0\,[\cos(0) - i\sin(0)] + \frac{\sqrt{2}}{2}\,[\cos(\pi/4) - i\sin(\pi/4)] + 1\,[\cos(\pi/2) - i\sin(\pi/2)] \\
&\quad + \frac{\sqrt{2}}{2}\,[\cos(3\pi/4) - i\sin(3\pi/4)] + 0\,[\cos(\pi) - i\sin(\pi)] - \frac{\sqrt{2}}{2}\,[\cos(5\pi/4) - i\sin(5\pi/4)] \\
&\quad - 1\,[\cos(3\pi/2) - i\sin(3\pi/2)] - \frac{\sqrt{2}}{2}\,[\cos(7\pi/4) - i\sin(7\pi/4)] \\
&= 0 + \left(\frac{1}{2} - \frac{1}{2}i\right) + (-i) + \left(-\frac{1}{2} - \frac{1}{2}i\right) + 0 + \left(\frac{1}{2} - \frac{1}{2}i\right) + (-i) + \left(-\frac{1}{2} - \frac{1}{2}i\right) \\
&= -4i
\end{aligned}$$

For the rest of the coefficients, we get:

$$
\begin{aligned}
X_2 &= 0\left[\cos(0) - i\sin(0)\right] + \frac{\sqrt{2}}{2}\left[\cos(\pi/2) - i\sin(\pi/2)\right] + 1\left[\cos(3\pi/2) - i\sin(3\pi/2)\right] \\
&\quad + \frac{\sqrt{2}}{2}\left[\cos(2\pi) - i\sin(2\pi)\right] + 0\left[\cos(5\pi/2) - i\sin(5\pi/2)\right] - \frac{\sqrt{2}}{2}\left[\cos(3\pi) - i\sin(3\pi)\right] \\
&\quad - 1\left[\cos(7\pi/2) - i\sin(7\pi/2)\right] - \frac{\sqrt{2}}{2}\left[\cos(4\pi) - i\sin(4\pi)\right] \\
&= 0 + \left(-\frac{\sqrt{2}}{2}i\right) + (-1) + \left(\frac{\sqrt{2}}{2}i\right) + 0 + \left(\frac{\sqrt{2}}{2}i\right) + (1) + \left(-\frac{\sqrt{2}}{2}i\right) \\
&= 0
\end{aligned}
$$

$$
\begin{aligned}
X_3 &= 0\left[\cos(0) - i\sin(0)\right] + \frac{\sqrt{2}}{2}\left[\cos(3\pi/4) - i\sin(3\pi/4)\right] + 1\left[\cos(3\pi/2) - i\sin(3\pi/2)\right] \\
&\quad + \frac{\sqrt{2}}{2}\left[\cos(9\pi/4) - i\sin(9\pi/4)\right] + 0\left[\cos(3\pi) - i\sin(3\pi)\right] - \frac{\sqrt{2}}{2}\left[\cos(15\pi/4) - i\sin(15\pi/4)\right] \\
&\quad - 1\left[\cos(9\pi/2) - i\sin(9\pi/2)\right] - \frac{\sqrt{2}}{2}\left[\cos(21\pi/4) - i\sin(21\pi/4)\right] \\
&= 0 + \left(-\frac{1}{2} - \frac{1}{2}i\right) + i + \left(\frac{1}{2} - \frac{1}{2}i\right) + 0 + \left(-\frac{1}{2} - \frac{1}{2}i\right) + i + \left(\frac{1}{2} - \frac{1}{2}i\right) \\
&= 0
\end{aligned}
$$

$$
\begin{aligned}
X_4 &= 0\left[\cos(0) - i\sin(0)\right] + \frac{\sqrt{2}}{2}\left[\cos(\pi) - i\sin(\pi)\right] + 1\left[\cos(2\pi) - i\sin(2\pi)\right] \\
&\quad + \frac{\sqrt{2}}{2}\left[\cos(3\pi) - i\sin(3\pi)\right] + 0\left[\cos(4\pi) - i\sin(4\pi)\right] - \frac{\sqrt{2}}{2}\left[\cos(5\pi) - i\sin(5\pi)\right] \\
&\quad - 1\left[\cos(6\pi) - i\sin(6\pi)\right] - \frac{\sqrt{2}}{2}\left[\cos(7\pi) - i\sin(7\pi)\right] \\
&= 0 + \left(-\frac{\sqrt{2}}{2}\right) + 1 + \left(-\frac{\sqrt{2}}{2}\right) + 0 + \left(\frac{\sqrt{2}}{2}\right) - 1 + \left(\frac{\sqrt{2}}{2}\right) \\
&= 0
\end{aligned}
$$

$$
\begin{aligned}
X_5 &= 0\left[\cos(0) - i\sin(0)\right] + \frac{\sqrt{2}}{2}\left[\cos(5\pi/4) - i\sin(5\pi/4)\right] + 1\left[\cos(5\pi/2) - i\sin(5\pi/2)\right] \\
&\quad + \frac{\sqrt{2}}{2}\left[\cos(15\pi/4) - i\sin(15\pi/4)\right] + 0\left[\cos(5\pi) - i\sin(5\pi)\right] - \frac{\sqrt{2}}{2}\left[\cos(25\pi/4) - i\sin(25\pi/4)\right] \\
&\quad - 1\left[\cos(15\pi/2) - i\sin(15\pi/2)\right] - \frac{\sqrt{2}}{2}\left[\cos(35\pi/4) - i\sin(35\pi/4)\right] \\
&= 0 + \left(-\frac{1}{2} + \frac{1}{2}i\right) + (-i) + \left(\frac{1}{2} + \frac{1}{2}i\right) + 0 + \left(-\frac{1}{2} + \frac{1}{2}i\right) + (-i)\left(\frac{1}{2} + \frac{1}{2}i\right) \\
&= 0
\end{aligned}
$$

$$
\begin{aligned}
X_6 &= 0\left[\cos(0) - i\sin(0)\right] + \frac{\sqrt{2}}{2}\left[\cos(3\pi/2) - i\sin(3\pi/2)\right] + 1\left[\cos(3\pi) - i\sin(3\pi)\right] \\
&\quad + \frac{\sqrt{2}}{2}\left[\cos(9\pi/2) - i\sin(9\pi/2)\right] + 0\left[\cos(6\pi) - i\sin(6\pi)\right] - \frac{\sqrt{2}}{2}\left[\cos(15\pi/2) - i\sin(15\pi/2)\right] \\
&\quad - 1\left[\cos(9\pi) - i\sin(9\pi)\right] - \frac{\sqrt{2}}{2}\left[\cos(21\pi/2) - i\sin(21\pi/2)\right] \\
&= 0 + \left(\frac{\sqrt{2}}{2}i\right) + (-1) + \left(-\frac{\sqrt{2}}{2}i\right) + 0 + \left(-\frac{\sqrt{2}}{2}i\right) + 1 + \left(\frac{\sqrt{2}}{2}i\right) \\
&= 0
\end{aligned}
$$

$$
\begin{aligned}
X_7 &= 0\left[\cos(0) - i\sin(0)\right] + \frac{\sqrt{2}}{2}\left[\cos(7\pi/4) - i\sin(7\pi/4)\right] + 1\left[\cos(7\pi/2) - i\sin(7\pi/2)\right] \\
&\quad + \frac{\sqrt{2}}{2}\left[\cos(21\pi/4) - i\sin(21\pi/4)\right] + 0\left[\cos(7\pi) - i\sin(7\pi)\right] - \frac{\sqrt{2}}{2}\left[\cos(35\pi/4) - i\sin(35\pi/4)\right] \\
&\quad - 1\left[\cos(21\pi/2) - i\sin(21\pi/2)\right] - \frac{\sqrt{2}}{2}\left[\cos(49\pi/4) - i\sin(49\pi/4)\right] \\
&= 0 + \left(\frac{1}{2} + \frac{1}{2}i\right) + i + \left(-\frac{1}{2} + \frac{1}{2}i\right) + 0 + \left(\frac{1}{2} + \frac{1}{2}i\right) + i + \left(-\frac{1}{2} + \frac{1}{2}i\right) \\
&= 4i
\end{aligned}
$$

All the coefficients are:

X_0	X_1	X_2	X_3	X_4	X_5	X_6	X_7
0	$-4i$	0	0	0	0	0	$4i$

Please note that X_1 and X_7 are complex conjugates. This has to do with symmetry that the Fourier transform produces. Note that the symmetry does not include the first point. The symmetry is from points 2 to n. Plotting these coefficients in Fourier space gives Figure 2.4. Since the coefficients are complex, the magnitude is used when plotting them.

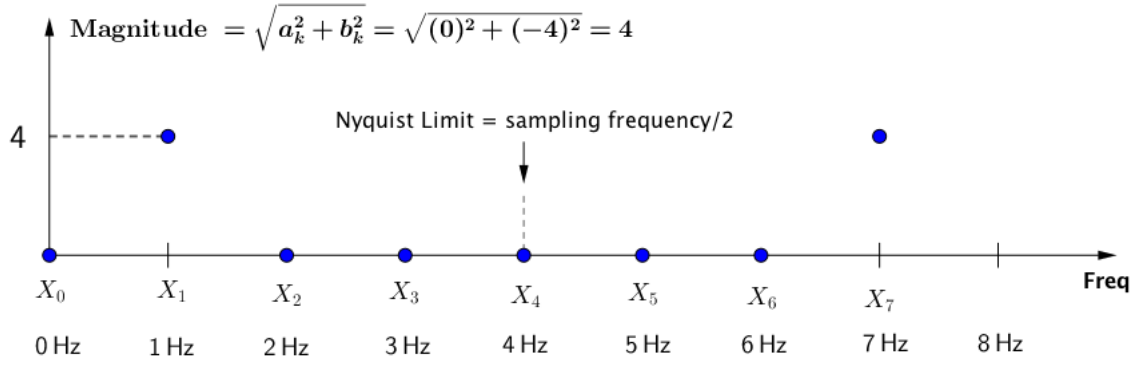

Figure 2.4: FFT of the 1 Hz sine wave.

Please notice the middle of the plot is the Nyquist Limit. This is also referred to as the Nyquist Frequency. Given a sampling rate of v, it is defined as:

$$f_{Nyquist} = \frac{1}{2}v\,.$$

The Nyquist limit is the highest frequency able to be extracted by a discrete Fourier transform given a sampling rate of v [13,18]. If the highest frequency to be extracted is f, a sampling rate of $2f$ is required. This is because at least two points are needed per cycle in order to determine a wavelength frequency. A good analogy of this would be to consider a camera's shutter speed. The length of the shutter being open is similar to the sampling rate.

You would need to locate the same point in two different locations to determine how fast it is moving (frequency). So when it comes to this double sided discrete FFT, the values to the right of the Nyquist Limit are not new frequencies but mirrored images of the first half. This is where the symmetry comes into play. In order to get the correct single sided discrete FFT, you would need to fold the coefficients over right to left and all the magnitudes would double in length. For our example, we had a magnitude of $-4i$. This then will be doubled to $-8i$ for the single sided discrete FFT. Figure 2.5 shows how this fold should appear. The new coefficients are now:

X_0	X_1	X_2	X_3
0	$-8i$	0	0

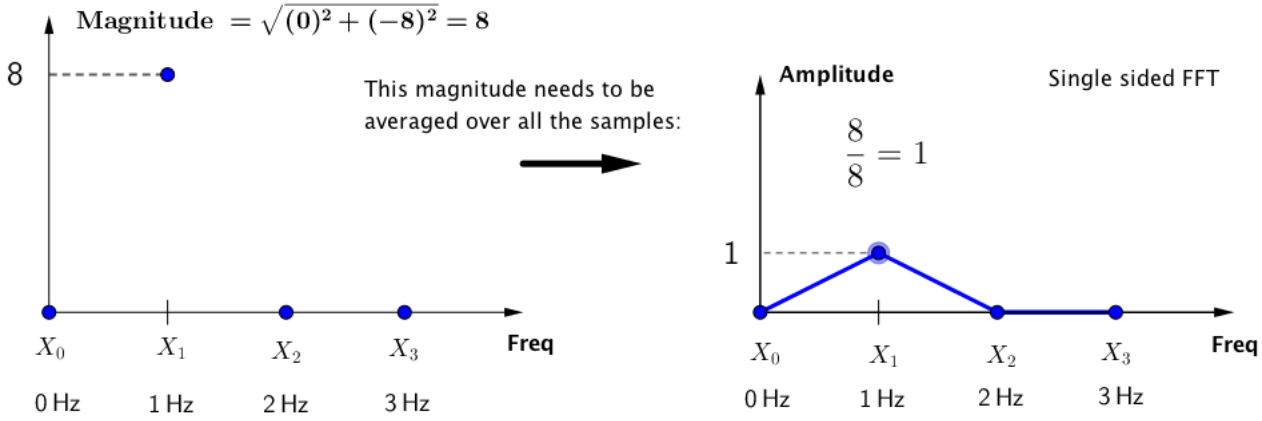

Figure 2.5: Single sided DFFT of magnitude and amplitude.

The left plot in Figure 2.5 shows a magnitude of 8 but that still needs to be averaged over the number of sample points in the original data which is 8. Even if X_2 and X_3 were not zero, they would still need to be incuded in the average. Since these coefficients were found over 8 different points, they need to be divided by 8 to identify their amplitude at that specific frequency. Dividing 8 by 8 gives us the amplitude of 1 which confirms what we had in Figure 2.3 with a frequency of 1 Hz. How can we determine that it is a sine graph beginning at the origin? The phase shift is performed on a cosine graph. Given a

coefficient of $-8i$, that means $a = 0$ and $b = -8$. We would be unable to use the formula $\theta = \tan^{-1}\left(\frac{b_k}{a_k}\right)$ because it is undefined for $a_k = 0$. If $\tan(\theta)$ is undefined, then $\theta = \frac{\pi}{2}$ or $\theta = \frac{3\pi}{2}$. Since the coefficient is $-8i$, that is a vector that travels directly down from the origin which means the angle must be $\frac{3\pi}{2}$. Figure 2.6 illustrates this. When a cosine graph is shifted to the right by $\frac{3\pi}{2}$ units, a sine graph is obtained as desired. Figure 2.7 illustrates this phase shift.

