A Survey of Various Data Compression Techniques

Charles Alexander Smith

April 1, 2010

1 Introduction

This paper is a survey of various methods of data compression. When the computer age came about in the 1940's, storage space became an issue. Data compression was the answer to that problem. The compression process takes an original data set and reduces its size by taking out unnecessary data. There are two main types of compression, lossy and lossless. This paper will deal exclusively with lossy compression. So, through the compression/decompression algorithm, some data from the original file is deleted and not recovered. There are many methods of compression, and this paper will go into depth about compression using Fourier Transformations and Wavelet Compression. Each of these methods has strengths and weaknesses in regards to different types of file; the paper will also cover those aspects of the two methods.

2 Background

2.1 History

The beginnings of data compression began as early as 1838 with its use is morse code.[1] Letters that were common in English were given shorter codes to save time as the messages were being typed.[1] Over 100 years later, as the computer ago was on the rise, this simple morse code method, was built upon and became a study that is known as Information Theory. However, even before Information Theory, a mathematician by the name of Joseph Fourier observed that "any sufficiently smooth function could be decomposed into sums of sine waves with frequencies corresponding to successive integers."[2] Fourier's method was applied to sounds. After breaking a function into frequencies, Fourier was able to drop the highest and lowest frequencies, but keep the rest. This change was unable to be heard by the human ear, and space was able to be saved during recordings. This methodology is still used today in Compression algorithms. This method was also applied to images in the 1950's, in the attempt to decrease the amount of data that needed to be sent to television; however, no solution was given at that time. In fact, it took another 30 years until the notion of image compression became prevelant in mainstream technology.[2] The area of Wavelet Compression was founded by Alfred Haar, a Hungarian Mathematician, in the early 20th Century.[3] Haar created the first known wavelet, which is now known as a Haar Wavelet. This area of compression has grown rapidly since the 1970's when interest in Wavelets and their uses began to spread through the mathematical field. Fractal compression is a lossy image compression, which is has proven to be more efficient than image compression using Fourier transformations. In 1987, Michael Barnsley was at the top of field in the development of fractal compression, and currently has many patents on the algorithms.[4] Fractal Compression has incredible compression ratios, which is an attractive aspect to much commercial use, especially to companies that transfer much data over the internet. Many companies, from Microsoft to NetFlix, now use this type of algorithm. Without this method, streaming information over the internet would take many times longer. Although Compressive Sensing has been studied for over 40 years, it did not see significant strides until 2004.[5] At that time, a mathematician, by the name of Emmanuel J. Candes, was performing research with magnetic resonance imaging.[5] He found that an image could be reconstructed even when the data seemed insufficient by the Nyquist-Shannon criterion.[5] This criterion states that any bandlimited signal can be reconstructed if the sampling rate, per second, is greater than twice the highest frequency in the original file.[6] With this finding, the area of compressive sensing exploded, and is a current hot-spot in Information Theory.

2.2 Mathematical Background

In order for us to understand the theory behind compression and decompression, it is necessary to recall several key ideas from Linear Algebra. These underlying ideas make the compression process possible. Let $\beta = \{\mathbf{b}_1, \mathbf{b}_2, \cdots, \mathbf{b}_k\}$ be a set of vectors in $\mathbb{C}^n$. We say that a *linear combination* of these vectors is any expression of the form

$$c_1\mathbf{b}_1 + c_2\mathbf{b}_2 + \cdots + c_k\mathbf{b}_k$$

where $c_1, c_2, \ldots, c_k$ are scalars. A set of vectors are *linearly independent* if

$$\sum_{j=1}^{k} c_j\mathbf{b}_j = \mathbf{0} \text{ implies that } c_j = 0 \text{ for all } j = 1, 2, \ldots, k.$$

Additionally, the set of all linear combinations of the vectors in β is called the *span* of β. A *basis* is a set of vectors that are both linearly independent and spanning.

The *conjugate* of a complex number $z = a + bi$ will be denoted by $\overline{z} = a - bi$. If $\mathbf{B}$ is an $n \times k$ matrix with complex entries,

$$\mathbf{B} = (b_{j,l} : j = 1, 2, \ldots, n,\ l = 1, 2, \ldots k,)$$

then the *conjugate transpose* of $\mathbf{B}$ is the $k \times n$ matrix $\mathbf{B}^*$ given by

$$\mathbf{B}^* = \left(\overline{b_{l,j}} : l = 1, 2, \ldots k,\ j = 1, 2, \ldots, n\right).$$

The *inner product* of two vectors $\mathbf{b}_j$ and $\mathbf{b}_l$ is defined by

$$(\mathbf{b}_j, \mathbf{b}_l) = \mathbf{b}_l^*\mathbf{b}_j$$

Note that the inner product of two vectors is a single complex number. Two vectors are said to be *orthogonal* if their inner product is equal to zero. A set of non-zero vectors is said to be orthogonal if every pair of distinct vectors from it are orthogonal.

The length or *norm* of a vector $\mathbf{b}$ is defined by

$$\|\mathbf{b}\| = \sqrt{\mathbf{b}^*\mathbf{b}}.$$

A set β of non-zero vectors is said to be *orthonormal* if β is orthogonal and the length of each vector in β is one.

Theorem 1. *If $\{\mathbf{b}_1 \cdots \mathbf{b}_k\}$ is orthogonal, then $\{\mathbf{b}_1 \cdots \mathbf{b}_k\}$ is linearly independent.*

Proof. First, assume that $\mathbf{c}_1\mathbf{b}_1 + \cdots + \mathbf{c}_k\mathbf{b}_k = 0$. To show linear independence, we must show that $c_1 = c_2 = \cdots = c_n = 0$. To do this, fix a $j \in \{1, 2, \cdots, n\}$. Now, compute the inner product of our first equation with $\mathbf{b}_j^*$.

$$\mathbf{b}_j^*\left(\mathbf{c}_1\mathbf{b}_1 + \cdots + \mathbf{c}_j\mathbf{b}_j + \cdots + \mathbf{c}_k\mathbf{b}_k\right) = \mathbf{b}_j^*0$$

Using the distributive law, we obtain the simplified equation

$$\mathbf{c}_1\mathbf{b}_j^*\mathbf{b}_1 + \cdots + \mathbf{c}_j\mathbf{b}_j^*\mathbf{b}_j + \cdots + \mathbf{c}_k\mathbf{b}_j^*\mathbf{b}_k = 0 \tag{1}$$

And by orthogonality,

$$\mathbf{b}_j^*\mathbf{b}_l = \begin{cases} 0 \text{ if } j \neq l \\ \|\mathbf{b}_j\|^2 \text{ if } j = l \end{cases}$$

So, using this information in conjunction with (1),

$$\mathbf{c}_1(0) + \cdots + \mathbf{c}_j\|\mathbf{b}_j\|^2 + \cdots + \mathbf{c}_k(0) = 0$$

This shows that $\mathbf{c}_j\|\mathbf{b}_j\|^2 = 0$; however, since we know that $\mathbf{b}_j$ is nonzero, it must be true that $\mathbf{c}_j = 0$. And, since j was arbitrary in $\{1, 2, \cdots, k\}$, we have completed our proof. $\square$

Corollary 1. *If $\{\mathbf{b}_1 \cdots \mathbf{b}_n\}$ is an orthogonal set in $\mathbb{C}^n$, then it is a basis for $\mathbb{C}^n$.*

Proof. By Theorem 1, the n vectors $\{\mathbf{b}_1, \mathbf{b}_2, \cdots, \mathbf{b}_n\}$ are linearly independent. Since the dimension of $\mathbb{C}^n$ is n, it follows that the vectors must form a basis for $\mathbb{C}^n$. □

Given the basis $\beta = \{\mathbf{b}_1 \cdots \mathbf{b}_n\}$ in $\mathbb{C}^n$, we will define the *basis matrix*, $\mathbf{B}$, as the $n \times n$ matrix whose columns are composed of the basis vectors $\mathbf{b}_i$, for $i = 1, 2, \cdots, n$.

