COMPRESSING IMAGES WITH WAVELETS

JAMES SUMNERS

Abstract. In this article we examine two methods of image compression. We compare the Discrete Cosine Tranform II, the basis of the JPEG image format, with the Discrete Wavelet Transform. Specifically, we examine how the Haar Wavelet can be used to compress images, and how it compares with JPEG compression. The main focus of the article is the Discrete Wavelet Transform and the theory behind it.

1. Introduction

On a daily basis, we deal with large amounts of digital information. Text, audio, video, and images are just a few examples of the types of "data" we encounter. All of these things require storage space. Compression enables us to store large amounts of data in a smaller space.

Each type of data (text, audio, etc.) usually has one, or multiple, algorithms that work particularly well for that specific kind of data. There are also some general algorithms that work on almost any type of data. Also, each algorithm can be classified as "lossless" or "lossy." Some methods can be both lossy and lossless. A lossless compression algorithm retains all of the original meaning so that it can be fully restored from the compressed data. A lossy compression algorithm "loses" some of the original information, but retains enough so that the original meaning can be recovered.

The most popular image format in use today is the Joint Photographic Experts Group format – JPEG. This format is based on the Discrete Cosine Transform II, commonly known as the DCT. The DCT can be used to compress multiple types of data. In addition to JPEG pictures, the DCT, in modified form, can be found on iPods all over the world. It is used to compress the music found on these devices [Liu and Lee(1999), sec. 0]. No matter how it is used, the DCT is a lossy compression scheme.[1]

The Discrete Wavelet Transform (DWT) can also be used for multiple types of data [Graps(1995), sec. 4.2]. Unlike the DCT, though, the DWT can be a lossless compression scheme [Mulcahy(1996), sec. 5].

Thank you to Dr. David B. Williams for his patience and advice.

[1]Actually, the full JPEG specification includes a lossless method. But the most widely used specification is the JPEG File Interchange Format, JFIF, which is strictly a lossy format. In this article JPEG refers to JFIF [CCI(1992), sec. 4.2].

2. DCT Compression

2.1. **Overview.** We will begin by examining the standard JPEG algorithm. It is the standard image format used all over the World Wide Web, and it is a lossy compression method. On almost every modern digital camera the default image format is JPEG. The format has become ubiquitous in today's digital world. But how does it work?

As mentioned in section 1, the JPEG image format is an implementation of the Discrete Cosine Transform. As defined in [Wallace(1991), sec. 4], the JPEG algorithm consists of three stages: the Foward Discrete Cosine Transform (FDCT), a quantization step, and an entropy encoding step. We will leave entropy encoding out of the calculations in this paper for the sake of simplicity.

Before we can apply the JPEG algorithm to an image, we must define what an image is and how the JPEG algorithm works. Consider a $2^m \times 2^m$ pixel grayscale image to be a $2^m \times 2^m$ matrix A, with entries whose values are in the range $[0, 255]$. Each entry in A is the level of gray in the image at pixel $A_{i,j}$. Given this relationship, we can apply the FDCT to encode the image and quantize the results to achieve some level of compression.

The FDCT is defined as [Wallace(1991), sec 4.1]:

$$W_{p,q} = \frac{1}{4}C(p)C(q)\left[\sum_{i=1}^{n}\sum_{j=1}^{n} A_{i,j}\cos\left(\frac{(2i+1)p\pi}{16}\right)\cos\left(\frac{(2j+1)q\pi}{16}\right)\right] \tag{1}$$

where

$$C(t) = \begin{cases} \frac{1}{\sqrt{2}}, & t = 1 \\ 1, & \text{otherwise} \end{cases}$$

and $n = 2^m$ is the dimension of the $2^m \times 2^m$ image matrix being encoded. Since the JPEG standard calls for the image being encoded to be broken into 8×8 blocks, we let $n = 8$ and encode each block of the image *independently* of others. If the dimensions of the matrix A are not a power of 2, the matrix can be padded with columns and/or rows of zeros to achieve the required dimension.

Of particular note here is the coefficient $A_{i,j}$, which is the image data at the i, j position of our uncompressed image matrix A. The result of the transform is a new value, $W_{p,q}$, for the transformed matrix W at the p, q position.