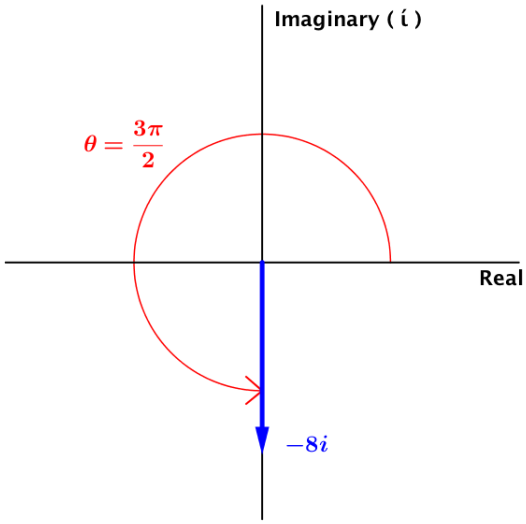

Figure 2.6: Imaginary plot of Fourier coefficient with phase shift shown.

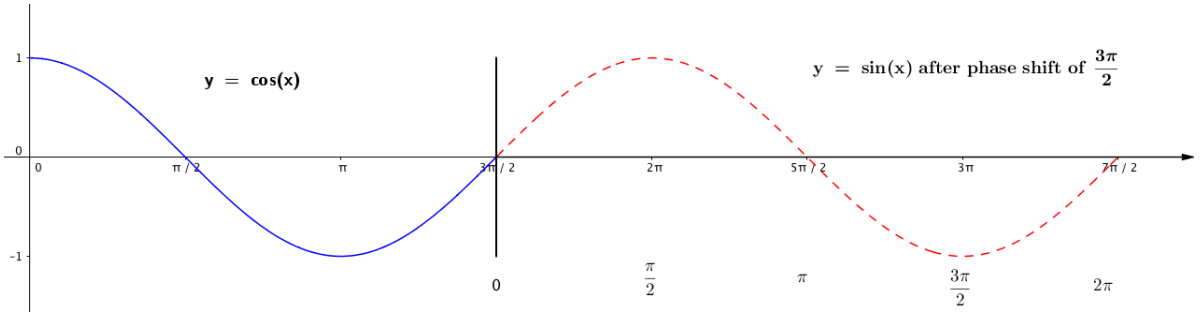

Figure 2.7: Applying a phase shift of the cosine wave to get a sine wave.

Now consider visualizing the DFFT of a more complicated function. Define it as $S(t)$.

$$S(t) = 0.7\sin(2\pi \cdot 80t) + 1.2\sin(2\pi \cdot 150t) + \sin(2\pi \cdot 200t)$$

Here, there are 3 different sine modes with non zero with amplitudes of 0.7, 1.2, 1 at frequencies 80, 150, and 200 respectively. Plotting $S(t)$ over 100 milliseconds in Figure 2.8 shows us that there multiple frequencies hidden in the data signal. Note that the wave is not fully resolved in the plot because it is graphed over only 100 points over the interval of 100 milliseconds. It would look sharper if more points were taken into account. Taking a Discrete Fast Fourier Transform (DFFT) will help identify which frequencies are present as well as their respective amplitudes. The DFFT can be viewed in Figure 2.9 and clearly shows that thesignal has amplitudes and frequencies that are consistent with the function $S(t)$. This demonstrates how powerful DFFT's can be when it comes to data processing for digital signals.

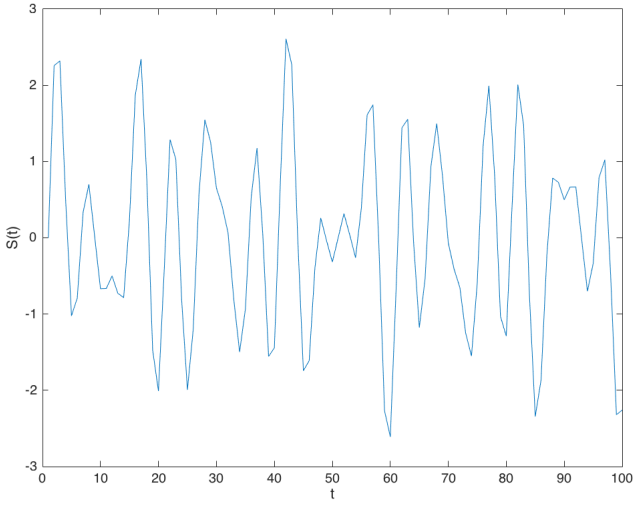

Figure 2.8: 100 points of $S(t)$ plotted with t in milliseconds.

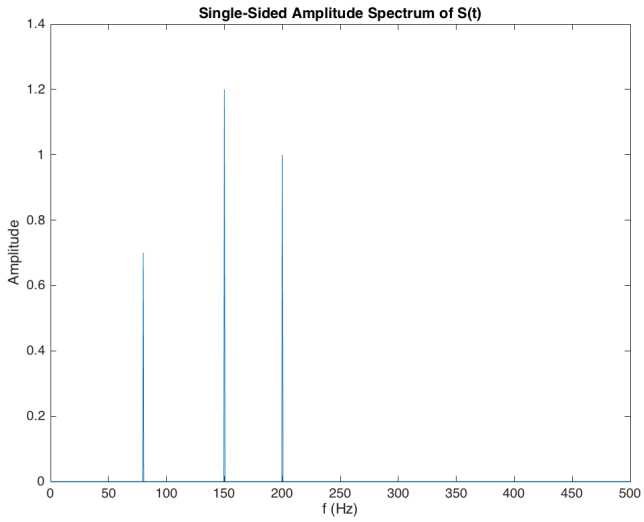

Figure 2.9: Plot of the frequency domain of $S(t)$.

Analyzing the FFT

As mentioned earlier, taking an FFT will produce complex numbers. From equation 2.2, we have values $a_0, a_k, b_k, \forall k \in \{1, n-1\}$ where n is the number of points. Using the python function `numpy.fft.fft()` will produce these coefficients. The first term a_0 will be a real number, the rest of the numbers will be complex with a_k being the real part and b_k being the imaginary part of each coefficient. It is also important to note that after the first term, the rest of the data is symmetric in the sense that the complex conjugates begin to appear for the second half of the points. If there are n points, the kth coefficient is the complex conjugate of the $(n-kth)$ coefficient. $a_k = a_{n-k}$ and $b_k = -b_{n-k}$ for $k \in \{1, \lfloor n/2 \rfloor\}$. For example, if an FFT was taken using 10 discrete points, the first coefficient would be real and coefficients 2 through 10 would be complex. The complex conjugates would be at points 2 and 10, 3 and 9, 4 and 8, 5 and 7, and since 6 is in the middle, it would not have a complex conjugate. Consider taking the FFT of integer values 1 through 10. Using the python package `numpy` , the FFT is easily obtained and the result was:

x_n	X_k
1	55
2	-5+15.388i
3	-5+6.882i
4	-5 + 3.633i
5	-5 + 1.625i
6	-5
7	-5-1.625i
8	-5-3.633i
9	-5-6.882i
10	-5-15.388i

Note that the first value is real, and the rest are symmetric with their complex conjugates. When viewing a plot of the FFT, the complex numbers become an obstacle. Thus, instead the absolute value of each complex number is used to get a real value, where the absolute value of a complex number is defined as:

$$|a + bi| = \sqrt{(a + bi)(a - bi)} = \sqrt{a^2 + b^2}\,. \tag{2.3}$$

Expression 2.3 is also called the magnitude of the complex number. After taking the absolute value of each term, it is necessary to shrink down the size back to the corresponding amplitudes. The Fourier coefficients have been scaled by the size of the data signal so each is divided by the length of the signal in order to bring the amplitudes back to the appropriate scale.

Since the FFT creates a symmetric set of points, only a single side of the spectrum is of interest. It would look as if the data points are folded in half from right to left and lay on top of each other. In doing this, the amplitude would double for each point, so only

the left side of the FFT is needed for the coefficients with double the magnitude. This is demonstrated in Figure 2.10.

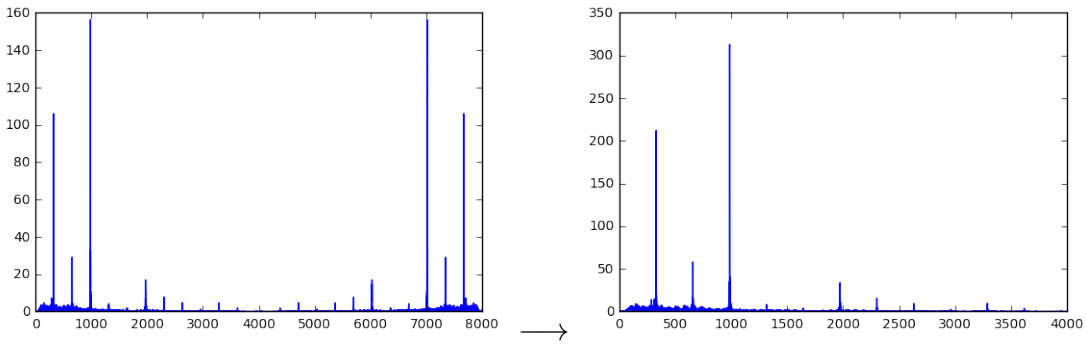

Figure 2.10: Folding an FFT over from right to left.

Figure 2.10 presents an audio sample of a single note from an acoustic guitar. It was recorded at a sample rate of 8000. A sample rate is the number of points per second. It is also referred to as the sample frequency. As seen in the first plot, the highest domain value is 8000. It is important to know that when recording audio with a sample rate of f, the highest frequency that can be identified is exactly half the sample rate ($f/2$) which is the Nyquist limit that was mentioned earlier. This is why we only wish to view the left hand side of the FFT which reaches a maximum frequency of 4000 Hz. The sample frequency constructed for $S(t)$ in Figure 2.8 was 1000 points per second which is why the maximum frequency for Figure 2.9 is 500 Hz.

Inverse Fourier Transform

The Fourier transform takes data from a time domain and moves it into a frequency domain. So the inverse Fourier transform does just the opposite. It moves the frequency spectrum back into a time domain. The inverse transform is found with the following summation:

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k \cdot e^{i2\pi kn/N} \tag{2.4}$$

The values of X_k, N, x_n, and k are defined in the equation (2.1). The main difference is that the values x_n are now found at time n. It is important to remember that when we analyzing an FFT, the magnitude of the coefficients are used which means some information is lost along the way such as the phase. Whenever an inverse transform is taken, it must be performed on the complex symmetric data. An inverse transform on the magnitude values will not produce the same results. We must also make sure that our data is symmetric complex conjugates and that the first value a_0 is a real number. This a_0 value is the vertical shift at the beginning of the signal. The data can be very sensitive when reverting back to a time domain so caution must be taken when manipulating the coefficients of the FFT. This issue will examined later in the following chapter when there is a discussion on constructing a mapping.

Peak Extraction

When analyzing an FFT, the target of interest is the peaks that appear. These peaks occur at locations where the corresponding frequency is dominant in the audio sample. Some audio samples are cleaner and easier to identify peaks than others. Consider an audio sample of an instrument playing a single note and compare this to an audio sample of someone speaking. The voice sample is obviously more complex. So the FFT of the voice will have much more going on with many peaks. There will be many more non-zero coefficients.

The reason these peaks are of interest is that they identify which frequencies are most prevalent in the audio sample. Recall the objective of this thesis is to determine a method of converting someone's vocal frequencies to a targets voice. In order to do this, it is desired to find key characteristics of a persons voice and these large peaks help identify which

frequencies are fundamental to their voice. After identifying these peaks, we then determine a way to map them to another person's key peaks. This discussion of mapping will appear in chapter 3.

Since voice samples can be rather complex and appear noisy, it can be helpful to smooth out the data signal to quiet down the extra noise in the signal [8]. This is done by taking a single point in the signal and replacing it with the average of itself and the points around itself. Using an r-point radius smoothing along the signal will help smooth the data . When using this smoothing technique, you will lose a bit of the amplitude but the key frequencies will remain essentially the same.

For example, if $r = 3$, then point x_i would be replaced with $\frac{x_{i-3}+x_{i-2}+x_{i-1}+x_i+x_{i+1}+x_{i+2}+x_{i+3}}{7}$ for $i \in \{3, n-3\}$. This isn't defined for the endpoints of the data since there would be a lack of points to include in the average. The endpoints would then be defined as:

$$\begin{aligned} x_0 &\rightarrow \text{no change} \\ x_1 &\rightarrow \text{3 point average} \\ x_2 &\rightarrow \text{5 point average} \\ x_{n-2} &\rightarrow \text{5 point average} \\ x_{n-1} &\rightarrow \text{3 point average} \\ x_n &\rightarrow \text{no change} \end{aligned}$$

To view an example, Figure 2.11 represents an audio sample being smoothed out with a radius of 3. Notice that the $y-$axis has changed a bit since the amplitudes would have shrunk down. Figure 2.12 zooms into the audio sample on the domain $[6000, 6500]$ which gives a better view of how the data appears more smooth. It can easily be seen that sharp jagged edges become smooth and less harsh. Figure 2.13 gives us a view at how the FFT space has less noisy. Note that the two audios played back do sound slightly different. After the smoothing, the audio is a little dimmer to hear because the amplitude has been cut down. The speech sounds the same, but it is more quiet.