Theorem 2. *If β is a basis in $\mathbb{C}^n$, then the associated basis matrix, $\mathbf{B}$, is invertible.*

Proof. From basic linear algebra, we know that $\mathbf{B}$ will be invertible if and only if $\mathbf{B}x = \mathbf{s}$ has a solution for all $\mathbf{s}$ in $\mathbb{C}^n$. Let $\mathbf{s} \in \mathbb{C}^n$. Since β is a basis for $\mathbb{C}^n$, there exists a vector $\mathbf{x}$ whose components are the scalars x_1, x_2, $\cdots$, x_n, such that

$$\mathbf{s} = x_1\mathbf{b}_1 + x_2\mathbf{b}_2 + \cdots + x_n\mathbf{b}_n$$

So, $\mathbf{B}x = \mathbf{s}$ has the solution $x = c$, where c is a vector of scalars that satisfies the above equation. To show uniqueness, suppose there exists a vector y such that $y \neq x$ and $\mathbf{B}y = \mathbf{s}$. Then, it is true that

$$\mathbf{B}(c - y) = \mathbf{B}c - \mathbf{B}y = s - s = 0,$$

which yields

$$(c_1 - y_1)\mathbf{b}_1 + \cdots + (c_n - y_n)\mathbf{b}_n$$

Since all of the indices are linearly independent, we say that $c_i - y_i = 0$ for all i in $\mathbb{N}$. However, this introduces a contradiction since $y = c = x$. This shows uniqueness, which proves that $\mathbf{B}$ is invertible. □

Theorem 3. *Let β be an orthogonal basis in $\mathbb{C}^n$ with the associated basis matrix $\mathbf{B}$. Then $\mathbf{B}^*\mathbf{B}$ is a diagonal matrix.*

Proof. The (j, l) entry of $\mathbf{B}^*\mathbf{B}$ is $\mathbf{b}_j^*\mathbf{b}_l$. However, because the vectors are orthogonal, we have

$$\mathbf{b}_j^*\mathbf{b}_l = \begin{cases} 0 & \text{if } j \neq l \\ \|\mathbf{b}_j\|^2 & \text{if } j = l \end{cases}$$

Hence,

$$\mathbf{B}^*\mathbf{B} = \begin{bmatrix} \|\mathbf{b}_1\|^2 & 0 & \cdots & 0 \\ 0 & \|\mathbf{b}_2\|^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \|\mathbf{b}_n\|^2 \end{bmatrix}$$

□

Corollary 2. *If $\{\mathbf{b}_1 \cdots \mathbf{b}_n\}$ is an orthonormal basis in $\mathbb{C}^n$, then the associated basis matrix $\mathbf{B}$ is orthonormal.*

Proof. We need to show that $\mathbf{B}^*\mathbf{B} = I$, where I is the $n \times n$ identity matrix. However, this directly follows from Theorem 3 and the property of orthonormality. □

Note the implication that if the columns of $\mathbf{B}$ are orthonormal, then $\mathbf{B}^{-1} = \mathbf{B}^*$.

3 Overview of the Compression/Decompression Process

For our purposes a *signal* is a vector $\mathbf{s}$ in $\mathbb{C}^n$. Usually, n will be quite large. The individual entries in a signal, $\mathbf{s}$, are *samples* that represent the values of some continuous waveform, $s(t)$, at discrete times. We will always assume that the samples are evenly spaced in time; in other words, we assume that they were obtained at a constant *sampling frequency*, which we will denote as f_s. For example, if we sample an audio signal, $s(t)$, at the standard rate of $f_s = 44,100$ hertz for $T = 5$ seconds, the result will yield a signal vector $\mathbf{s}$ with $n = f_sT = 220,500$ samples.

Both the Fourier and wavelet compression algorithms studied in this project have equivalent overall structure. The original signal, $\mathbf{s}$, is written in terms of some basis, β, in $\mathbb{C}^n$:

$$\mathbf{s} = \mathbf{Bc} = c_1\mathbf{b}_1 + c_2\mathbf{b}_2 + \cdots + c_n\mathbf{b}_n \tag{2}$$

The vector $\mathbf{c} = (c_1, c_2, \cdots, c_n)$ is called the *coefficient vector* of $\mathbf{s}$ with respect to the basis β. By the results of the previous section, we have shown that $\mathbf{c} = \mathbf{B}^{-1}\mathbf{s}$. Depending on the nature of the signal, $\mathbf{s}$, and the basis, β, it is likely to occur that many of the components of the coefficient vector are close to zero. If this is the case, those coefficients that are close to zero are able to be ignored, and an accurate approximation of the original signal can be created by saving only the coefficients that are "large enough". A thresholding function is used to determine what "large enough" means in the context of any given compression. Thus, compression algorithms all have the same top-level form:

$$\text{Original Signal} \rightarrow \text{Linear Transform} \rightarrow \text{Thresholding} \rightarrow \text{Compressed Signal}$$

The process of reconstructing the now compressed signal is simply taking the linear combination of basis vectors that are specified by the compressed coefficients:

$$\text{Compressed Signal} \rightarrow \text{Linear Transform} \rightarrow \text{Reconstructed Signal}$$

Now that we have the big-picture idea, we must acquire notation in order to properly model the compression.

The thresholding process is mandatory to the compression process. In this project, we will consider only the "all or nothing" thresholding:

$$\chi_\tau(x) = \begin{cases} x & \text{if } |x| \geq \tau \\ 0 & \text{if } |x| < \tau \end{cases}$$

The thresholding function χ is applied component-wise to the coefficient vector $\mathbf{c}$:

$$\chi_\tau(\mathbf{z}) = (\chi_\tau(\mathbf{z_1}), \chi_\tau(\mathbf{z_2}), \cdots, \chi_\tau(\mathbf{z_n}))\,, \text{for } \mathbf{z} \in \mathbb{C}^n$$

As can be seen from this piece-wise function, if $\chi_\tau(x)$ is below τ, our thresholding value, then that corresponding data will be set to zero. So, for every piece of data that is below the set threshold, additional bytes are removed from the original size of the file; hence, the higher the threshold, the smaller the compressed file, and the lower the threshold, the more space the compressed file will take, but the quality of the file will have more integrity.