For example, consider the image matrix

$$A = \begin{bmatrix} 64 & 60 & 58 & 60 & 60 & 59 & 72 & 77 \\ 63 & 58 & 138 & 61 & 59 & 58 & 50 & 116 \\ 148 & 134 & 138 & 106 & 116 & 81 & 54 & 131 \\ 92 & 61 & 126 & 97 & 100 & 106 & 72 & 72 \\ 61 & 59 & 116 & 61 & 81 & 59 & 61 & 130 \\ 61 & 111 & 81 & 58 & 58 & 58 & 80 & 38 \\ 63 & 59 & 59 & 58 & 59 & 58 & 113 & 111 \\ 63 & 61 & 60 & 60 & 60 & 59 & 58 & 144 \end{bmatrix}. \tag{2}$$

Applying the FDCT gives the transformed matrix

$$W = \begin{bmatrix} 633.125 & -6.778 & 35.514 & -69.589 & 13.875 & -23.986 & 43.603 & 29.271 \\ 32.875 & 42.209 & -23.966 & -5.645 & 4.698 & -0.912 & 33.066 & 10.690 \\ -63.831 & -43.730 & 39.309 & -3.295 & 12.160 & 2.397 & -20.375 & -17.778 \\ -81.439 & -14.794 & -5.017 & 13.552 & -43.187 & 17.017 & -41.645 & 3.919 \\ -25.875 & -35.737 & -13.817 & 6.886 & 19.875 & 2.075 & 18.576 & -4.202 \\ 17.534 & 0.565 & -12.169 & 51.186 & -5.314 & 29.399 & -30.966 & -8.670 \\ 25.222 & 53.900 & 2.874 & 23.990 & 1.592 & -22.004 & -29.809 & -20.828 \\ 32.208 & -14.640 & 34.963 & -17.711 & 36.033 & -18.130 & -0.810 & 5.338 \end{bmatrix}. \tag{3}$$

If each nonzero entry takes one byte of storage space, we clearly have not compressed this image. There were only nonzero entries before the transformation, and there are only nonzero entries after the transformation. To achieve compression, we apply a quantization step. We do this by defining a quantization matrix Q, divide each entry in W by the corresponding entry in Q, and round to the nearest integer. That is, $Z_{i,j} = \text{round}(\frac{W_{i,j}}{Q_{i,j}})$, where Z is the newly compressed image matrix. The entries in Q will define the level of compression.

Continuing the example, if we let Q be defined as in [Wallace(1991), sec. 7.3], we have

$$Q = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix} \tag{4}$$

and

$$Z = \begin{bmatrix} 40 & -1 & 4 & -4 & 1 & -1 & 1 & 0 \\ 3 & 4 & -2 & 0 & 0 & 0 & 1 & 0 \\ -5 & -3 & 2 & 0 & 0 & 0 & 0 & 0 \\ -6 & -1 & 0 & 0 & -1 & 0 & -1 & 0 \\ -1 & -2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{5}$$

after quantization. Notice that we now have one relatively large entry in the top left of the matrix, several small entries, and many zero entries. The DCT with quantization leaves us a matrix with one overall average coefficient and sixty-three detail coefficients that are "near zero" [Wallace(1991), sec. 4.1]. Because of the rounding that was performed, the DCT with quantization transformed the data and eliminated some details from the image. Different quantization matrices will retain more, or fewer, details as desired by the users. By storing only the nonzero entries of Z, we can store an approximate representation of A in a smaller space.

By defining the inverse quantization as $T_{i,j} = Z_{i,j}Q_{i,j}$, and the Inverse Discrete Cosine Transform (IDCT) as

$$A_{i,j} = \frac{1}{4}\left[\sum_{i=1}^{n}\sum_{j=1}^{n} C(p)C(q)T_{p,q}\cos\left(\frac{(2i+1)p\pi}{16}\right)\cos\left(\frac{(2j+1)q\pi}{16}\right)\right] \tag{6}$$

we can decompress the image. We will call this decompressed image matrix B.