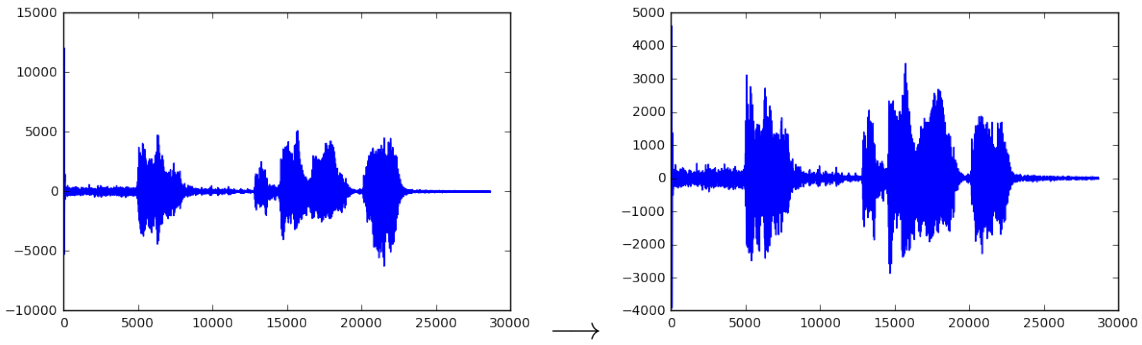

Figure 2.11: Audio sample before and after smoothing.

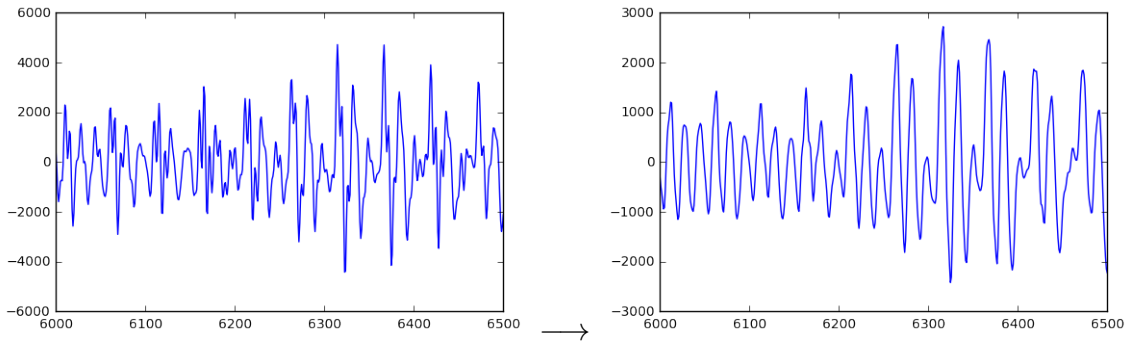

Figure 2.12: Zoomed in view of audio sample before and after smoothing.

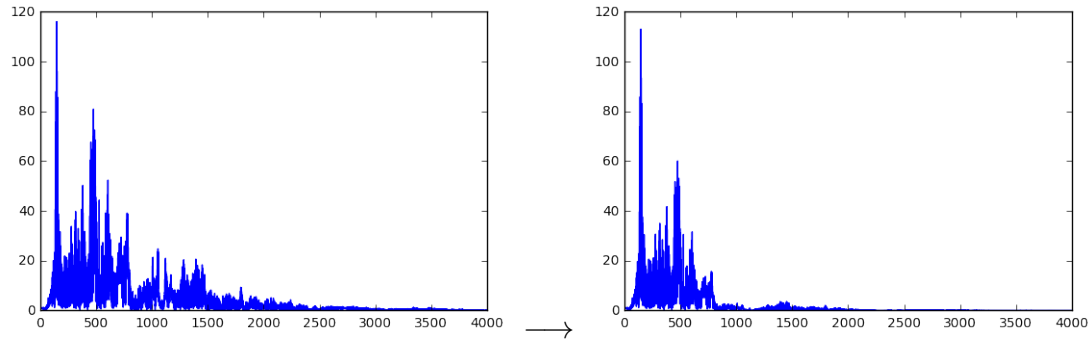

Figure 2.13: FFT of audio sample before and after smoothing.

Once there is a nice window to view the FFT, extracting the dominant peaks is the next step. Since different audio samples have different amplitudes, it can be helpful to normalize the data and declare a threshold with which to extract the peaks. Since the values are all

positive, divide all points by the largest point in the set which results in all amplitude values to be between 0 and 1. Picking the threshold to ignoring anything below 20%, Figure 2.14 shows the normalized FFT with selected peaks.

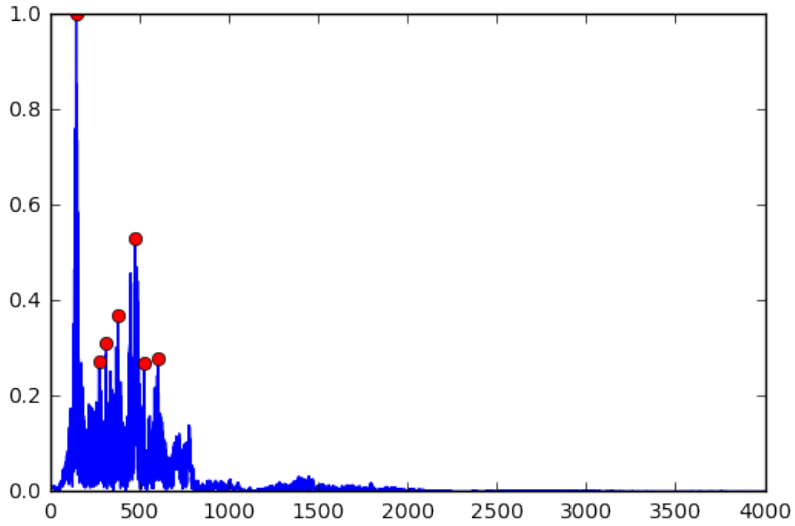

Figure 2.14: Normalized FFT with peaks selected.

Once the peaks have been extracted from both the source and target, the next step is to identify a way of mapping the two samples. Methods discussed in the following chapter consider scalar multiplication and shifts in coefficients.

Chapter 3

CONSTRUCTING A MAPPING

Comparing Frequencies

Essentially, the goal is to determine a way to transform a person's vocal frequencies to a target's vocal frequencies. To begin in identifying this sort of transformation, consider something simpler than speech. Compare two instruments since the sound is less complicated than speech. It is important that the two recordings have the same length and same pattern of music played. This could be a single note or a chord. Both recordings should display the same frequency based on the note or chord played and a few harmonics of that note as well.

A harmonic is a single oscillation whose frequency is an integral multiple of a fundamental frequency [13]. For example, middle A on a piano (also known as A4) has a frequency of 440 Hz. The next octave up (A5) has a frequency of 880 Hz which is exactly double the frequency of A4. This can also be thought of in terms of wavelength. Since the frequency is doubled, the wavelength will be halved. A4 has a wave length of 78.4 cm where as A5 has a wave length of 39.2 cm. When both notes are played in unison, they blend together and the result is harmony.

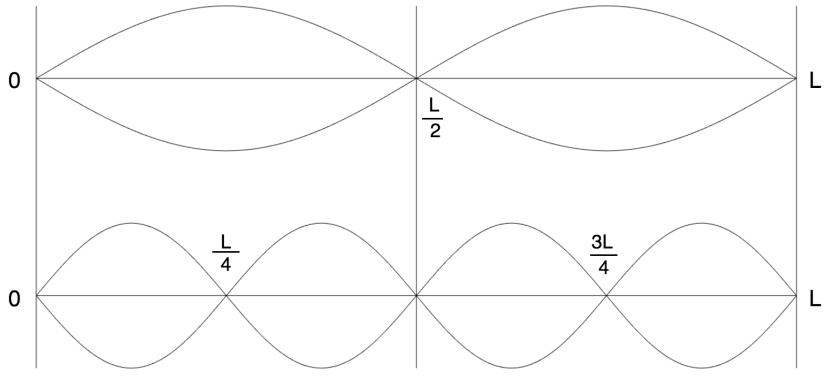

Figure 3.1: Fundamental frequency and its first harmonic.

Figure 3.1 is an example of a fundamental frequency shown with its first harmonic below it. The frequency for the harmonic is twice the first since it takes two full periods for

to reach the position that the fundamental frequency has traveled in one period. Completing a full period refers to a wave traveling the distance of its wavelength which is the distance between two peaks. It is also apparent that the wavelength of the harmonic is exactly half the wavelength of the fundamental note. When it comes to instruments such as a piano where a string is struck, the sound waves emitting from that string will bounce around and pass through things. If it comes into contact with something else that can vibrate at a harmonic frequency, it will in turn cause that thing to vibrate. If A4 is struck on a piano, the sound wave will actually cause any harmonics to vibrate as well. This effect creates the sound known as resonance. It is important to understand because when you record an audio sample of an instrument, you will pick up harmonic frequencies based on the fundamental note being played which doesn't necessarily represent the timbre of that particular instrument. Timbre refers to the unique sound that a certain instrument is associated with [13].

Considering Regression

When first confronted with the task of constructing a mapping between the two audio samples, regression was the first method I considered. The initial plan was to extract somewhere around 4 and 6 peaks from both audio samples, plot them on a Cartesian plane, and identify an appropriate model or function to represent these points. The method discussed here considers identifying a polynomial to represent the function. Given n points, a polynomial or order $n-1$ would be found. For example, a line (order 1) would need two points. A quadratic (order 2) would need three points.

Consider an example of comparing frequencies of a banjo and guitar playing the same C chord. Figure 3.2 shows the normalized FFT of each instrument with 4 peaks selected. Please keep in mind that this normalized data is based of the magnitude of the single side FFT.

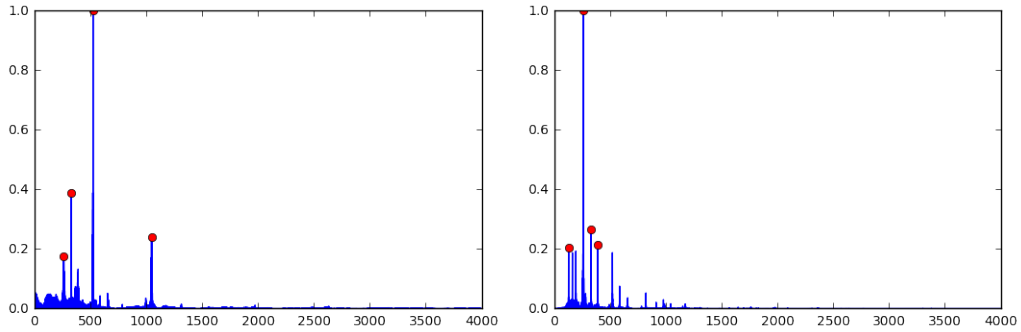

Figure 3.2: FFT of a banjo (left) and guitar (right) playing the same C chord.

The first plot in Figure 3.3 shows the relationship of the coefficients between the two audio samples. Something that is important to mention here is that any regression calculated right now will not help establish a mapping outside the range of our points. So anything below our lowest point and above our highest may not fit nicely. The lowest point refers to the left most peak in both the FFTs where as the highest points refers to the right most peak in the FFTs. Therefore, including a point below at (0,0) and a point above will help extend the range of the mapping. For the upper point, something as simple as multiplying the last point by a value of about 1.2 will be sufficient. This way, both of the coefficients for the last point move upwards by the same scale. The second plot in Figure 3.3 introduces these two new points.

x	0.0	86.03	117.10	188.36	485.98	583.17
y	0.0	186.76	195.54	240.94	906.29	1087.55

Since there are 6 points now, the data can be fit to a polynomial or order 5. The function will look like $y = a_0 + a_1x + a_2x^2 + a_3x^3 + a_4x^4 + a_5x^5$ where x is the coefficients of basis functions for the banjo's FFT. The value y represents how to move coefficients from banjo to guitar. The values of a_i are shown in equation 3.1. The normal equations $\mathbf{Xa} = \mathbf{y}$ can be solved by computing $\mathbf{a} = X^{-1}\mathbf{y}$.