After the thresholding takes place, we are left with only the most important signal components since the rest have been set to 0. We will denote these threshold components by $\mathbf{c}_0$:

$$\mathbf{c}_0 = \chi_\tau(c)$$

The actual data compression is achieved by only saving the non-zero components of $\mathbf{c}_0$ along with their indices. Thus, we may visualize the compression process as

$$\mathbf{s} \rightarrow B^{-1} \rightarrow \mathbf{c} \rightarrow \chi_\tau \rightarrow \mathbf{c}_0$$

or equivalently,

$$\mathbf{c}_0 = \chi_\tau\left(\mathbf{B}^{-1}\mathbf{s}\right)$$

However, it is only practical to perform this process if we end with a signal that we are able to use. So, we must decompress $\mathbf{c}_0$ in order to obtain a useful, reconstructed signal. The reconstruction process applies the saved coefficients, $\mathbf{c}_0$, to the basis β, resulting in the reconstructed signal, $\mathbf{s}_0$:

$$\mathbf{s}_0 = \mathbf{Bc}_0 = \mathbf{B}\chi_\tau\mathbf{B}^{-1}\mathbf{s}.$$

A relevant question at this point in the process is how well the reconstructed signal, $\mathbf{s}_0$, approximates the original signal $\mathbf{s}$. One measure of this is the size of the *residual vector* $\mathbf{s} - \mathbf{s}_0$:

$$\|\mathbf{s} - \mathbf{s}_0\| = \left(\sum_{k=1}^{n} \left(\mathbf{s}\,(i) - \mathbf{s}_0\,(i)\right)^2\right)^{1/2}$$

We have the following result relating the size of the residual vector to the signal components that were “thrown away” during the compression process.

Theorem 4. *For any basis β of $\mathbb{C}^n$ we have*

$$\|\mathbf{s}-\mathbf{s}_0\|^2 = (\mathbf{c}-\mathbf{c}_0)^*\mathbf{B}^*\mathbf{B}(\mathbf{c}-\mathbf{c}_0)$$

In particular, if β is an orthonormal basis, then $\|\mathbf{s}-\mathbf{s}_0\| = \|\mathbf{c}-\mathbf{c}_0\|$.

Proof. Recall that

$$\mathbf{s}-\mathbf{s}_0 = \mathbf{B}\left(\mathbf{c}-\mathbf{c}_0\right)$$

So,

$$\|\mathbf{s}-\mathbf{s}_0\|^2 = \left(\mathbf{s}-\mathbf{s}_0\right)^*\left(\mathbf{s}-\mathbf{s}_0\right) = (\mathbf{c}-\mathbf{c}_0)^*\mathbf{B}^*\mathbf{B}(\mathbf{c}-\mathbf{c}_0)$$

□

The ideas that have been introduced in this section will now be applied to specific bases. In Section 4, we will use the Fourier basis, which is derived from the complex exponentials $e^{2\pi i f t}$. In Section 5, we turn to the wavelet basis consisting of shifts and translations of the Haar wavelet. Matlab code that implements compression using these bases has been written, and is included in the appendix. We note that the bases used in both the Fourier and wavelet compression techniques are orthogonal.

4 Fourier Method

The *Fourier Transform* is an operation that takes a complex-valued function and transforms it into another complex-valued function. In this paper, we will denote the Fourier Transform by FT. For one-dimensional transforms, we can view the process as mapping $\mathbb{C}^n$ into itself. These one-dimensional transforms can be applied to audio files. In this type of application, the function begins in the time domain and the FT transforms the data into a set in the frequency domain. This allows us to analyze sound files based on frequencies. Straight away we can see that this is highly appealing for use in data compression. For example, the human ear can only hear frequencies in a certain range; so, using the FT, we can get rid of some data right away. We will now explore the mathematics of the Discrete Fourier Transform. We will do this by identifying the basis, β, in the DFT case. Consider the basis matrix

$$\beta = \mathbb{F} = \left(e^{-2\pi i m n/N} : 0 \le n, m \le N-1\right)$$

where m stands for the row index and n is for the column index. The columns of $\mathbf{B}$ form the Fourier basis, β.

$$\mathbb{F}\mathbf{s} = \mathbf{S}$$

In order for the Fourier basis to be mathematically sound, we must show that it follows the properties from our general basis. The following theorem proves its orthogonality.

Theorem 5. *$\mathbb{F}^*\mathbb{F}$=$\mathbb{F}\mathbb{F}^*$=NI.*

Proof.

$$\begin{aligned}\left(\mathbb{F}^*\mathbb{F}\right)_{j,l} &= \sum_{k=0}^{N-1} \overline{f_{k,j}} f_{k,l} \\ &= \sum_{k=0}^{N-1} e^{2\pi i k j/N} e^{-2\pi i k l/N} \\ &= \sum_{k=0}^{N-1} e^{2\pi i k(j-l)/N}\end{aligned}$$

This presents two cases; one where $j = l$ and the other when $j \neq l$. When $j = l$, each term in the sum is 1, so the sum is N. We will represent the second case, where $j \neq l$, by

$$\sum r^k, \text{ for } r = e^{2\pi i(j-l)/N}$$

Now, by the geometric series

$$\begin{aligned}\sum r^k &= \frac{1-r^N}{1-r} \\ &= \frac{1-e^{2\pi i(j-l)}}{1-e^{2\pi i(j-l)/N}} \\ &= 0\end{aligned}$$

because $e^{2\pi i(j-l)} = 1$.

Therefore,

$$(\mathbb{F}^*\mathbb{F})_{j,l} = \begin{cases} N & \text{if } j = l \\ 0 & \text{if } j \neq l \end{cases}$$

and so, $\mathbb{F}^*\mathbb{F} = NI$. Similarly, $\mathbb{F}\mathbb{F}^* = NI$. $\square$

Corollary 3. $\|\mathbf{s} - \mathbf{s}_0\|^2 = \mathbf{N}\,\|\mathbf{c} - \mathbf{c}_0\|^2$

In addition to being able to compress one-dimensional data, like audio files, we can extend the Fourier Transform to higher dimensions, which allows us to compress image data. Before, we took the Fourier Transform of each sample in a row, saved the data depending on the threshold, and then recreated a row with the kept data. Now, however, we simply extend this process to rows and columns. So, we perform a similar process for the first row, step down a column, perform the process again, and iterate this step until we have gone through the $n \times n$ matrix. An equivalent process is performed when reconstructing the image data.

4.1 Results

The process of compressing a data file can be done in a multitude of ways. In the Matlab code I wrote for this project, I used an "all or nothing" approach. If the data point meets the thresholding requirements, then that sample is kept; otherwise, it is thrown out and replaced with zero. The code first takes the file and transforms it into the frequency domain by the mapping of the Fourier Transform. Some of these frequencies have relatively large amplitudes, while others have rather insignificant amplitudes. The frequencies that have large amplitudes hold more information than the rest; so, it necessary to keep more data points with large amplitudes than low amplitudes. The algorithm used for this project arranges the data points based on the frequencies' amplitudes and then proceeds to threshold according to the user input.

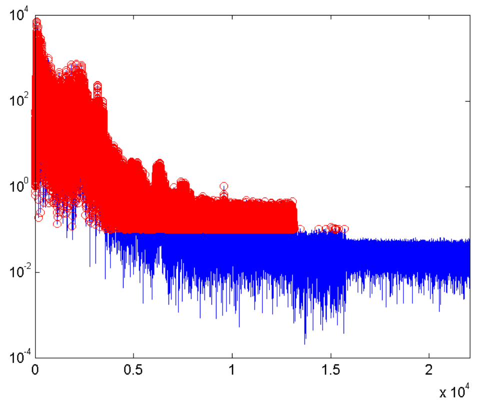


Figure 1: Frequency domain representation of an audio signal, showing the largest 50 percent of frequencies in red.