So if we first apply the inverse quantization, and then the IDCT, to our compressed image Z, we get

$$B = \begin{bmatrix} 73 & 105 & 101 & 69 & 64 & 69 & 36 & 31 \\ 89 & 114 & 152 & 118 & 76 & 88 & 50 & 87 \\ 167 & 172 & 199 & 164 & 88 & 83 & 9 & 90 \\ 139 & 154 & 151 & 136 & 116 & 100 & 31 & 58 \\ 13 & 82 & 91 & 72 & 73 & 46 & 82 & 51 \\ 32 & 108 & 76 & 39 & 45 & 11 & 60 & 18 \\ 28 & 28 & 6 & 18 & 52 & 56 & 51 & 40 \\ 27 & 9 & 26 & 32 & 11 & 21 & 21 & 62 \end{bmatrix}. \tag{7}$$

Clearly, $B \neq A$. But if we take the difference of A and B,

$$A - B = \begin{bmatrix} -9 & -45 & -43 & -9 & -4 & -10 & 36 & 46 \\ -26 & -56 & -14 & -57 & -17 & -30 & 0 & 29 \\ -19 & -38 & -61 & -58 & 28 & -2 & 45 & 41 \\ -47 & -93 & -25 & -39 & -16 & 6 & 41 & 14 \\ 48 & -23 & 25 & -11 & 8 & 13 & -21 & 79 \\ 29 & 3 & 5 & 19 & 13 & 47 & 20 & 20 \\ 35 & 31 & 53 & 40 & 7 & 2 & 62 & 71 \\ 36 & 52 & 34 & 28 & 49 & 38 & 37 & 82 \end{bmatrix} \tag{8}$$

we see that they are relatively the same. The restored matrix (7) is close enough to (2) that they should be visually similar when viewed as a grayscale image.

2.2. **Application.** Consider Figure 10 on page 18. We see in this figure a fractal image in its uncompressed original state as generated by **Mandelbrot on Cocoa** (Appendix A). That is, all of the data originaly generated to represent the image is intact.

Now consider Figure 11 on page 19. Notice that Figure 11 is not as clear as Figure 10. In particular, the region between the center fractal and the left side is not as well defined. The lack of detail in the restored image is a result of the DCT compression with a threshold value set to 20. This results in a *Mean-Squared Error (MSE)* equal to 95.97 between the original image and the restored image. We calculate the MSE by Equation (10), where m and n are the dimensions of the image matrix.

$$C = A - B \tag{9}$$

$$MSE = \frac{\sum_{j=1}^{m} \sum_{i=1}^{n} C_{i,j}^2}{m \cdot n} \tag{10}$$

The MSE gives a metric by which we can determine the visual quality of the restored image as compared to the original. A low MSE value indicates a visually accurate restored image. A higher MSE indicates a restored image with increasingly noticeable distortion.

If we define the *compression ratio (CR)* to be the ratio of nonzero entries in A to the number of nonzero entries in Z, we find that the DCT, applied to Figure 10 with threshold level 20, results in a compression ratio of $5.9 : 1$.

$$CR = \frac{262144}{44122} = 5.9 \tag{11}$$

Using Matlab's `spy` function, we can visualize the MSE and the CR. In Figure 1 on page 6 we see a `spy` plot of the original image matrix A.

The blue color represents nonzero entries in the matrix. Clearly, the original image has no nonzero entries. This is also true of the restored image matrix B. But if we take the difference of A and B, we have a new