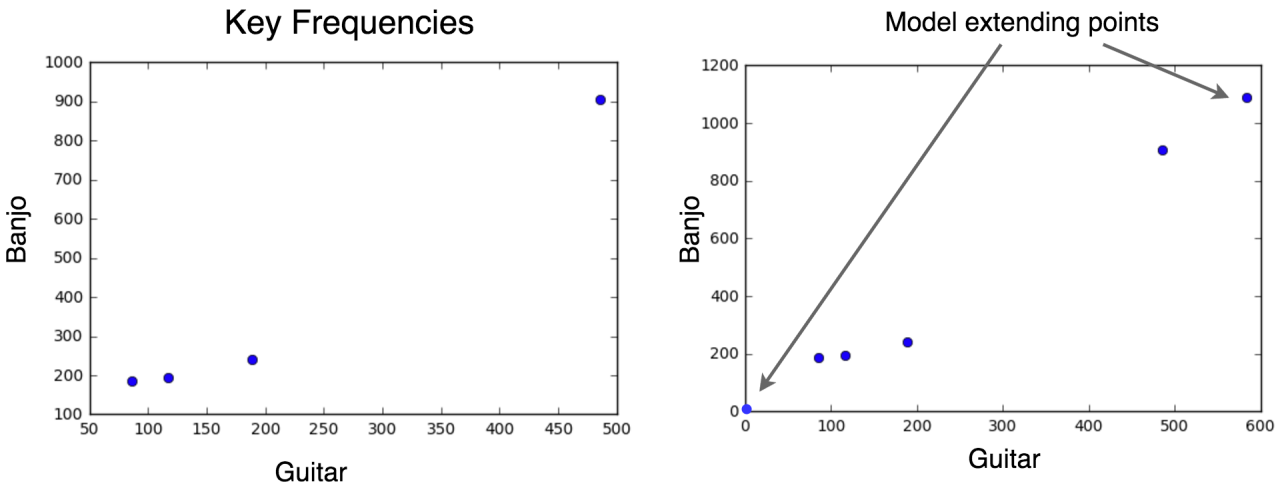

Figure 3.3: Extending the range of frequencies by inserting endpoints out farther.

$$\begin{bmatrix} 1 & x_1 & x_1^2 & x_1^3 & x_1^4 & x_1^5 \\ 1 & x_2 & x_2^2 & x_2^3 & x_2^4 & x_2^5 \\ 1 & x_3 & x_3^2 & x_3^3 & x_3^4 & x_3^5 \\ 1 & x_4 & x_4^2 & x_4^3 & x_4^4 & x_4^5 \\ 1 & x_5 & x_5^2 & x_5^3 & x_5^4 & x_5^5 \\ 1 & x_6 & x_6^2 & x_6^3 & x_6^4 & x_6^5 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \end{bmatrix} \tag{3.1}$$

Using the points from the second plot in Figure 3.3, coefficients a_i can be found for a polynomial of order 5. Figure 3.4 shows the best fit order 5 polynomial through each of the 6 points. The coefficients for a_i are as follows:

a_0	0.0
a_1	5.29214863
a_2	−5.53250637e-02
a_3	2.60692055e-04
a_4	−4.84067923e-07
a_5	3.12842597e-10

$$y = 5.29214x - 5.53250\text{e-}02x^2 + 2.60692\text{e-}04x^3 - 4.84068\text{e-}07x^4 + 3.12843\text{e-}10x^5 \quad (3.2)$$

Equation (3.2) represents this order 5 polynomial and you will notice that it is mostly linear. The linear coefficient has a significant effect compared to the rest of the coefficients are quite small. Figure 3.4 displays the plot of this polynomial through the selected points.

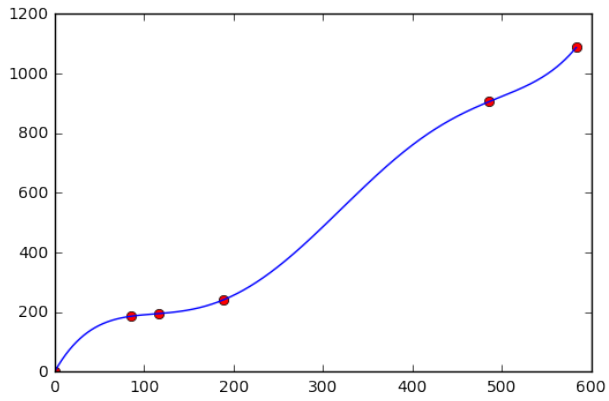

Figure 3.4: Order 5 polynomial fit through the 6 points.

The objective now, is to record a third audio sample, take the FFT and perform the regression over the coefficients on those frequencies. A third audio sample was taken from playing a banjo chord. Figure 3.5 shows the FFT of this audio sample on the left. The

right side plot shows the how the coefficients have changed after being transformed by the polynomial from equation 3.2.

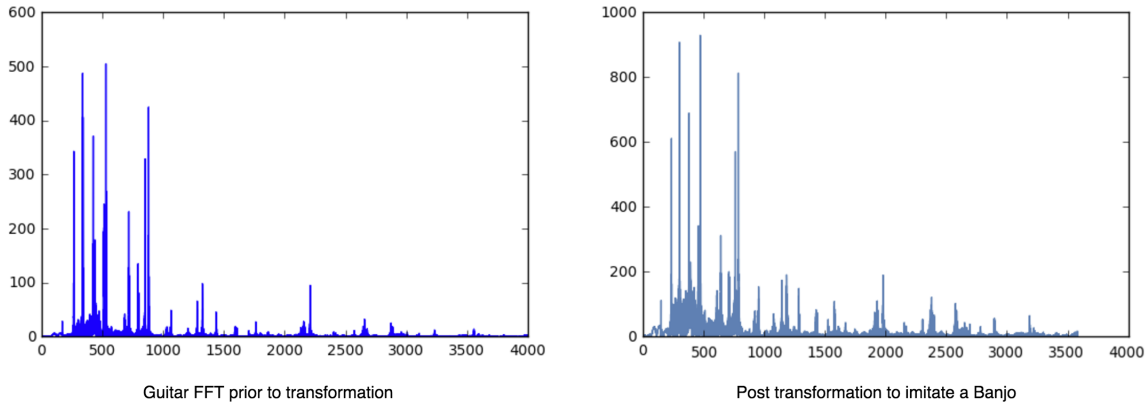

Figure 3.5: Third audio FFT before and after transformation of coefficients.

Once the regression has been applied to map the coefficients, the next step in the process is to take an inverse Fourier transform (IFFT) to move the data back to the time domain. Several problems now arise. When analyzing the FFTs, they were viewed with magnitudes. While this is easier visualize the data, information will be lost along the way. The complex numbers in the FFT contain information such as where the wave begins. The complex number represents the phase shift of the wave so the initial position is important. Setting the phase for each frequency to zero will have repercussions when performing an IFFT on the data. In fact, when taking the IFFT on only real coefficients, the result became a complex signal which didn't produce any intelligible sound when played digitally through speakers. This was an important insight as it is now understood that any transformation of the FFT must be done on the complex numbers and not on the magnitudes. This unsuccessful result isn't quite as heartbreaking if you compare it to Thomas Edison learning how not to make a light bulb.

Note that fitting a polynomial to the model is not the only option. In fact, with a large number of key frequencies, the order the polynomial grows and becomes a very sensitive

and nonideal model. I considered using splines to fit the data, however, after the failed results with the polynomial fitting, regression was simply not the correct direction. I had also considered two seperate mappings; the real to real coefficients as well as the complex to complex coefficients. This, however, will not work either as the real and imaginary parts should not be seperated as they held determine the phase shift of that frequency which is highly important when taking the inverse transform. This is discussed further in more detail in chapter 4.

Ratios of Magnitudes

As previously mentioned at the end of the last section, some sort of transformation will need to be applied on the complex coefficients and not on the magnitudes. One method to consider would be finding a ratio of magnitudes between two audio samples for each frequency coefficient and using it as a scalar multiplier to transform a future audio recording. Let us denote our ratio to be $r_i = \dfrac{d_i}{c_i} = \dfrac{a+bi}{w+vi}$ where c_i is the ith Fourier coefficient from the FFT of the source voice that is desired to be changed and d_i is the Fourier coefficient from the FFT of the target voice desired to be replicated. It is important that each sample have the same number of points.

When recording a new audio sample (third sample), the Fourier coefficients will be multiplied by the ratio $|r|$. This way the coefficients will stretch or shrink according to the ratio provided when comparing the two audio samples. Define the coefficients for the third audio sample to be K_i. Therefore, the new transformed coefficients will be written as $\widetilde{K}_i = |r_i| \cdot K_i$ or consider it in vector form:

$$\begin{bmatrix} \widetilde{K}_1 \\ \widetilde{K}_2 \\ \vdots \\ \widetilde{K}_n \end{bmatrix} = \begin{bmatrix} |r_1| \\ |r_2| \\ \vdots \\ |r_n| \end{bmatrix} \cdot \begin{bmatrix} K_1 \\ K_2 \\ \vdots \\ K_n \end{bmatrix}$$

Since $|r_i|$ is a real number and K_i is a complex number, the result $\widetilde{K}_i$ will be a complex number and there won't be any loss of information in the transformation as was seen in the previous section. One thing that may be helpful to note is that the magnitude for the ratio r_i can be calculated either by $|r_i| = \dfrac{|d_i|}{|c_i|}$ or as $|r_i| = \left|\dfrac{d_i}{c_i}\right|$. This can be shown below:

$$\begin{aligned} \left|\frac{a+bi}{w+vi}\right| &= \left|\frac{a+bi}{w+vi} \cdot \frac{w-vi}{w-vi}\right| \\ &= \left|\frac{(aw+bv)+(bw-av)i}{w^2+v^2}\right| \\ &= \sqrt{\left(\frac{aw+bv}{w^2+v^2}\right)^2 + \left(\frac{bw-av}{w^2+v^2}\right)^2} \qquad (3.3) \\ &= \sqrt{\frac{a^2w^2+2abwv+b^2v^2}{(w^2+v^2)^2} + \frac{b^2w^2-2abwv+a^2v^2}{(w^2+v^2)^2}} \\ &= \sqrt{\frac{a^2(w^2+v^2)+b^2(w^2+v^2)}{(w^2+v^2)^2}} \\ &= \sqrt{\frac{a^2+b^2}{w^2+v^2}} \\ &= \frac{\sqrt{a^2+b^2}}{\sqrt{w^2+v^2}} \\ &= \frac{|a+bi|}{|w+vi|}. \end{aligned}$$

Step (3.3) uses the absolute value of complex numbers defined in equation (2.3). Since both yield the same result, for our code, define the ratio as $|r_i| = \dfrac{|d_i|}{|c_i|}$ since it is a more efficient method due to fewer calculations it must perform. This is easily verified by calculating the

difference in time it takes between the two algorithms. Since our magnitudes are positive, the ratio will be positive and centered around zero. Figure 3.6 represents how a plot of the ratio should be at a given point n. The plot shouldn't look entirely smooth since there are a finite number of points and the ratio is unique from point to point. It may be greater than one at one point, and less than one at the next point. While it is not necessary to plot the ratio, it will give a good idea of how much the ratio changes over the spectrum.

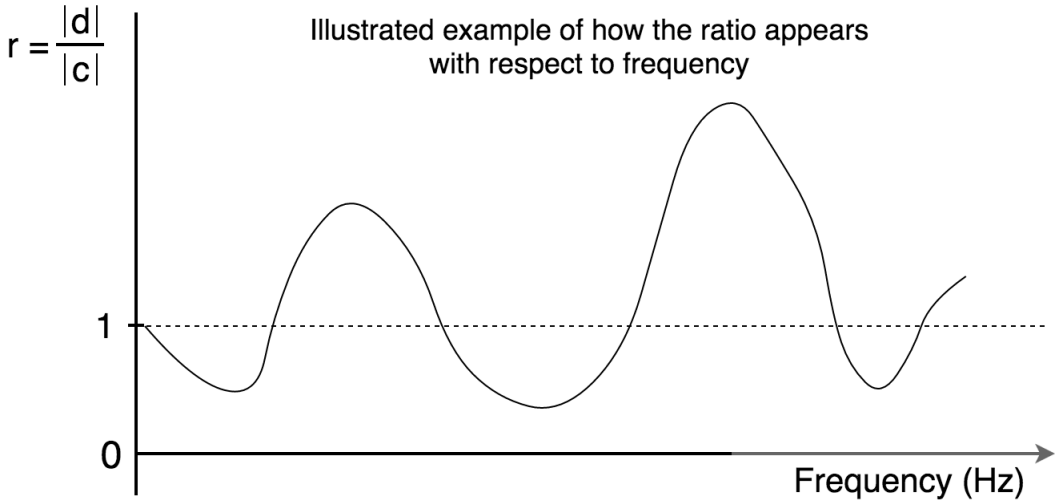

Figure 3.6: A plot of the ratio of coefficients between audio samples at each frequency.

Consider viewing the frequency spectrum with two voice samples. Figure 3.7 has the double sided spectrum of the FFT of two audio samples produced by distinct voices repeating the same sample of speech. The smoothing technique discussed in the previous chapter has been implemented for this example.

Figure 3.8 shows ratio of the coefficients shown in Figure 3.7. The second audio sample is the numerator of the ratio as that is the target (desired) voice. Something to notice here is that the plot doesn't much look like that in Figure 3.6. Since the ratio is centered at 1, the values larger than 1 will skew the visual representation upward. In fact, one ratio can be seen to be larger than 120. This large value will stretch the y-axis considerably. If

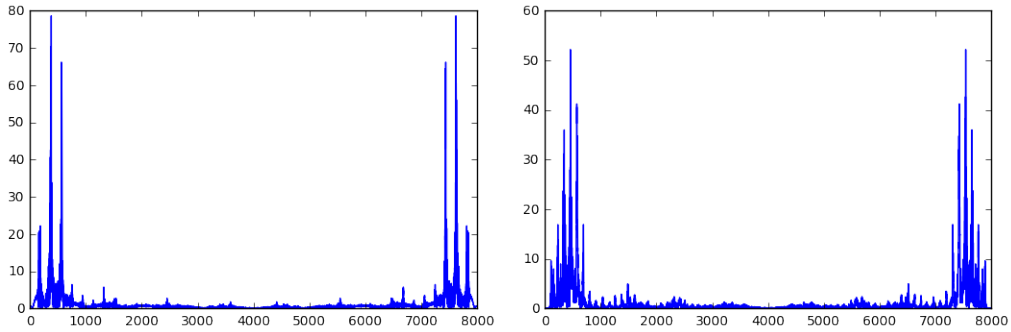

Figure 3.7: FFT's of two audio samples with distinct voices and same sample of speech.

on the other hand the ratio was 1/120, the value would be very close to zero but wouldn't shrink the vertical window at all. Another thing that should catch the reader's eye is that the original FFT of the two audio samples does not show much happening in the range from 2000 to 6000, but the ratio shows that there are large spikes in that range. This is due to some values from the first FFT being extremely close to zero in that range. When the denominator of the ratio is extremely close to zero, then the ratio can get rather large. These spikes shouldn't be too much to worry about because they will be multiplied by small numbers close to zero in the third audio sample. That means the ratio will have only a small affect on those frequencies.