This figure is extremely useful in visualizing the thresholding process. If we set the thresholding to keep fifty percent of the data, the algorithm finds the least important fifty percent and does not keep it. Thus, only the top fifty percent is used in the recreated signal. Once it has recongized which frequencies are kept and which are thrown away, it creates a new matrix, whose only information is found in the kept frequencies. At this point, the Inverse Fourier Transform is taken, and the signal is able to be played once again.

After the thresholding and recreation takes place, we are able to plot both the original file and the decompressed file superimposed on one another. This allows us to observe discrepancies in the time and frequency domain. The following figure shows this.

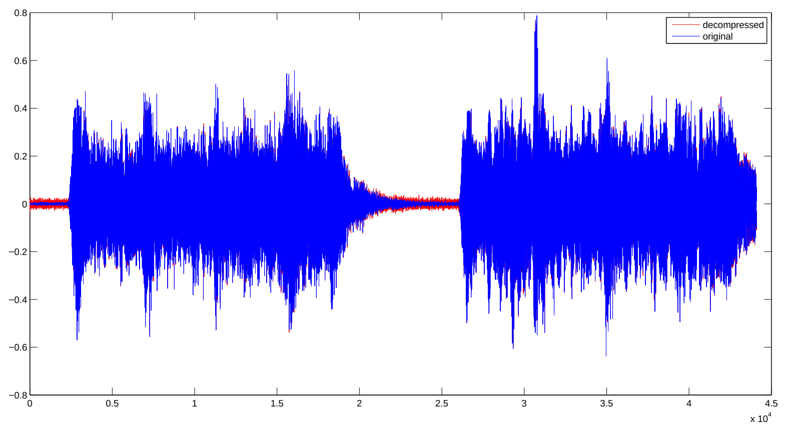


Figure 2: Time domain representation of an original file, blue, and its compressedcounterpart, red. The thresholding ratio is 2-1.

Upon careful observation of this figure, however, we are able to notive a difference between the original file and its decompressed counterpart. This discrepancy will be discussed later in this section.

Throughout much experimentation with data compression using the Fourier Transform, we will find that the method works better with some signals than others. For example, after observing the basis and the general nature of the Fourier transform, it is true that smooth, continuous signals are optimum for this method. However, before exploring different types of signals, we must first have an understanding of the unwanted products that are introduced by the Fourier Transform. This product is the introduction of high frequency noise in the reconstructed signal, which we call an *artifact.* There are certain characteristics that a signal can have that will introduce more artifacts into the reconstruction. For instance, the Fourier Transform does not work well with sharp spikes in amplitude. The sharp difference in amplitude will cause quantization errors that are spread throughout the reconstructed signal.[7] So, the longer the input signal, the more significant the artifacts will become. Before we take a look at these various signal types, first we should be aware that there are methods for lessening the artifacts. The easiest way to deal with this is simply decrease the *compression ratio.* (The compression ratio is the ratio between the size of the compressed signal versus the original signal). As we throw away more samples, we introduce more jumps in amplitude, which is not ideal for the F.T. On the other hand, the lower the ratio, the less efficient our compression is. In my code, the compression ratio is user input, so that the user is able to experiment with what ratios work better with different signals. Another way to decrease the amount of artifacts in the reconstructed code is to compress in *windows.* This method segments the original signal into n parts, and each of these parts is compressed/decompressed individually. So, if the signal is split into n sections, and the third section provides a significant ammount of noise, those artifacts are not compounded in section four or the rest of the sections. Now that we know why the F.T. performs poorly with some signals, we are able to analyze different signal types and compare them based on the quality of the reconstructed signal. The following figure depicts a section of voice in the time domain.

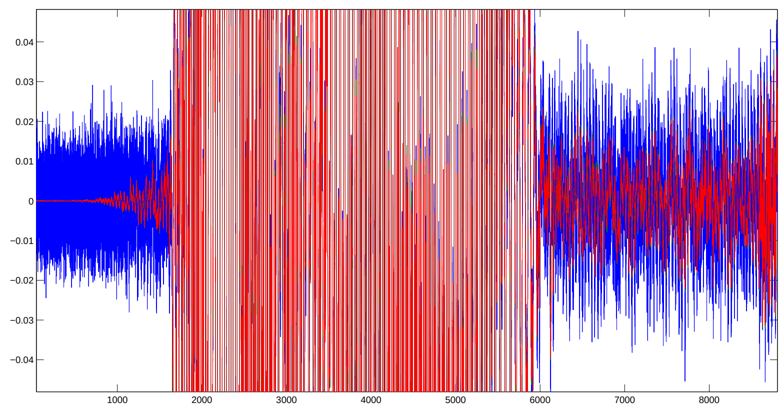


Figure 3: Magnification of an audio signal in the time domain. The red lines correspond to the original signal, and the blue lines represent the reconstructed signal after a compression ration of 4 to 1. In the reconstructed signal, quantization errors introduced from the sharp jump in amplitude just after sample 1500 have been spread throughout the signal.

As you can see, in the "silent" sections, that is, the sections to the left and right, the reconstructed signal contains noise that was not present in the original signal. This is the visual representation of an artifact. This particular artifact was created by the quantization error in the sound-byte, which is compounded the longer the signal is. If we compare Figure 3 with the following figure,

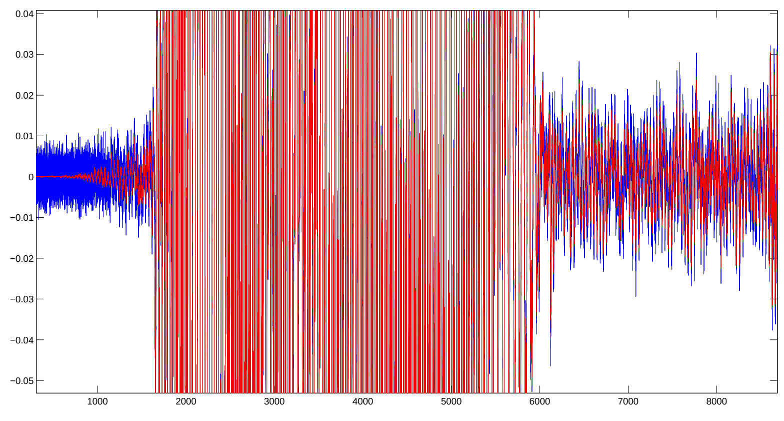


Figure 4: The same section of audio signal from Figure 3, but with a compression ratio of 2 to 1. Quantization errors are still spread throughout this signal, but there are less of them due to the lower compression ratio.

While, at first glance, these two images may appear equal, the reader should note the values on the y-axis of the graph. The artifacts from the first figure have a magnitude rougly three times as large as the second figure. Now, we will view a section from a signal that contains a man humming. The hum will immediately follow dialogue.