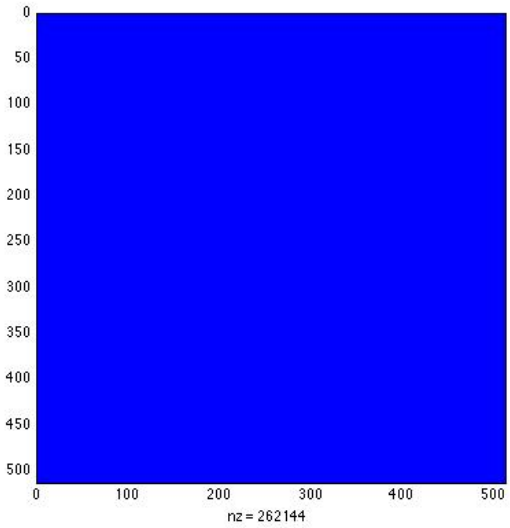


FIGURE 1. Spy plot of the original image matrix

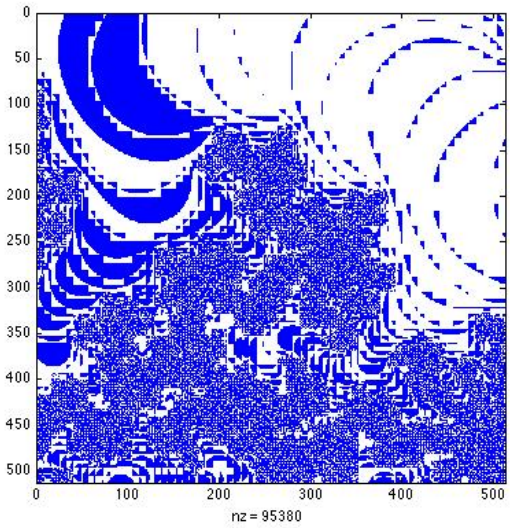


FIGURE 2. Spy plot of the MSE

matrix C that represents the error between A and B. The plot of which is in Figure 2.

The blue areas in Figure 2 represent the error between Figure 10 and Figure 11.

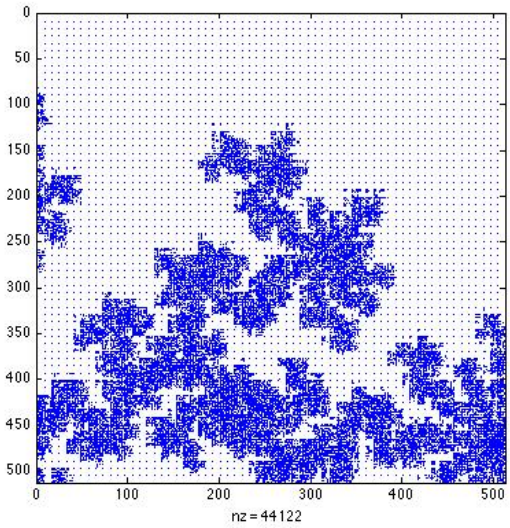


FIGURE 3. Spy plot of the CR

A visual representation of the compression ratio is the `spy` plot of the compressed image matrix Z. This plot is found in Figure 3.

The blue areas in Figure 3 represent the nonzero entries in Z. Clearly, there are far more zeros in Z than there are in A. Therefore, it takes far less space to store Z than it does A.

This example shows us how the DCT compression method affects the quality of the image. There is an inverse relationship between the compression level and the visual quality of the restored image. This is demonstrated by the MSE plot. If the MSE plot were solid white, there would be no difference between A and B.

We will see that this is true with the DWT as well, when thresholding is applied. What we want to find out is how the DWT visually compares to the DCT for approximately equal compression ratios.

3. DWT Compression

3.1. **Overview.** Just as the DCT is a finite series expansion, so is the Discrete Wavelet Transform. There is a key difference, though. Whereas the DCT relies on all values of the function being approximated, the DWT relies on only a few [Davidson and Donsig(2002)]. Therefore, the DWT has better local properties than the DCT. For example, they are local in frequency and time [Brani Vidakovic(1994), sec. 1]. This means that while the DCT has trouble at the edges of its 8×8 blocks [Strang(1999), sec. 7], the DWT has the potential to be more accurate in these areas.

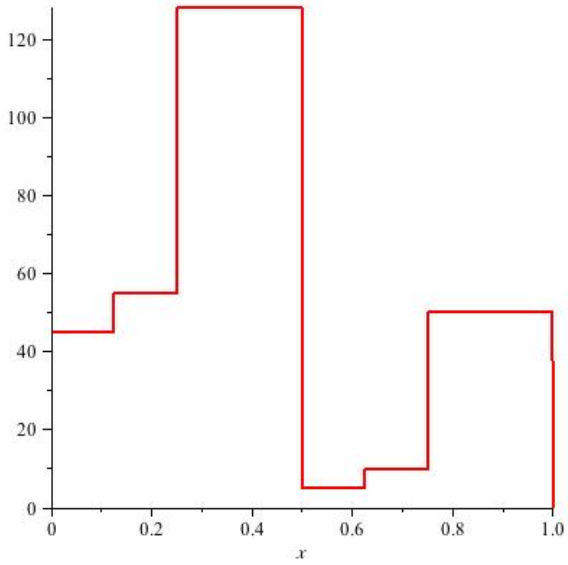


FIGURE 4. Plot of Eq. (13).