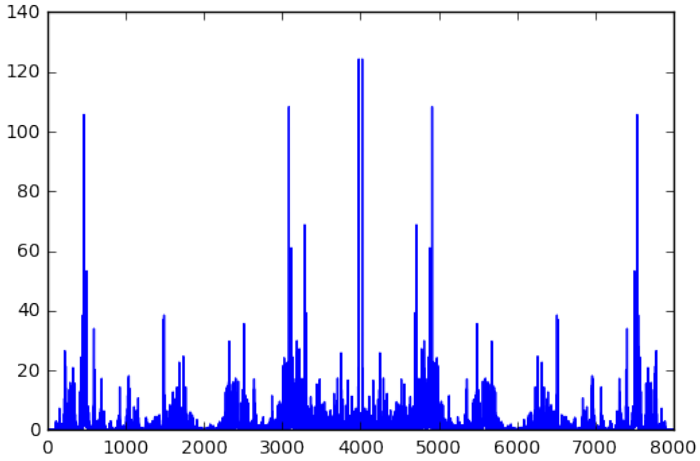

Figure 3.8: Ratio of the two FFT coefficients

Figure 3.8 does visually show the magnitudes of the FFT, but the data in complex form is still present so the information such as phase is not lost. An inverse fast Fourier transform can now be implemented and play back the audio. The result from this example did not produce an intelligible audio. Perhaps, the large ratios are too much for the transformation to handle. To combat these large ratios, consider reducing the power of the ratio by compressing the values towards the center at 1. The ratio can't simply be reduced by multiplying it by a factor of one half because it will not be centered at 1 anymore. Any values below 1 would get smaller and anything between 1 and 2 would drop below 1. To keep the ratio centered at 1, the power of the ratio by can be reduced transforming r using the following formula:

$$\widetilde{r} = \frac{(h-1)r + 1}{h}, \quad \text{for } h > 1 \tag{3.4}$$

Here the current ratio is r and the adjusted ratio is $\widetilde{r}$. This will only work for values of $h > 1$ and is only necessary for values of h close to 1. The closer h is to 1, the larger the ratio will be reduced. Figure 3.9 shows the ratio being reduced when we have a value of $h = 1.25$. The shape should appear almost identically except that the range has shrunk down. It can be seen that the y max has shrunk from 140 to 14. The values below 1 have moved upwards closer to 1.

Before shrinking the ratio, the audible result was unintelligible. After applying the reduction, you could hear the audio a little clearer but the result was not the changed voice we wished to hear. While disappointing that it does not work, it did show us that a simple ratio multiplication would not be the way. Another method will be necessary.

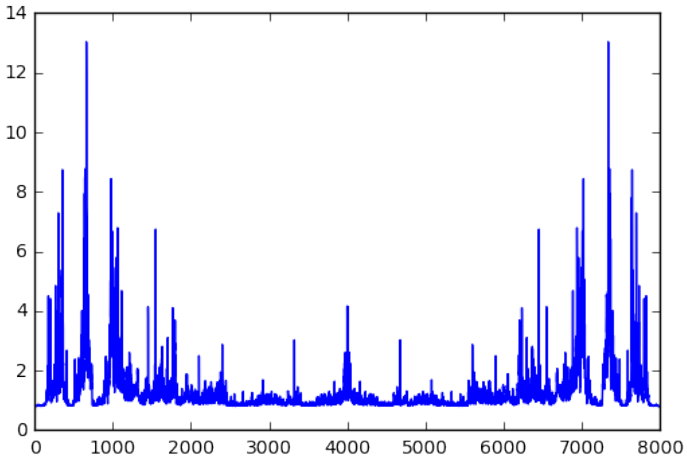

Figure 3.9: Ratio of the two FFT coefficients.

Effects of Linear Shifts

Instead of moving peaks up and down in the FFT (compress/stretch the magnitude), let us consider shifting them left and right in the domain [24]. To begin this method, it is important make sure shifting left and right will produce audible results so it can be determined if this is the right path. Consider starting with a simple linear shift where each frequency moves the same direction the same amount.

Consider the same sample rate of 8000Hz used before. To say moving the frequencies left, it is referring to the single sided spectrum (0 to 4000 Hz). In terms regular FFT (two sided spectrum), the frequencies will be moving outward from the center. Figure 3.10 shows an audio sample being shifted left in the frequency domain. This means all frequencies in the audio sample were shifted down by the same amount. In this example from the figure, the frequencies were shifted down by 300 Hz. That means a vocal frequency of 1000 Hz becomes 700 Hz. The playback of this audio successfully sounded like a voice transformation. The source voice became much deeper and sounded like what can be called the "Kidnapper voice."

Since shifting the frequencies left was successful, now consider a shift to the right. That means the double sided FFT would have frequencies moved in towards the center at 4000

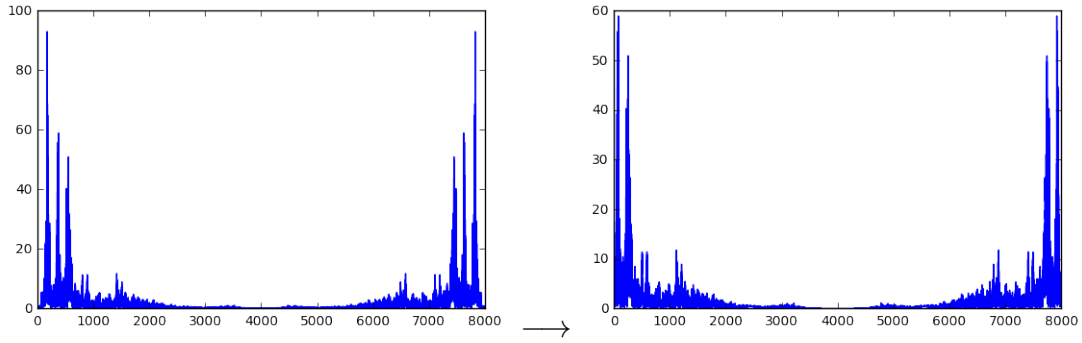

Figure 3.10: Linear shift left to lower frequencies.

Hz. Figure 3.11 shows an audio sample being shifted right in the frequency domain. In this example, frequencies were also shifted by 300 Hz, so a vocal frequency of 1000 Hz became 1300 Hz. The playback of audio was also a success as the source voice became more nasally and higher as what can be called the "Chipmunk Voice."

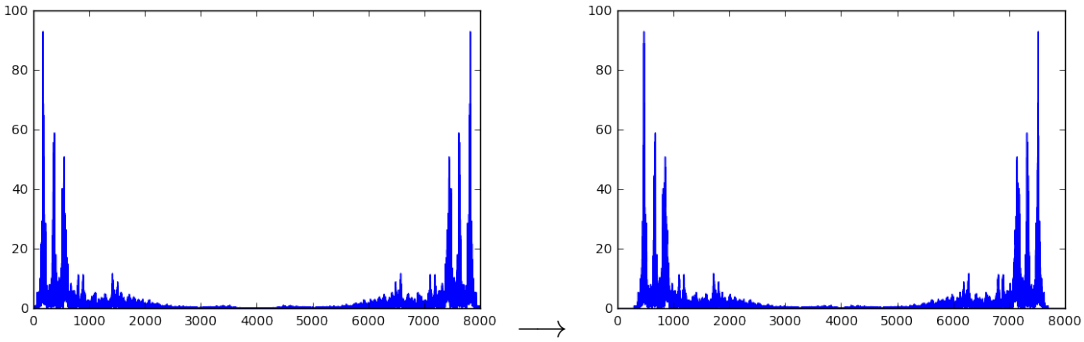

Figure 3.11: Linear shift right to higher frequencies.

Something that should be understood is that when performing a linear shift, a loss of information will occur. For the case where we shifted left 300 Hz, all frequencies in the range of 0 Hz to 299 Hz were removed and replaced with the frequencies from 300 Hz to 599 Hz. As well as all the frequencies from 3700 Hz to 3999 Hz were replaced with zeros.

Range Shift

Now that the linear shift is known to be successful, consider only shifting a portion of the frequencies. To begin, only consider shifting a range of frequencies. Each person has a unique range of frequencies that their speech follows when they talk in their normal speech pattern. So consider the range frequencies of a source voice: (A, B) and a target voice's range of frequencies: (a, b). The desire is to shift frequencies: $A \to a$ and $B \to b$. This shift will either get wider or narrower depending on the voice samples provided. Everything else outside the range will be unaffected and any frequencies that don't overlap will be ignored and set to be zero. Figure 3.12 represents a simple visual of how this appears in FFT space. In the case of this diagram, it goes from a small range to a larger one. That means coefficients should be either duplicated or left as empty spaces with value zero.

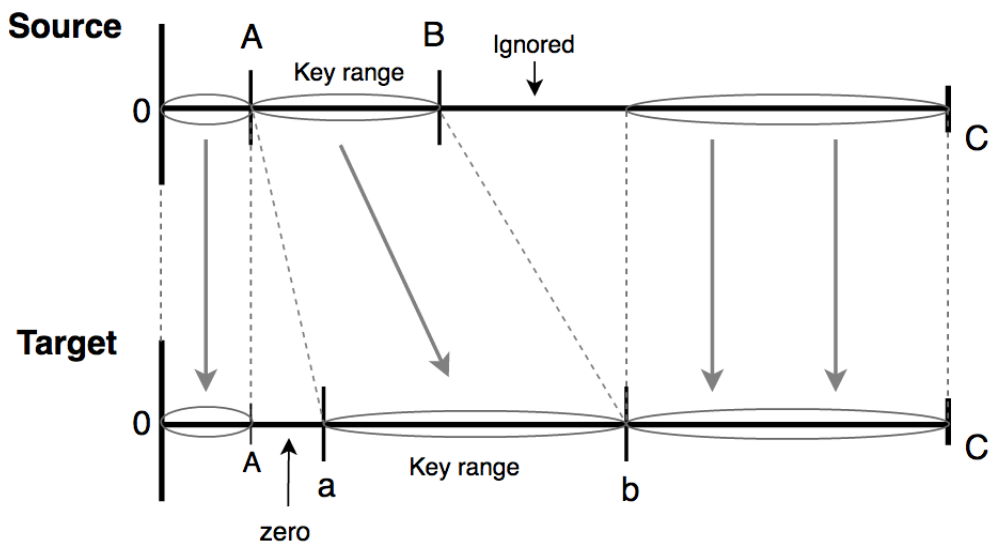

Figure 3.12: Shifting source's key range to target's key range of frequencies.

For an example, take a look at audio samples of a guitar and a mandolin that play the same chord. Let the guitar be the source and the mandolin be the target. In this case it should be expected that the range shift move up in frequencies. The mandolin should have a higher range since the instrument has a brighter sound than the guitar. After identifying

peaks in the FFT of both audio samples, a frequency range for the guitar of (165,1177) and a range for the mandolin of (291,1476) have been identified. Figure 3.13 shows the single sided FFT of these two instruments with the ranges displayed.

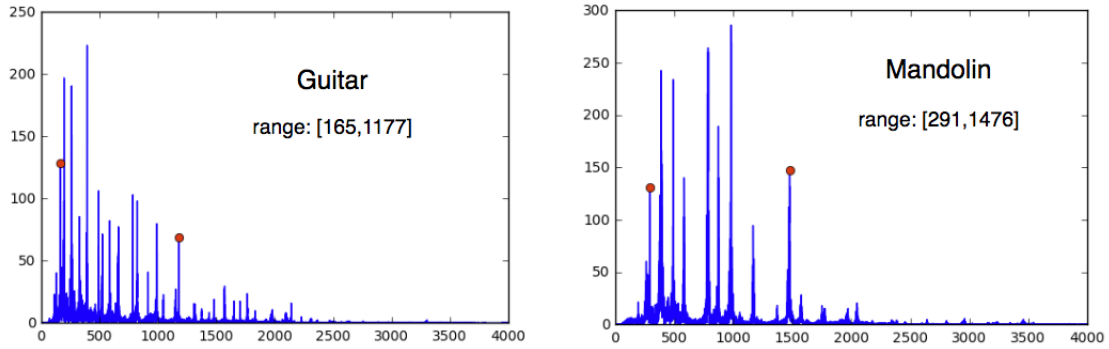

Figure 3.13: Key frequency range for guitar and mandolin.

Figure 3.14 shows the guitar's key range of frequencies shifted in the direction of the mandolin's key range of frequencies. It can be seen that there is an empty space from values 165 to 191 since there are no coefficients to fill in there similarly as to what is shown in Figure 3.12. Any duplicate coefficients were replaced with zeros in this example.

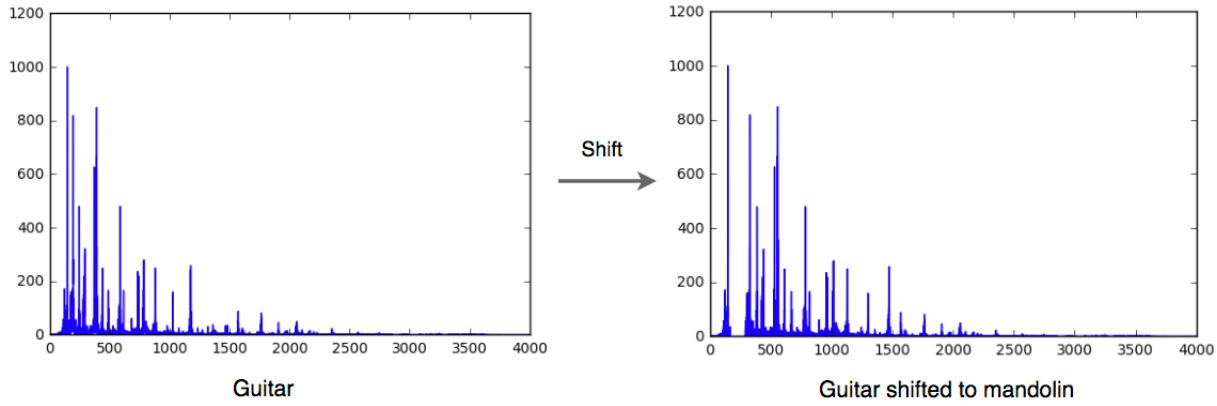

Figure 3.14: Pre and post shift of guitar key frequency range.