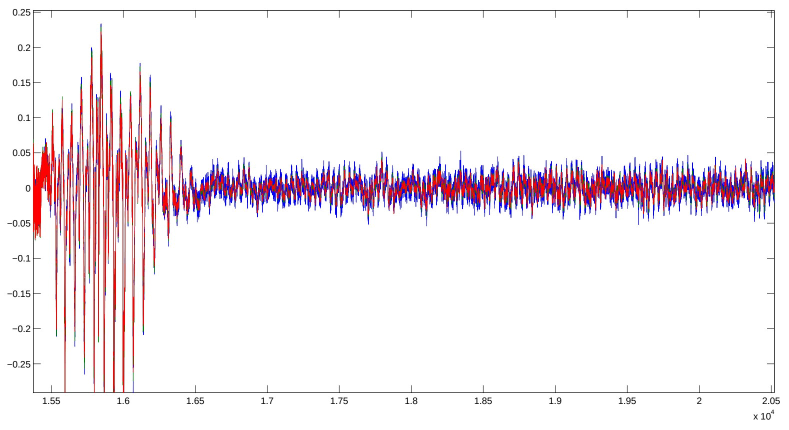


Figure 5: The red lines correspond to the original signal and the blue lines to the reconstructed. This file was compressed at a 4 to 1 ratio.

The following figure represents the same sound-byte, but with a different commpression ratio.

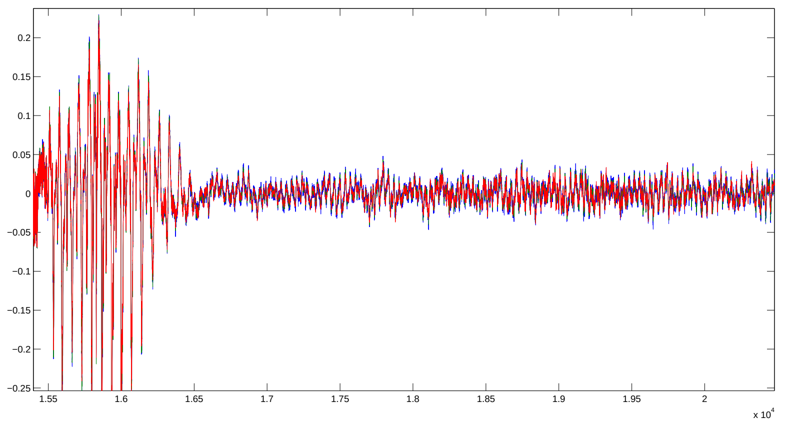


Figure 6: The previous sound byte, but with a compression ratio of 2 to 1.

If we compare these two, previous figures, we conclude that there is little difference among them, eventhough they were compressed at very different ratios. Recalling the difference in the reconstructed signals from the first signal we looked at, we must conclude that the nature of the second sound correlates to the Fourier Transform in a “nicer” way. This makes sense if we recall that the F.T. correlates to signals based on a sine wave. So, input signals that are smooth and continuous, like a sine wave, will work best with the Fourier Transform. Now we will analyze how the Fourier Transform reacts with a bird-song. If we think about the nature of a bird song, we might be able to predict how well this method will work. Bird songs are somewhat erratic with sharp starts, stops, and quick changes in frequencies. This seems like a signal that would introduce many artifacts into the reconstructed signal.

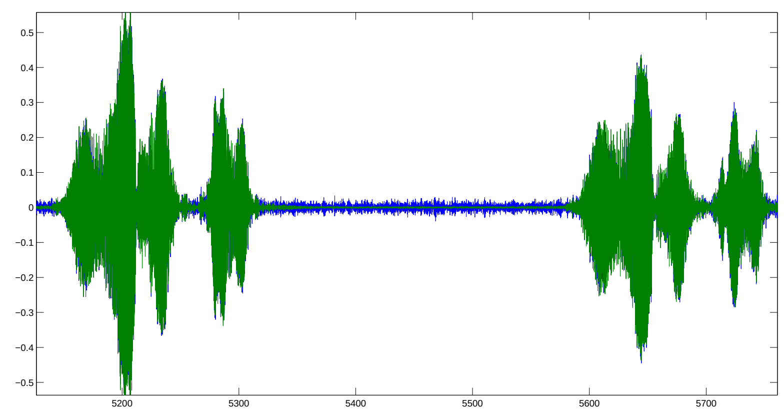


Figure 7: A time domain representation of the song of an American Robin. Green depicts the original signal.

We can see in this figure, that the parts of the signal with information have high integrity in regards to the reconstructed signal being very similar to the original signal.

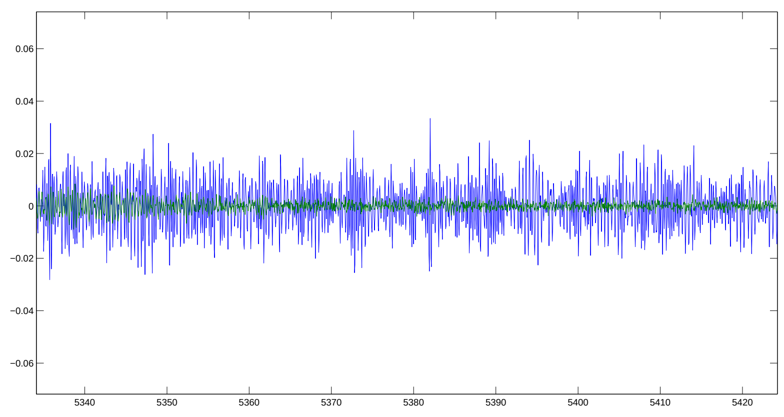


Figure 8: A magnification of the silent portion of Figure 7.

Here, we see that the Fourier Transform introduces many artifacts to the reconstructed signal, here represented by blue. The artifacts are so significant here is because of the sharp changes in the signal on either side of the silence; that is, the sine wave correlates poorly with the spikes in amplitude of frequency. Now that we know some of the characteristics of signals that behave poorly with the Fourier Transform, let's think of a type of signal that will work very well with this type of compression. Since breaks in noise is bad, let us recognize that we need a continuous noise in our signal. Secondly, sharp changes in amplitude of frequency is also bad; so, if we choose a signal that is in a close range of frequencies, within an octave, and has consistent volume, we can predict that the reconstructed signal will have high integrity with the original.

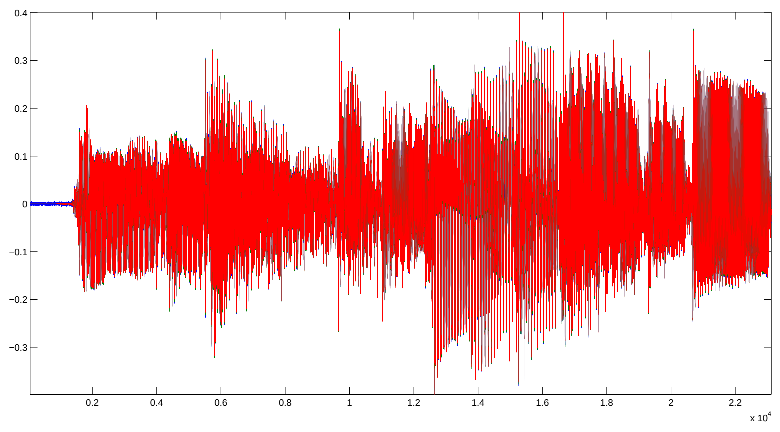


Figure 9: The introduction of “Californication” by the Red Hot Chili Peppers in the time domain.