Consider

$$\begin{bmatrix}45 & 55 & 128 & 128 & 5 & 10 & 50 & 50\end{bmatrix} \tag{12}$$

as a eight pixel grayscale image. We can represent this image as a piecewise constant function

$$f(x) = \begin{cases} 45 & x \in [0, \frac{1}{8}) \\ 55 & x \in [\frac{1}{8}, \frac{1}{4}) \\ 128 & x \in [\frac{1}{4}, \frac{1}{2}) \\ 5 & x \in [\frac{1}{2}, \frac{5}{8}) \\ 10 & x \in [\frac{5}{8}, \frac{3}{4}) \\ 50 & x \in [\frac{3}{4}, 1). \end{cases} \tag{13}$$

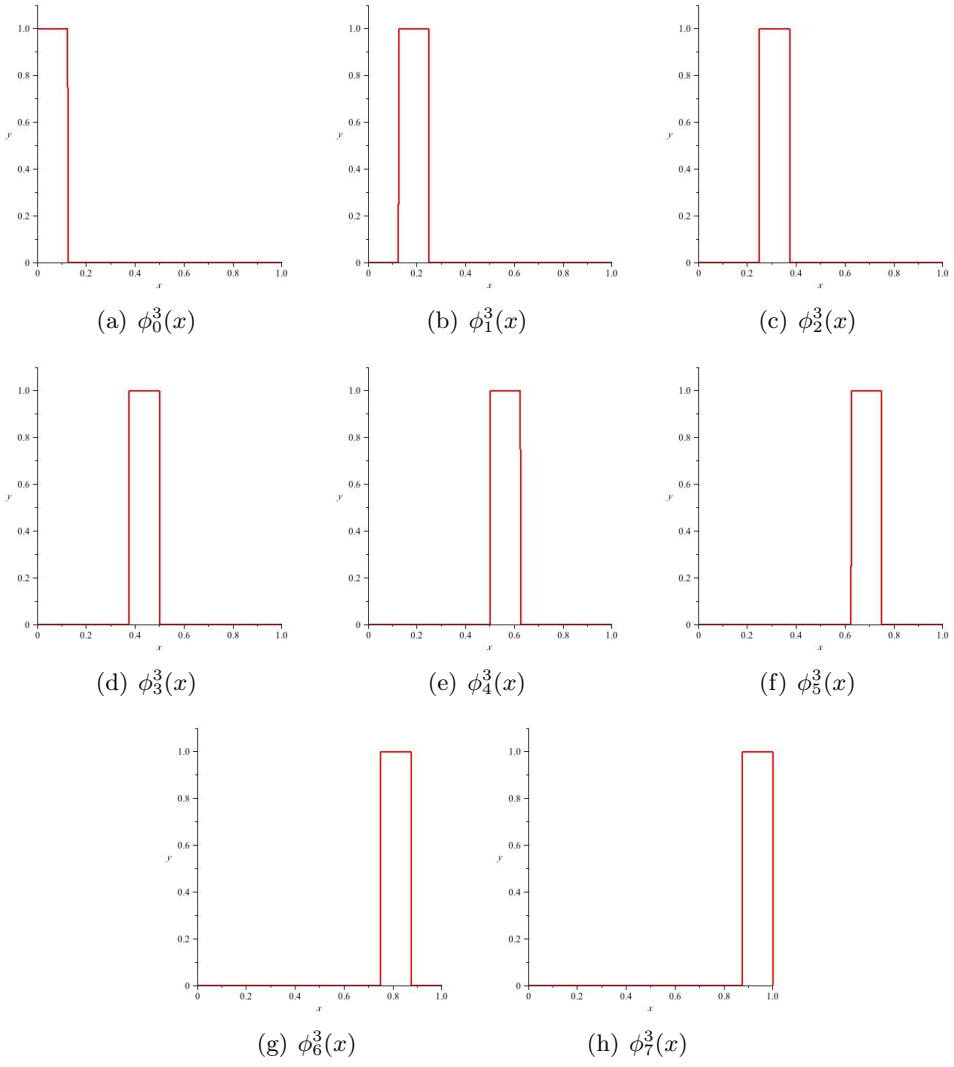
on the interval $[0, 1)$ and plot it as in Figure 4.