When comparing key ranges, it is always likely that the two will have different length. That means either some coefficients are lost or duplicated in the mapping. Suppose the

source has a key range from 100 to 200 Hz and the target has a key range from 150 to 300 Hz. The ratio of the length in each interval would be:

$$\frac{200-100}{300-150} = \frac{2}{3} \approx 0.667$$

The fraction 2/3 shows that 2 coefficients in the source must map to 3 coefficients for the target. That means 1 value must be a duplicate. Or 66.7% of the target points get mapped directly from the source and the other 33.3% of them duplicates. Or if the range's were reversed and the fraction was 3/2, that means for every three points, two would get mapped directly and one would be dropped away from the source. Figure 3.15 illustrates this compression and stretch. Please note, that if desired, one could simply replace any duplicate with a coefficient of zero if it is necessary for the voice conversion. An alternative approach is that you would reduce the amplitude whenever there are duplicates or increase the amplitude for any time a coefficient is dropped away.

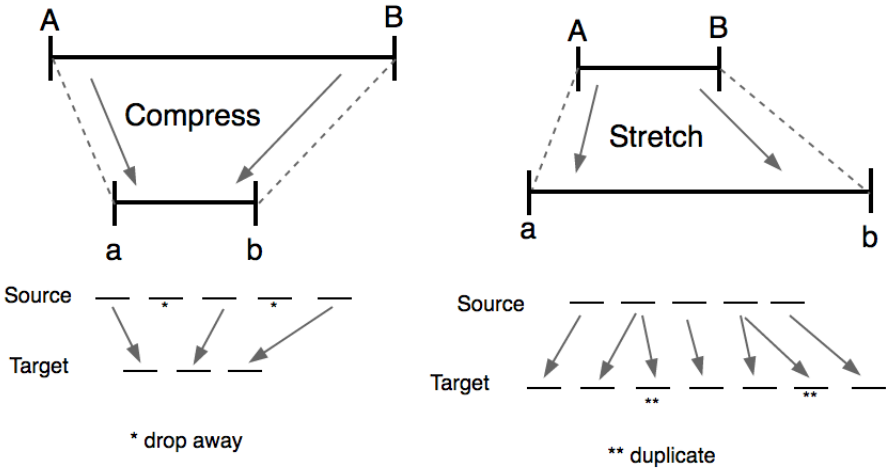

Figure 3.15: Illustration of compressing/stretching frequencies.

Peak Shift

For a varied approach, consider shifting individual peaks instead of just a single range. For this method, it is important that to extract the same number of key peaks from both audio samples. Figure 3.16 illustrates how this would be carried out. If n peaks are considered, a shift of $n + 1$ ranges of frequencies would occur. The intervals between peaks would compress or stretch based on where the target's key peaks are located.

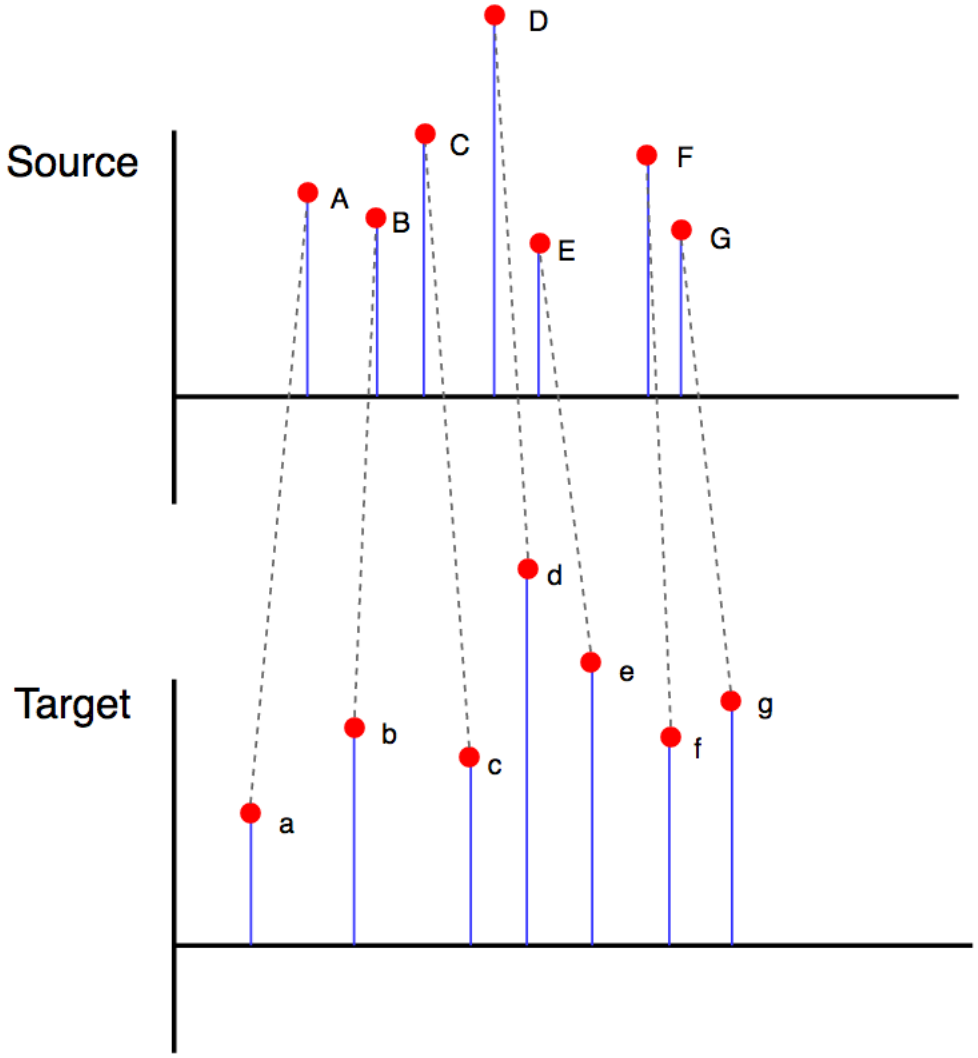

Figure 3.16: Shifting source's key peaks to target's key peaks.

Revisit the example comparing the guitar and mandolin. Figure 3.17 shows 9 key frequencies extracted from both instruments. The goal now is to move those peaks from the guitar to the peaks in the mandolin by compressing and stretching the intervals between the peaks. Figure 3.18 shows the FFT of a third audio sample of a guitar and the desired shift of mapping between the peaks in Figure 3.17.

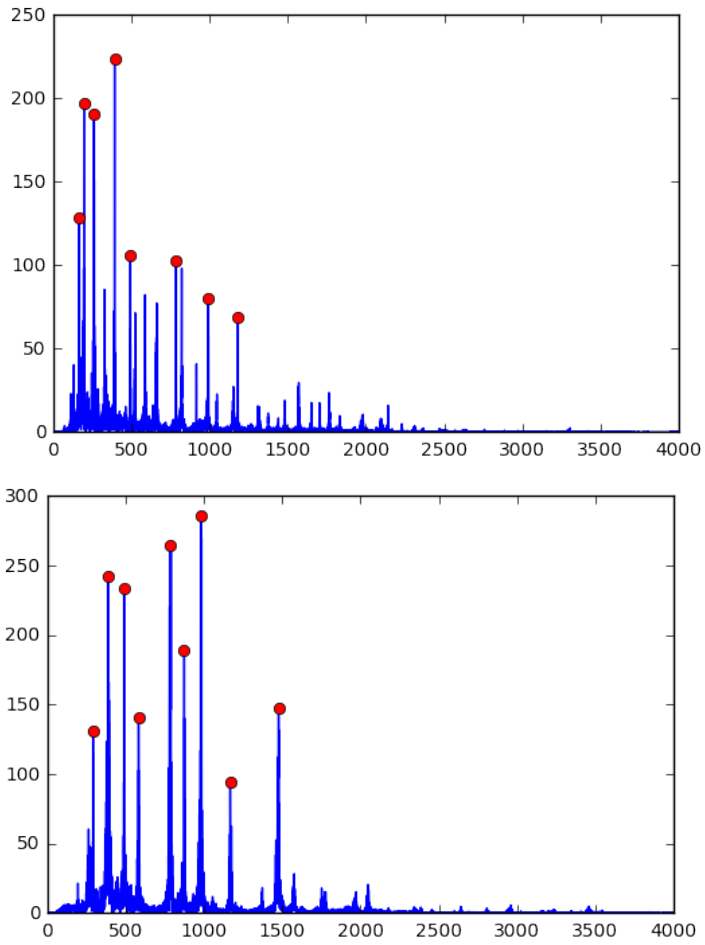

Figure 3.17: Key peaks for guitar (top) and mandolin (bottom).

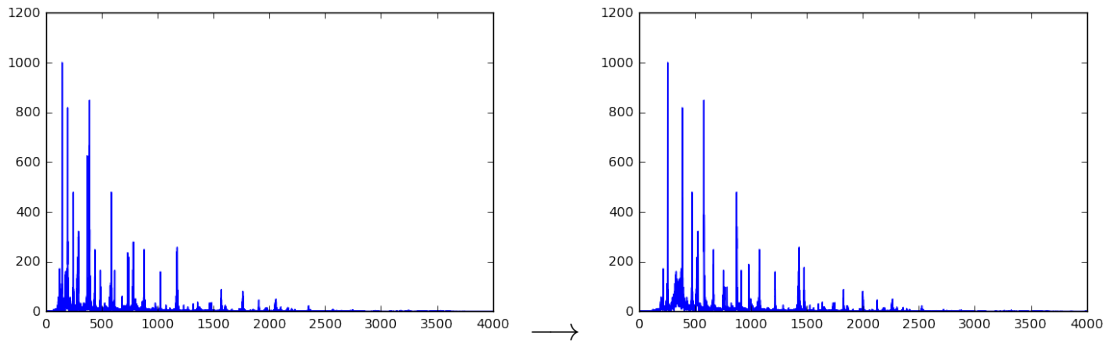

Figure 3.18: Shifting of a guitar FFT based on mapping of peaks from guitar to mandolin.

Chapter 4

RESULTS

Regression

Recall from the previous chapter, the first method discussed for constructing a mapping for vocal frequencies was by using regression. The results of this method showed that regression proved to be an ineffective way in determining a relationship between the two voices. Firstly, there is the issue that this regression is constructed between the magnitudes of the two sample's FFT coefficients. This magnitude loses the information such as the phase shift. Once regression has produced a polynomial for mapping the coefficients, your third audio sample will be filtered through that function. The issue is that it results in new coefficients that only have the magnitude information. It wasn't until taking the inverse Fourier transform that it becomes apparent at how important the phase shift is to build the audio signal.

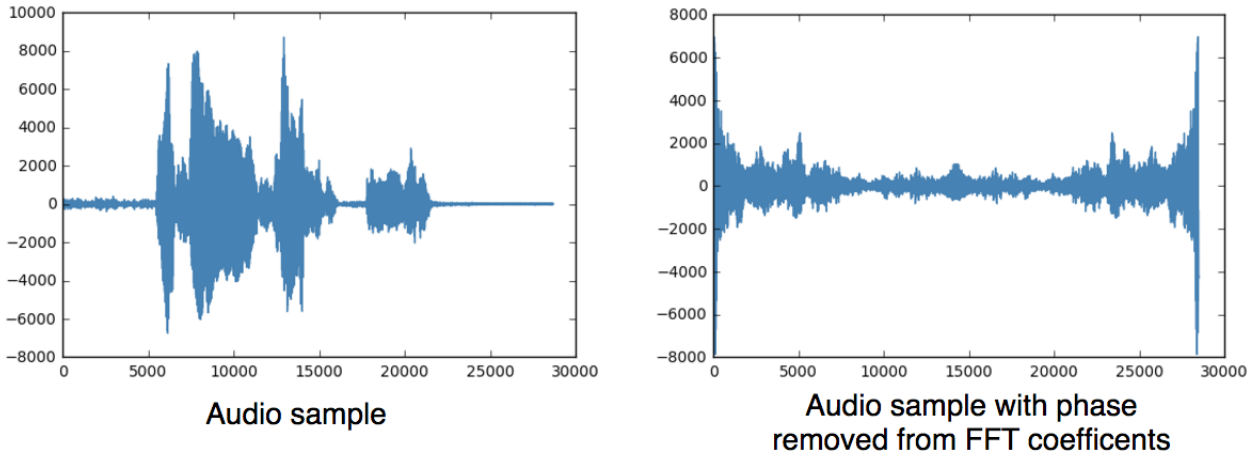

Figure 4.1: Loss of phase shift in signal

If you take an FFT of a signal, convert the coefficients to magnitudes, and then take the IFFT, the result will be a very strange set of points that do not resemble the original signal. Figure 4.1 show how different that wave will become when you take the magnitude's IFFT.