We can see in the figure that with a compression ratio of 9 to 1, which is a very rigorous ratio (only 10 percent of the original data is kept), that very little artifact is introduced. If one listens to this difference, the reconstructed signal will have a slightly “hollow” sound, on account of so much information being thrown away; however, the quality is remarkable for such a compression ratio. This result is directly correlated with the behaviour of the original signal. The figures used in this section are products from my Matlab code.

4.2 Image Compression

Analyzing image files is a little different than audio files, mainly because these files are visual rather than audible. For example, before, we discussed how the compression of audio files created a “hollow” sound in the reconstructed signal; that is to say, it took a sharpness away from the original file. This same type of effect happens with picture files. The figures below show the difference between the original file and the reconstructed file.

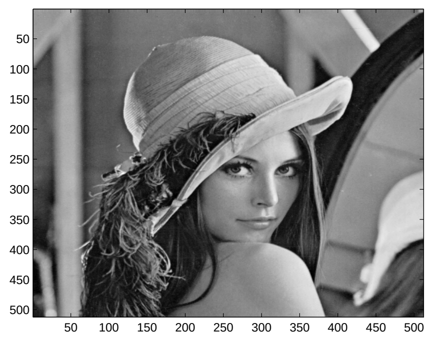


Figure 10: This is the original image of the famous image compression model “Lenna”[8] .

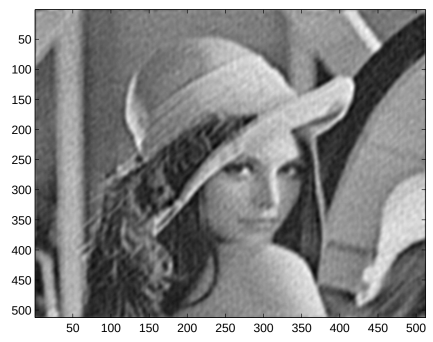


Figure 11: The reconstructed "Lenna" from Figure 10 with a threshold of 80 percent.

We see that the sharpness and definition of the lines in the picture begin to blur when the decompression takes place. In previous sections, we also discussed the difference between the original file and the reconstructed file; or, more specifically, $\|\mathbf{s} - \mathbf{s}_0\|$. When analyzing image compression, we are able to create a figure of this very difference.

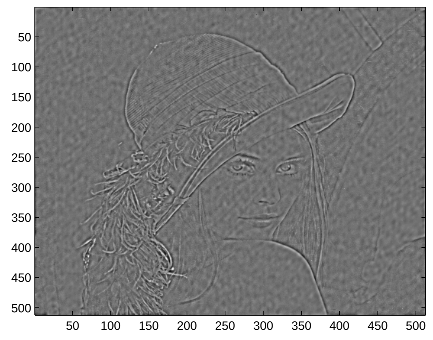


Figure 12: This is the space domain representation of the difference between the images from Figure 11 and Figure 10

Artifacts are slightly different when present in the image files. In order to create a better visualization of these artifacts, we will construct the image below.

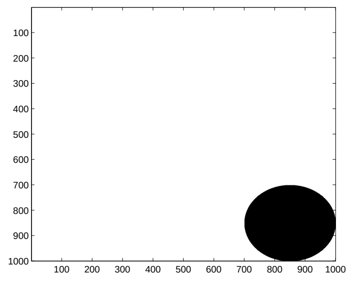


Figure 13: The image was created in Matlab and depicts a black circle on a white background. The parameters of the image are able to be changed by the user.

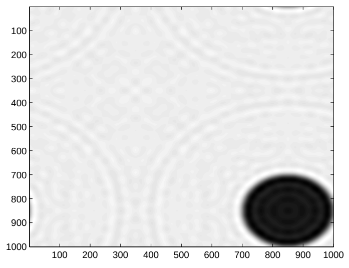


Figure 14: The image of the reconstructed image. In this compression, 90 percent of the data was kept.

In the reconstruced picture, we are able to see a pattern that was not present in the original file. In this two-dimensional compression, artifacts create a type of “ripple” effect. Much like in the audio files, where artifacts are introduced at the “edges” of the data in the space domain, artifacts in picture files are created where there are hard-lines with a sharp contrast of color. In a black and white case, artifacts are very significant because white is represented by the number 0, and black is represented by the number 1. So, when the Fourier Transform is taken of these neighboring 0’s and 1’s, artifacts are introduced. If we recall the method of thresholding in the compression process, we remember that the unwanted data is set to the value of 0; or, in this case, the color white. Therefore, when the thresholding is more aggresive, we receive more artifacts, since there are more data points that contrast with their neighbors in the matrix.

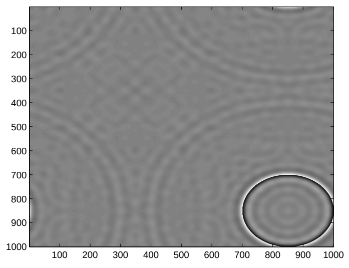
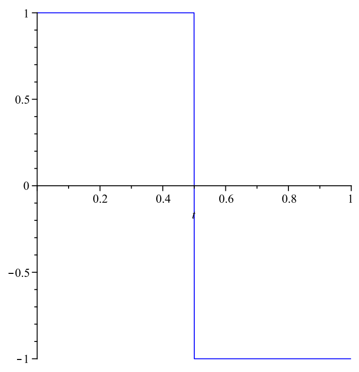


Figure 15: This figure is the space domain representation of the difference between the original cirlce file, and the compressed file.

Here, we can easily see that the largest difference in the data is around the edge of the circle. This is a direct consequence of the artifact phenomenon.

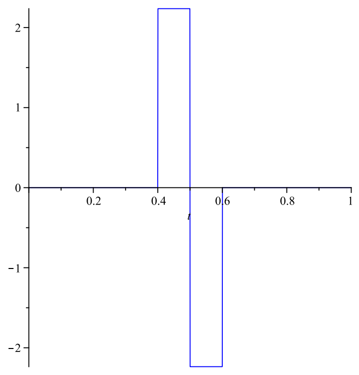
5 Wavelet Method

A wavelet is a single oscillation of a signal that is used to corrolate to information in signal processing. There are many types of wavelets; however, the Haar wavelet was used in this project. The original wavelet is called the *Mother Wavelet.* This Mother Wavelet can be shifted and scaled in order to more closeley correlate with the input signal. The piece-wise function of the Haar Wavelet is depicted in the following figure.



$$\Psi(t) = \begin{cases} 1 \text{ if } t \in [0, 1/2) \\ -1 \text{ if } t \in [1/2, 1) \\ 0 \text{ otherwise} \end{cases}$$

The idea of wavelet analysis is to correlate a signal against shifted and scaled versions of the Mother wavelet. These so-called *daughter wavelets* are of the form



$$\Psi_{a,b}(t) = \frac{1}{\sqrt{a}}\Psi\left(\frac{t-b}{a}\right)$$

The parameter a is the *scale factor*, and b is the *shift factor*. The factor $\frac{1}{\sqrt{a}}$ is a normalizing constant that insures that

$$\int_{-\infty}^{\infty} |\Psi_{a,b}(t)|^2 \, dt = 1,$$

which means that all daughter wavelets will have the same energy.