The loss of phase shift drastically affects audio. In fact, listening to this audio will sound as if the voice is being played forward and backward as well as overlapping itself. Another reason this method of regression isn't a proper route would be that it gets more challenging with complex voice data. In the previous chapter, the regression was performed on instruments. These instrument samples only produced as many as four unique key frequencies. When it comes to speech, the Fourier space has much more happening. There are many more frequencies being combined in the audio sample of someone speaking. This could result in comparing many more key frequencies. The order of the polynomial would increase in size making it a very sensitive model. The order would increase but the domain over which you are constructing it would essentially stay the same. Matrix $\mathbf{A}$ in the equation $\mathbf{Ax} = \mathbf{b}$ would become an increasingly ill-conditioned matrix making the system very sensitive when solving for $\mathbf{x}$.

Shifting Coefficients

The previous chapter discussed several different methods in shifting coefficients. The method discussed about the ratio of magnitudes is an attempt to have a vertical shift in the audio samples FFT diagram. This vertical shift should really be more thought of as an extension or shortening of the coefficients. The issue with this method is that frequencies from the source are not being moved toward the frequencies of the target. The result in listening to this mapping just produced a lot of noise. Suppressing the size of the ratio between the two samples is able to create an audible result, however, it does not do anything in the terms of changing the voice. Scaling with vertical shifts is not the correct direction for this topic.

There is some success with the linear shifts. A uniform linear shift of all coefficients left or right in the frequency domain produces a change in the voice. Shifting left successfully converts a voice lower into the "Kidnapper voice" and shifting right converts a voice higher into the "Chipmunk voice." The question at hand becomes can you shift the frequencies in

certain spots and maybe shift over a different sized domain? With the several examples discussed in the previous chapter, the result doesn't exactly produce a desired voice change.The audio can sound distorted with extra noise present in the background. It may in part be due to the fact that previously unused frequencies are now showing up. Pockets of silence will have frequencies canceling each other out when they are combined. When they shifting occurs, some unused frequencies are no longer being canceled out which can produce these unwanted noises in the audio.

Chapter 5

CONCLUSION

After many experiments in identifying a relationship between two samples in Fourier space, it appears that it isn't possible to establish a successful result in voice conversion. The Fourier coefficients have shown to be too sensitive when they are moved around too much. Slight changes in Fourier space can produce large ramifications reverting back to audio with an inverse transform. If you consider the the transformation of Fourier coefficients as a matrix $\mathbf{A}$, then it would be reasonable to say that $\mathbf{A}$ is ill-conditioned. The question now becomes what has been learned from these experiments and where to go from now.

For the methods covered in this thesis, an FFT was taken over the whole length of speech. That means the frequencies we are seeing occur over a time of about 4 to 5 seconds. It may perhaps be more effective in identifying a person's unique vocal frequencies if the length of time was stretched longer. More words would be pronounced giving a wider range of speech available. Each person's voice is unique to them and it can be quite complicated. A larger library of words spoken is necssary to identify fundamental properties of thier vocal frequencies [9, 26]. Figure 5.1 shows how a longer sample will appear in both time and Fourier space. The FFT will appear denser than the examples in previous chapters. It has approximately 215,000 points over the domain of 4000 Hz.

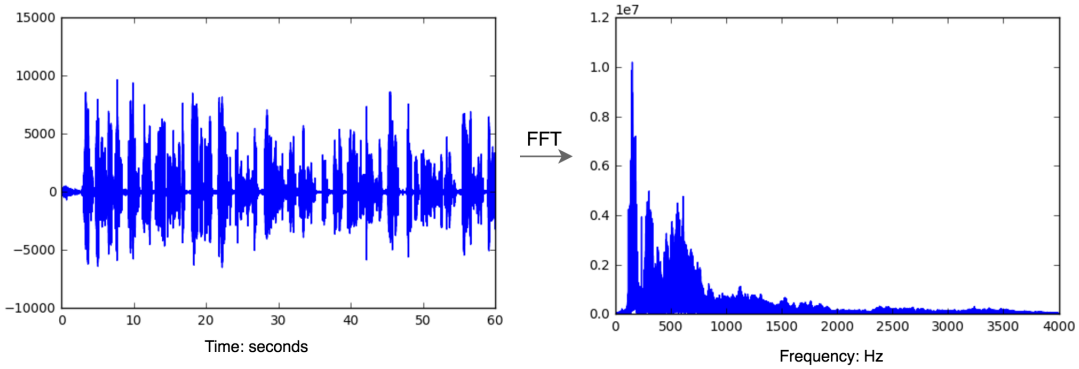

Figure 5.1: FFT of an audio sample recorded over 60 seconds

As discussed in the first chapter, other studies included a large supply of sample audios from both the source and target speaker. As many as 100 different words were spoken in order to best identify a comparison. Also, previous research examined frequencies over time. So the plot will look like an FFT but with an added z-axis that represents time. The next step forward with this research would be modifying the Fourier coefficients at different pockets along the time spectrum.

Another helpful tool that should be investigated is the use of dynamic time warping [4]. This would be helpful in comparing the audio signals between the two speakers. Dynamic time warping (DTW) is helpful in matching up the two speakers words in the same time. When recording two people say the same sample of speech, the words may not appear at the same time along the audio signal because the speakers may talk faster than one another or perhaps one started a little later than the other. Some research has approached using this method of DTW and constructing Gaussian mixture models (GMM) to compare the two audio signals. A good extension to this research would be to explore the usefulness of GMMs. A GMM is constructed similar to the Fourier transform discussed in this research. The idea is to determine coefficients for a set of basis functions that follow normal models. Equation (5.1) represents this sum of basis functions. The value a_k are the coefficients found for each mean μ_k and variance Σ_k at point k.

$$p(x) = \sum_{k=1}^{N} a_k \cdot N(\mu_k, \Sigma_k) \tag{5.1}$$

The idea is to construct two probability distributions, call them $p(x)$ (source speaker) and $p(y)$ (target speaker). Once the distributions have been established, find the conditional relationship: $E[p(y)|p(x)]$. This relationship will then be applied to a third audio sample to perform a voice conversion. In pursuing this method, it would be useful to consider both

applying the GMM to the time domain signal as well as the frequency spectrum in Fourier space.

Figure 5.2 illustrates how a wave form can be modeled by combining seperate Gaussian functions. Each Gaussian function is defined as:

$$f(x) = a \cdot e^{\left(\frac{-(x-\mu)^2}{2\sigma^2}\right)}$$

The mean μ represents where the function is centered and the variance σ^2 determines how spread out the function is. The coefficient a represents the height of the function at its mean.

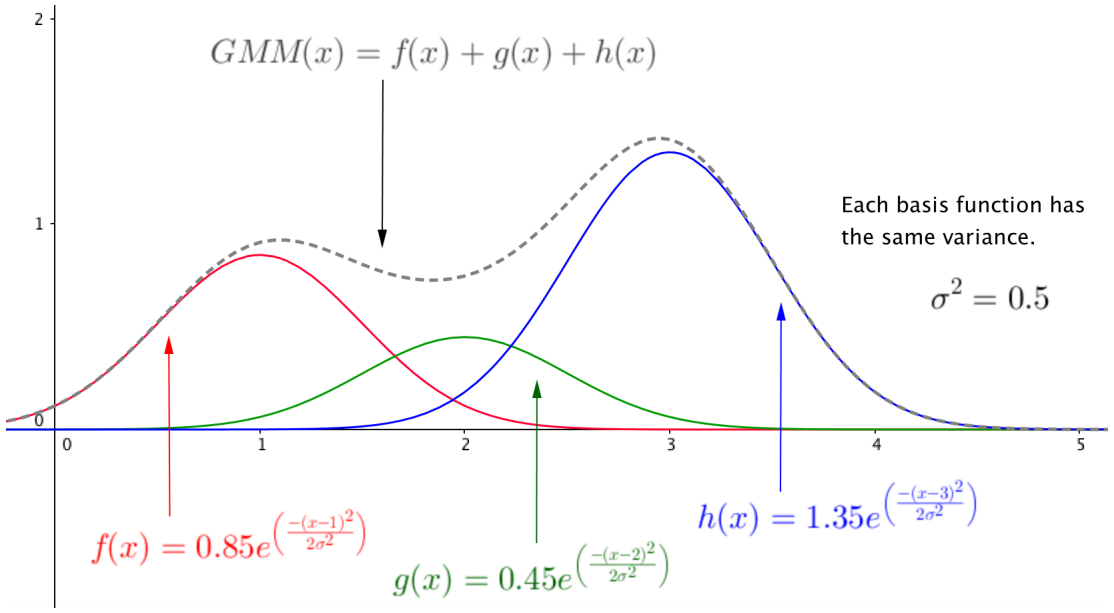

Figure 5.2: Mixed model of three Gaussian functions

The next step forward to continue this research would be to explore the GMM on the audio signal. Since the mapping would be done on the signal instead of the Fourier transform, the complex coefficients wouldn't be a concern. While Figure 5.2 has fixed variance as well as sequential means, the ideal method would include identifying the appropriate mean and

variance for each basis function based on the signal given. Hopefully a path down this direction will yield positive results in voice conversion.

REFERENCES

[1] M. Abe, S. Nakamura, K. Shikano, and H. Kuwabara, *Voice Conversion through vector quantization*, J. Acoust. Soc. Jpn. (E), 11, (1990), pp. 71-76.

[2] H. Benisty, D. Malah, and K. Crammer, *Grid-based approximation for voice conversion in low resource environments*, EURASIP Journal on Audio, Speech, and Music Processing, 3, (2016).

[3] D. Bressoud, *A Radical Approach to Real Analysis*, MAA, Washington, DC, (1994).

[4] D. Erro, A. Moreno, and A. Bonafonte, *Voice conversion based on weighted frequency warping*, IEEE Transactions on Audio, Speech, and Language Processing, 18, (2010), pp.922-931.

[5] H. Gu and S. Tsai, *A voice conversion method combining segmental GMM mapping with target frame selection*, Journal of Information Science and Engineering, 31, (2015), pp. 609-626.

[6] A. Kain, *High Resolution voice transformation*, Ph.D. dissertation, Oregon Health and Science University, 2001.

[7] H. Kawahara, *Speech representation and transformation using adaptive interpolation of weighted spectrum: vocoder revisited*, Proc. ICASSP, Munich, Germany, 1997, pp. 1303-1306.

[8] H. Kawahara, I. Masuda-Katsuse, and A. de Cheveigné, *Restructuring speech representations using a pitch-adaptive time-frequency smoothing and an instantaneous-frequency smoothing and an instantaneous-frequency-based F0 extraction: possible role of a repetitive structure in sounds*, Speech Communication, 27, (1999), pp.187-207.

[9] H. Kuwabara and Y. Sagisaka, *Acoustic Characteristics of speaker individuality: Control and Conversion*, Speech Communication, 16, (1995), pp. 165-173.

[10] N. Maeda, H. Banno, S. Kajita, K. Takeda, and F. Itakura, *Speaker conversion through non-linear frequency warping of STRAIGHT spectrum*, Proc. EUROSPEECH, Budapest, Hungary, 1999, pp. 827-830.

[11] Yu. V. Marapulets, A. B. Tristanov, and B. M. Shevtsov, *Time-frequency analysis of sound range acoustic emission by the sparse approximation method*, Doklady Earth Sciences, 456, (2014), pp. 705-708.

[12] R. J. McAulay and T. F. Quatieri, *Speech analysis/synthesis based on a sinusoidal representation*, IEEE Trans. Acoust., Speech, Signal Processing, 34, (1986), pp. 744-754.

[13] T. Rossing, *The Science of Sound*, 2nd ed., Addison-Wesley, (1990).

[14] Y. Stylianou, and O. Cappé, *A system voice conversion based on probabilistic classification and a harmonic plus noise model*, Proc. ICASSP, Seattle, Oregon, 1998, pp. 281-284.

[15] Y. Stylianou, O. Cappé, and E. Moulines, *Continuous probabilistic transform for voice conversion*, IEEE Transactions on Speech and Audio Processing, 6, (1998), pp.131-142.

[16] Y. Stylianou, O. Cappé, and E. Moulines, *Statistical methods for voice quality transformation*, Proc. EUROSPEECH, Madrid, Spain, 1995, pp. 447-450.

[17] D. Suendermann, A. Bonafonte, H. Ney, and H. Hoege, *A study on residual prediction techniques for voice conversion*, Proc. ICASSP, (2005), pp.13-16.

[18] D. Sundararajan, *Discrete Fourier Transform : Theory, Algorithms and Applications*, Imperial College Press, (2001).

[19] T. Toda, A. Black, and K. Tokuda, *Voice conversion based on maximum-likelihood estimation of spectral parameter trajectory*, IEEE Transactions on Audio, Speech, and Langauge Processing, 15, (2007), pp. 2222-2235.

[20] T. Toda, H. Saruwatari, and K. Shikano, *STRAIGHT-based voice conversion algorithm based on Gaussian mixture model*, Proc. ICSLP, Beijing, China, 2000, pp. 279-282.

[21] O. Turk and L. M. Arslan, *Robust Processing techniques for voice conversion*, Computer Speech and Language, 20, (2006), pp. 441-467.

[22] J. Wen, S. Zhang and J. Yang, *A fast algorithm for undeetermined mixing matrix identification based on mixture of guassian (MoG) sources model*, Journal of Software, 9, (2014), pp. 184-189.

[23] N. Yadav and V. K. Jain, *Voice conversion using GMM with minimum distance spectral mapping plus amplitude scaling*, i-manager's Journal on Electronics Engineering, 7, (2016), pp. 9-15.

[24] H. Ye and S. Young, *Perceptually weighted linear transformation for voice conversion*, Eurospeech, (2003).

[25] H. Ye and S. Young, *Quality-enhanced voice morphing using maximum likelihood transformations*, IEEE Trans. Audio, Speech, and Language Processing, 14, (2006), pp. 1401-1412.