We will now explore the *continuous wavelet transform* of s. Let s be a signal defined on $[0, 1)$. The continuous wavelet transfrom of s is the function $C(a, b)$, which is called the Haar Wavelet Transform, is defined by

$$C(a,b) = \int_1^0 s(t)\,\Psi_{a,b}(t)\,dt$$

Theorem 6. *If $\Psi_{a,b}$ is the normalized Haar wavelet, then*

$$C(a,b) = \frac{1}{\sqrt{a}}\left(\int_{b+\frac{a}{2}}^{b} s(t)\,dt - \int_{b+a}^{b+\frac{a}{2}} s(t)\,dt\right)$$

Attached to this paper is code from Maple that shows the Haar wavelet transform for various signals.

Now we must ask ourselves what the Haar Wavelet Transform produces. Depending on our choice of (a, b), the Transform will yield various answers; fortunately, we can deduce what these values mean. If (a, b) produces a relatively large wavelet coefficient, $C(a, b)$, then that means that there is a high correlation between the daughter wavelet, $\Psi_{a,b}(t)$, and the original signal $s(t)$. On the other hand, if the wavelet coefficient is relatively small, then the daughter wavelet has a low correlation with $s(t)$.

Let us first fix $k \geq 0$ for all k in $\mathbb{Z}^+$, and consider a set, A_k of daughter Haar wavelets given by

$$A_k = \left\{\Psi_{a,b}(t) : a = 2^{-k}, b = j2^{-k},\, 0 \leq j \leq 2^k - 1\right\}$$

In this figure, we have depicted the first three *levels* of our Harr bases. Observing this figure, we note that no two parts of A_k overlap; in other words, their supports (the vertical lines), belong to only one of each of the daughter wavelets. Now, we denote our bases by the function

$$\beta_n = \bigcup_{k=0}^{n-1} A_k$$

Theorem 7. *β_n is an orthonormal set with respect to the inner product defined by*

$$(f, g) = \int_{-\infty}^{\infty} f(t) \cdot g(t)\, dt$$

Proof. It is straightforward that β_n is orthonormal from the equation

$$\int_0^1 \Psi_{k_1,j_1}(t)\, \Psi_{k_2,j_2}(t) = \begin{cases} 0 \text{ if } k_1 \neq k_2 \text{ or } j_1 \neq j_2 \\ 1 \text{ if } k_1 = k_2 \text{ and } j_1 = j_2 \end{cases}$$

$\square$

We remark that for any continuous function $s : [0, 1) \in \mathbb{R}$, it can be shown that there exists a sequence of functions s_n such that

1. $s_n \in \operatorname{span}(\beta_n)$
2. $\|s - s_n\|^2 \to 0$ as $n \to \infty$

It then follows from these two properties that

$$\beta = \bigcup_{n=0}^{\infty} \beta_n$$

forms a *basis* for all continuous functions that are defined on $[0, 1)$. It should also be noted that

$$|\beta_n| = 1 + 2 + 2^2 + \cdots + 2^{n-1} = 2^n - 1\ ,$$

which implies that the span of β_n is a vector space of dimension $2^n - 1$.

Studying the projections of arbitrary signals onto these subspaces is the essence of wavelet analysis. Since β_n is an orthonormal basis for the span(β_n), the orthogonal projection of a signal, $s(t)$, onto span(β_n) $= W_n$. This is given by

$$s_{W_n}(t) = \operatorname{proj}_{W_n}(s(t)) = \sum_{k=0}^{n-1} \sum_{j=0}^{2^k-1} c_{k,j} \Psi_{k,j}(t)\,,$$

when $c_{k,j}$ is defined as

$$c_{k,j} = \int_0^1 s(t)\, \Psi_{k,j}(t)\, dt$$

The process of the computation of the projection of $s(t)$ onto W_n of various signals can be viewed on the attached pages at the end of the paper.

5.1 Results

The methods of analyzing the results from wavelet compression are similar to those of studying the Fourier Transform results. First, let us start by simply viewing the original file and decompressed file in the time domain.

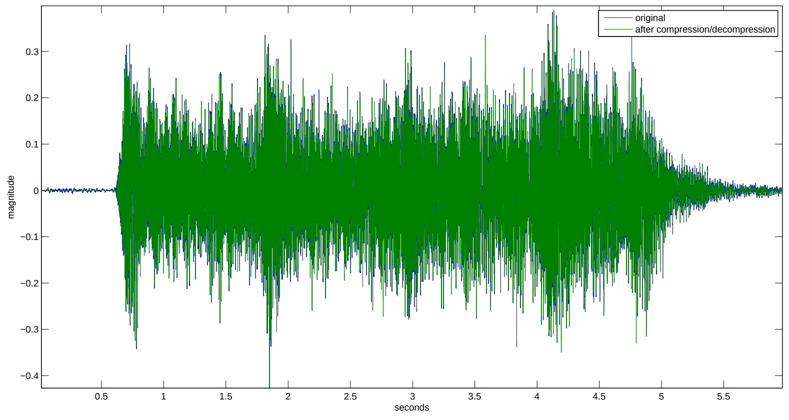


Figure 16: This is the time domain representation of the first six seconds of Vivaldi's "Concerto in G Minor for Violins". The blue represents the original file and the green depicts the compressed/decompressed file. It was compressed at a ratio of 4 to 1.

By observation, the two signals look to be very similar. Now we will look at a few magnifications of the previous figure.

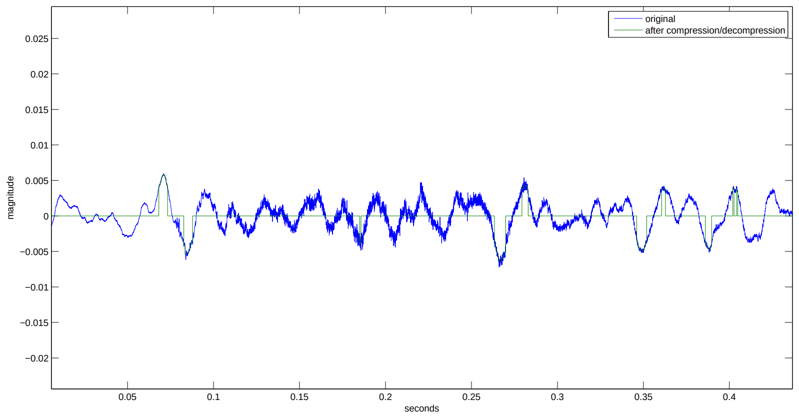


Figure 17: This is a segment from the first half second of Figure 16, which is silent.

As the reader can see, this method of compression seems to be tailor-made for silent clips of audio. There are a few oscillations that have a relatively large amplitude, which the daughter wavelet attempts to correlate with; however, on the whole, the Haar wavelet recognizes that the signal has almost zero information and matches that observation. In a way, this result produces an effect that is even better than the original file. In most cases, when silence is recorded, background noise is unwanted; however, there is always some noise present. With this method of compression, that noise is completely squelched and pure silence is obtained.

If we magnify a different part of the signal, where information is present, we obtain the following figure.

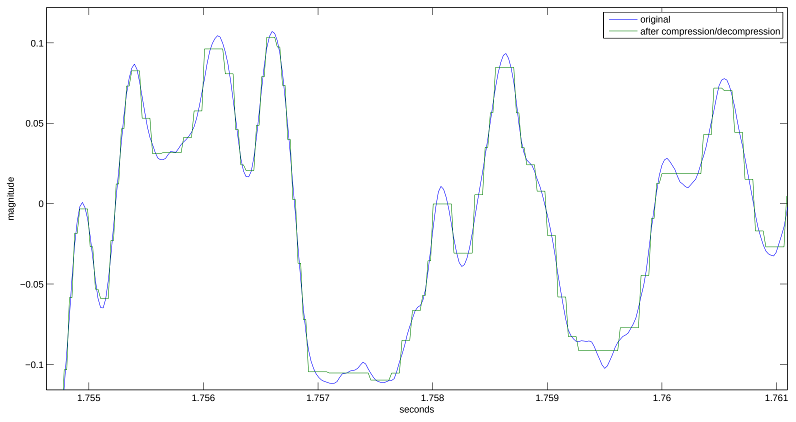


Figure 18: This is a magnification of a very small section of the sound file. The green depicts the attempt of the daughter wavelets to correlate with the original signal, which is in blue.

While there are many silimarlities between the original and decompressed file, there are obvious discrepancies between the two. This is a part of the original section that wavelets do not correlate well with. That is to say, the smooth, continuous part of a signal. The nature of the Haar wavelet is one that is sharp; and, while having more daughter wavelets helps this weakness, artifacts are created from the difference in signals. The "sharp" endges of the wavelet introduce high frequencies into the recreated signal. The following figure depicts the introduction of these high frequencies.

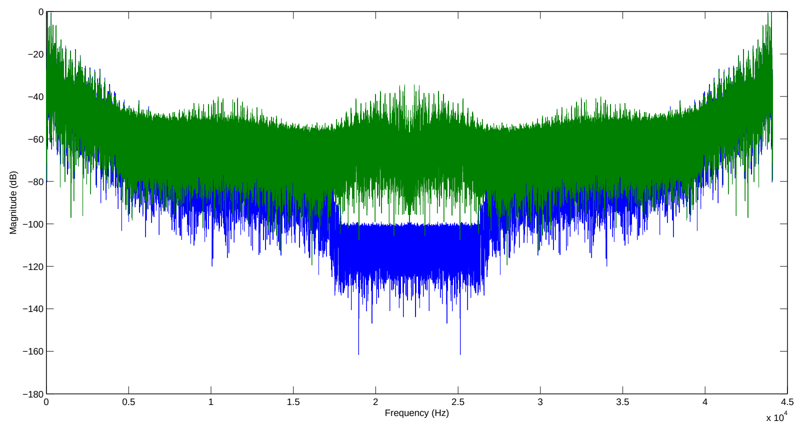


Figure 19: The frequency domain representation of a sound file. The blue segments depict the reconstructed signal. The reader should also note that this figure has a symmetry of its frequencies that begin reflecting at about 2,250 hertz.

As you can see from the figure, the reconstructed segments have larger magnitudes at frequencies from 1800 to 2600 hertz. These are the frequencies that are introduced through the compression process. Much like with the Fourier transform, we can deduce certain types of signals that have potential to work better with the wavelet compression. We want a signal that has a "sharp", discontinuous nature. A bird song is a good example of such a signal.

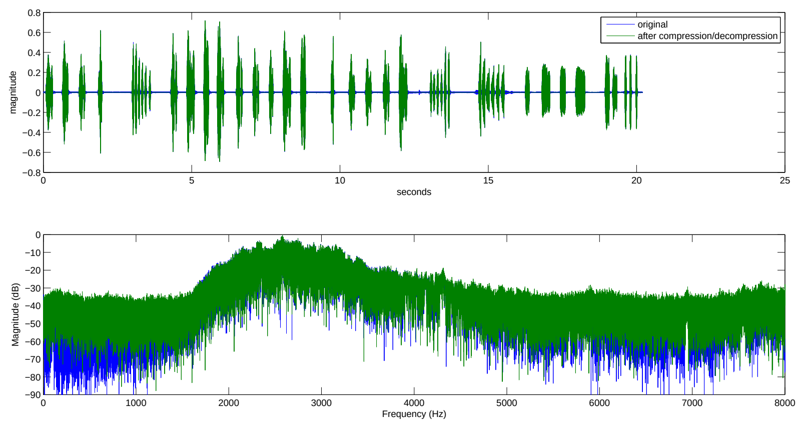


Figure 20: The top half of this figure depicts the time domain representation of a song of the American Robin; the bottom half shows the frequency domain.

Viewing the frequency domain, we observe that the variation from the original signal is not very significant. This is because the wavelets correlate well with the sharp variations of frequency and variation in the bird song.

6 Conclusion

In this paper, we have discussed various methods of data compression from the mathematical background up to the code that actually compresses the files. Through this process, we have also seen many signals and discussed behaviors of signals that correlate better with some methods than others. For example, a bird chirp correlates extremely well by the Haar wavelet compression, but it does not work well with the Fourier Transform. This result arises from the basic nature of the Haar wavelets; they are discontinuous, so they correlate well with discontinuous signals. On the other hand, smooth, continuous signals work better with the Fourier Transform since the Fourier Transform correlates well with continuous signals.

Standard computer technology has built on the ideas we have discussed in this paper, adding new techniques, such as block coding, compressive sensing, and others. [7] These technologies have evolved and become an ubiquitous part of information technology.

7 References

1. Wolfram, Stephen. "History [of data compression]". *Some Historical Notes.* Wolfram Media, 2002. http://www.wolframscience.com/reference/notes/1069b. 30 March, 2010.

2. Wolfram, Stephen. "History [of irreversible data compression]". *Some Historical Notes.* Wolfram Media, 2002. http://www.wolframscience.com/reference/notes/1072d. 30 March, 2010.

3. Wikipedia contributors. "Wavelet." Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 30 Mar. 2010. http://en.wikipedia.org/wiki/Wavelet#History. 30 Mar. 2010.

4. Wikipedia contributors. "Fractal compression." Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 30 Mar. 2010. http://en.wikipedia.org/wiki/Fractal_compression#History. 30 Mar. 2010.

5. Wikipedia contributors. "Compressed sensing." Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 30 Mar. 2010. http://en.wikipedia.org/wiki/Compressive_sensing. 30 Mar. 2010.

6. Wikipedia contributors. "NyquistShannon sampling theorem." Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 30 Mar. 2010. http://en.wikipedia.org/wiki/Nyquist-Shannon_sampling_theorem. 30 Mar. 2010.

7. Madisetti, Vijay K. Douglas B. Williams. The Digital Signal Processing Handbook. CRC Press: Boca Raton, 1997.

8. "Lenna" image retrieved from Google Images, September 2009.

9. Pendergrass, M., Personal Communication, September 2009 through March 2010.

10. Pendergrass, M., Lecture Notes for Applied Mathematics (Math 345), 2008,http://people.hsc.edu/faculty-staff/mpendergrass/Dr._Marcus_Pendergrass/Applied_Math.html. 30 Mar. 2010.

8 Appendices: Matlab Code

The following appendix contains Matlab code implementing the Fourier and wavelet compression and decompression algorithms discussed in this paper.