[26] E. Zetterholm, *Same speaker different voices: Astudy of one impersonator and some of his different imitation*, Proc. Int. Conf. Speech Sci. & Tech, 2006, pp. 70-75.

Appendix A

Code

```
# Import Packages

%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
from findpeaks import findpeaks
import pyaudio
import wave
from pylab import*
from scipy.io import wavfile
def plot(data):
        plt.plot(data, color='steelblue')

# Record a few seconds of audio and save to a WAVE file

CHUNK          = 1024
FORMAT         = pyaudio.paInt16
CHANNELS       = 1
RATE           = 8000
RECORD_SECONDS = 4


FILENAME1 = "output1.wav"


p = pyaudio.PyAudio()


stream = p.open(format=FORMAT,
                channels=CHANNELS,
```

```
                    rate=RATE,
                    input=True,
                    frames_per_buffer=CHUNK)

print("* recording")

frames = []

for i in range(0, int(RATE / CHUNK * RECORD_SECONDS)):
    data = stream.read(CHUNK)
    frames.append(data)

print("* done recording")

stream.stop_stream()
stream.close()
p.terminate()

wf = wave.open(FILENAME1, 'wb')
wf.setnchannels(CHANNELS)
wf.setsampwidth(p.get_sample_size(FORMAT))
wf.setframerate(RATE)
wf.writeframes(b''.join(frames))
wf.close()
```

Play a WAVE file

```
wf1 = wave.open(FILENAME1, 'rb')

p = pyaudio.PyAudio()
```

```
stream = p.open(format=p.get_format_from_width(wf.getsampwidth()),
                channels=wf.getnchannels(),
                rate=wf.getframerate(),
                output=True)

data1 = wf1.readframes(CHUNK)

while data1 != '':
    stream.write(data1)
    data1 = wf1.readframes(CHUNK)

stream.stop_stream()
stream.close()

p.terminate()
```

Read in a WAVE file and plot it

```
rate1, wav_data1 = wavfile.read(FILENAME1)
plot(wav_data1)
```

Convert signal to `int16` type and write an audio file

```
wave2 = wave2.astype(int16)
wavfile.write(FILENAME1,rate1, wave2)
```

Smooth the audio signal

```
#smoothing radius is r
r = 3
wav1 = np.zeros(len(wav_data1))
for i in range(r,len(wav1)-r):
```

```
    wav1[i] = sum(wav_data1[i-r:i+r+1])/(2*r+1)
plot(wav1)
```

Compute the FFT of an audio signal

```
L1 = len(wav_data1)
ff1 = np.linspace(0,8000,L1) # Freq axis for double sided FFT
f1 = np.linspace(0,4000,L1/2) # Freq axis for single sided FFT
xx1 = np.fft.fft(wav_data1)
xx1 = np.abs(xx1)/L1 # Double sided FFT coefficients
x1 = 2*xx1[0:L1/2] # Single sided FFT
```

Construct mapping with Regression

```
#audio 1 has been established with smoothing
L1 = len(wav1)
ff1 = np.linspace(0,8000,L1)
f1 = np.linspace(0,4000,L1/2)
xx1 = np.fft.fft(wav1)
xx1 = np.abs(real(xx1))/L1
x1 = 2*xx1[0:L1/2]
max1 = max(x1)
x1 = x1/max1
peaks1 = findpeaks(x1,50,.2) #pick peaks above threshhold of 20%
plt.plot(f1,x1)
plt.plot(f1[peaks1],x1[peaks1],'ro') #plot peaks
plt.show()
freq1 = f1[peaks1]
coef1 = x1[peaks1]*max1 # coefficients that will be used in regression


#audio 2 has been established with smoothing
```

```
L2 = len(wav2)
ff2 = np.linspace(0,8000,L2)
f2 = np.linspace(0,4000,L2/2)
xx2 = np.fft.fft(wav2)
xx2 = np.abs(real(xx2))/L2
x2 = 2*xx2[0:L2/2]
max2 = max(x2)
x2 = x2/max2
peaks2 = findpeaks(x2,50,0.22)
plt.plot(f2,x2)
plt.plot(f2[peaks2],x2[peaks2],'ro') #plot peaks
plt.show()
freq2 = f2[peaks2]
coef2 = x2[peaks2]*max2 # coefficients for the second audio

# include endpoints for regression
n = len(coef1)
C1 = np.zeros(n+2)
C1[1:n+1] = coef1
C1[n+1] = 1.2*C1[n]
C2 = np.zeros(n+2)
C2[1:n+1] = coef2
C2[n+1] = 1.2*C2[n]
plt.plot(C1,C2,'o') #plot coefficients

# Construct polynomial or order n-1
X = np.asmatrix(C1)
Y = np.asmatrix(C2)
b = np.transpose(Y)
```

```
A = np.zeros([n,n])
A = np.asmatrix(A)
for i in range(0,n):
    for j in range(0,n):
        A[i,j] = X[0,i]**(n-j-1)
coef = A**(-1)*b

#plot best fit line to coefficients
xaxis = np.linspace(0,C1[-1],1000)
yaxis = np.linspace(0,0,1000)
for i in range(0,1000):
    for j in range(n):
        yaxis[i] += coef[j]*xaxis[i]**(n-j-1)
plt.plot(X,Y,'ro')
plt.plot(xaxis,yaxis)

#Find FFT of third audio sample
L3 = len(wav_data3)
ff3 = np.linspace(0,8000,L3)
f3 = np.linspace(0,4000,L3/2)
xx3 = np.fft.fft(wav_data3)
xx3 = np.abs(real(xx3))/L3
x3 = 2*xx3[0:L3/2]

#push coefficients through regression model found from vector "coef"
n = len(coef)
ynew = np.linspace(0,0,L3/2)
for i in range(L3/2):
    for j in range(n):
```

```
        ynew[i] += coef[j]*x3[i]**(n-j-1)
plt.plot(f3,yaxis) # plot the transformation of the coefficients
```

Scalar multiplication by ratio of magnitudes

```
L1 = len(wav_data1)
f1 = np.linspace(0,8000,L1)
x1 = np.fft.fft(wav_data1)
X1 = np.abs(x1)/L1
L2 = len(wav_data2)
f2 = np.linspace(0,8000,L2)
x2 = np.fft.fft(wav_data2)
X2 = np.abs(x2)/L2
ratio = abs(x2)/abs(x1)
Freq = np.linspace(0,8000,L1)
plt.plot(Freq,ratio) #plot ratio over double sided FFT spectrum

#This will help us cut down on the ratio.
#You need to pick values of h>1.
#The closer to 1, the more it shrinks
#The farther h is from 1, a the smaller the reduction
h = 1.25
Ratio = np.zeros(len(ratio))
for i in range(len(ratio)):
    Ratio[i] = ((h-1)*ratio[i] + 1)/h
plt.plot(Freq,Ratio)

#Apply ratio multiplication on third audio sample
L3 = len(wav_data3)
f3 = np.linspace(0,8000,L3)
```

```
x3 = np.fft.fft(wav_data3)
x3 = x3*Ratio
```

Linear shift of frequencies

```
#shift out by t*8000/length(wave) frequencies (left shift)
t = 300
Y = np.fft.fft(wav_data1)
L = len(Y)/2
X = np.zeros(2*L)
X = X.astype(dtype=np.complex)
X[0:L-t] = Y[t:L]
X[t+L:2*L] = Y[L:2*L-t]
y1 = np.fft.ifft(X) #inverse transform


#shift in by t*8000/length(wave) frequencies (right shift)
t = 300
Y = np.fft.fft(wav_data1)
L = len(Y)/2
X = np.zeros(2*L)
X = X.astype(dtype=np.complex)
X[t:L] = Y[0:L-t]
X[L:2*L-t] = Y[L+t:2*L]
y1 = np.fft.ifft(X) #inverse transform
```

Identify shift lengths between peaks

```
#peaks have already been selected from both audio samples
#they are defined as peaks1 and peaks2
L = len(wav_data1)
n = len(peaks1)
```

```
range1 = np.zeros(n+2)
range2 = np.zeros(n+2)
range1[n+1] = L-peaks1[n-1]
range2[n+1] = L-peaks2[n-1]
for i in range(0,n):
    range1[i+1] = peaks1[i]-range1[i]
    range2[i+1] = peaks2[i]-range2[i]
print range1
print range2
#Find the scale to shrink or stretch
scale = range1/range2
print scale
range1 = range1.astype(np.int16)
range2 = range2.astype(np.int16)
```

Shift only the range of peaks

```
#this only shifts for the single sided FFT
#the data will need to be made symmetric afterwards
n = len(peaks1)
X3 = np.fft.fft(wav_data3)
L = len(X3)/2
new = np.zeros(L)
new = new.astype(dtype=np.complex)
if peaks1[0] < peaks2[0]:
    new[0:peaks1[0]] = X3[0:peaks1[0]]
else:
    new[0:peaks2[0]] = X3[0:peaks2[0]]
if peaks1[1] < peaks2[1]:
    new[peaks2[1]:L] = X3[peaks2[1]:L]
```

```
else:
    new[peaks1[1]:L] = X3[peaks1[1]:L]
k = 0
for i in range(peaks2[0],peaks2[1]):
    index = peaks1[0] + floor(scale[2]*k)
    k+=1
    index = int(index)
    new[i] = X3[index]

#run this if you wish to replace duplicates with zeros
for j in range(peaks2[0],peaks2[1]):
    if new[j] == new[j+1]:
        new[j] = 0.0
```

Shift all peaks by stretching and shrinking multiple ranges

```
#this only shifts for the single sided FFT
#the data will need to be made symmetric afterwards
n = len(peaks1)
X3 = np.fft.fft(wav_data3)
new = np.zeros(len(X3)/2)
new = new.astype(dtype=np.complex)
Peaks1 = np.zeros(n+2)
Peaks1[1:n+1] = peaks1
Peaks1[n+1] = len(X3)/2
Peaks1 = Peaks1.astype(np.int16)
Peaks2 = np.zeros(n+2)
Peaks2[1:n+1] = peaks2
Peaks2[n+1] = len(X3)/2
Peaks2 = Peaks2.astype(np.int16)
```

```
for j in range(0,n+1):
    k=0
    for i in range(Peaks2[j],Peaks2[j+1]):
        index = Peaks1[j]+floor(scale[j+1]*k)
        k+=1
        index = int(index)
        new[i] = X3[index]


#run this if you wish to replace duplicates with zeros
for j in range(len(new)-1):
    if new[j] == new[j+1]:
        new[j] = 0.0
```

Make data summetric

```
#this creates the double sided FFT from the single side
L = len(new)
x3 = np.zeros(2*L)
x3 = x3.astype(dtype=np.complex)
x3[0] = new[0]
for i in range(1,L):
    x3[i] = new[i]
    x3[2*L-i] = complex(real(new[i]),-imag(new[i]))
plot(x3) #plot double sided FFT
```

Construct Gaussian Mixture Model

```
# The size of the audio sample is the number of basis functions
def gaussian(x,wave):
    y = 0.0
    SD = 0.5
```

```
    for i in range(L):
        y += wave[i]*exp(-(x-i)**2/(2*SD**2))
    return y
x = np.linspace(0,L,L)
y1 = gaussian(x,wav_data1)

# The number of basis functions can be defined to be length n
# n must be less than or equal to the length of the audio signal
def gaus(x,n,L):
    y = 0.0
    SD = 0.5
    j = 0
    for i in range(n):
        y += wav_data1[j]*exp(-(x-j)**2/(2*SD**2))
        j += L/n
    return y
n=1000
x = np.linspace(0,L,L)
y1 = gaus(x,n,wav_data1)
```

findpeaks function that extracts peaks

```
""" Searches for peaks in data
     History:
          -nov 2015: Janko Slavic, update
          -mar 2013: janko.slavic@fs.uni-lj.si
"""

import numpy as np
```

```
def findpeaks(data, spacing=1, limit=None):
"""Finds peaks in `data` which are of `spacing` width and >=`limit`.
:param data: values
:param spacing: minimum spacing to the next peak (should be 1 or more)
:param limit: peaks should have value greater or equal
:return:
"""
    len = data.size
    x = np.zeros(len+2*spacing)
    x[:spacing] = data[0]-1.e-6
    x[-spacing:] = data[-1]-1.e-6
    x[spacing:spacing+len] = data
    peak_candidate = np.zeros(len)
    peak_candidate[:] = True
    for s in range(spacing):
        start = spacing - s - 1
        h_b = x[start : start + len]  # before
        start = spacing
        h_c = x[start : start + len]  # central
        start = spacing + s + 1
        h_a = x[start : start + len]  # after
        peak_candidate = np.logical_and(peak_candidate, np.logical_and
(h_c > h_b, h_c > h_a))

    ind = np.argwhere(peak_candidate)
    ind = ind.reshape(ind.size)
    if limit is not None:
        ind = ind[data[ind] > limit]
```

```
    return ind


if __name__ == '__main__':
    import matplotlib.pyplot as plt

    n = 80
    m = 20
    limit = 0
    spacing = 3
    t = np.linspace(0., 1, n)
    x = np.zeros(n)
    np.random.seed(0)
    phase = 2 * np.pi * np.random.random(m)
    for i in range(m):
        x += np.sin(phase[i] + 2 * np.pi * t * i)

    peaks = findpeaks(x, spacing=spacing, limit=limit)
    plt.plot(t, x)
    plt.axhline(limit, color='r')
    plt.plot(t[peaks], x[peaks], 'ro')
    plt.title('Peaks: minimum value {limit}, minimum spacing {spacing}
points'.format(**{'limit': limit, 'spacing': spacing}))
    plt.show()
```