If we define the Haar scaling functions as

$$\phi(x) = \begin{cases} 1, & x \in [0, 1) \\ 0, & \text{otherwise} \end{cases} \tag{14}$$

$$\phi_k^j(x) = \phi(2^j x - k), \text{ where } j \in \mathbb{N} \text{ and } k = 0, 1, \ldots (2^j - 1) \tag{15}$$

and $\mathcal{V}^3 = \text{span}(\{\phi_0^3, \phi_1^3, \ldots, \phi_7^3\})$, we can represent this image as a sum of the $\phi_k^3(x)$. This is possible because the set of functions $\{\phi_k^3(x)\}$ form a basis for a function space $\mathcal{V}^3$ on the interval $[0, 1)$. To see this, define the inner



(a) $\phi_0^3(x)$ (b) $\phi_1^3(x)$ (c) $\phi_2^3(x)$

(d) $\phi_3^3(x)$ (e) $\phi_4^3(x)$ (f) $\phi_5^3(x)$

(g) $\phi_6^3(x)$ (h) $\phi_7^3(x)$

FIGURE 5. $\mathcal{V}^3$ basis functions.

product of two functions to be:

$$< f(x)|g(x) >= \int_{-\infty}^{\infty} f(x)g(x)\,dx. \tag{16}$$

If $< f(x)|g(x) >= 0$, then the two functions are orthogonal. Otherwise, they are not orthogonal [Eric J. Stollnitz(1995), sec. 3].

We can see from Figure 5 that any two of the functions are orthogonal to each other. For example $\phi_0^3(x)$ and $\phi_1^3(x)$ are plotted in Figure 5 (a) and Figure 5 (b), respectively. Everywhere $\phi_0^3(x)$ is nonzero, $\phi_1^3(x)$ is zero. So $\phi_0^3(x)$ and $\phi_1^3(x)$ are nonzero on disjoint intervals and thus $< \phi_0^3(x)|\phi_1^3(x) >=$

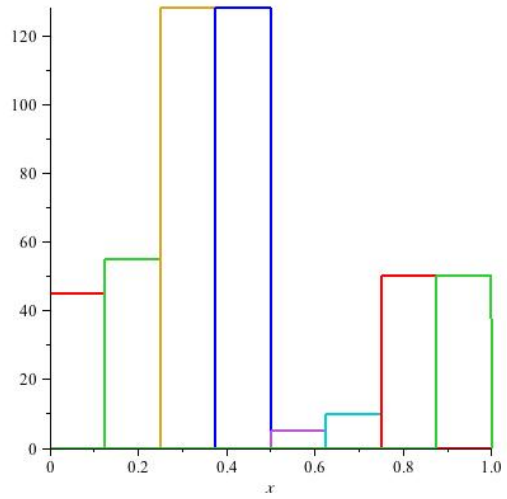


FIGURE 6. Basis functions for the 8 pixel image.

0. However, clearly $< \phi_0^3(x)|\phi_0^3(x) > \neq 0$. Since the $\phi_k^3(x)$ span $\mathcal{V}^3$ and are mutually orthogonal, they necessarily form a basis for $\mathcal{V}^3$.

Returning to our eight pixel image (12), we can plot the individual pixels by associating the color value of each pixel with each of the eight basis functions of $\mathcal{V}^3$. Then we have the functions:

$$\begin{array}{lll} 45\phi_0^3(x) & 55\phi_1^3(x) & 128\phi_2^3(x) \\ 128\phi_3^3(x) & 5\phi_4^3(x) & 10\phi_5^3(x) \\ 50\phi_6^3(x) & 50\phi_7^3(x). & \end{array}$$

Plotting all eight functions on the same plot gives us Figure 6. We can easily see how each function contributes to the overall plot if we plot the function,

$$f(x) = 45\phi_0^3(x)+55\phi_1^3(x)+128\phi_2^3(x)+128\phi_3^3(x)+5\phi_4^3(x)+10\phi_5^3(x)+50\phi_6^3(x)+50\phi_7^3(x).$$

The plot is exactly the same as the plot in Figure 4 on page 8. Thus, we have a way of representing an image as a sum of dyadic scaling functions.

3.2. **Averaging and Differencing.** The Haar scaling function (14) allows us to represent an image as a piecewise constant function on an interval, but it does nothing to compress the image. This is where the method of *averaging and differencing* comes into play.

Given a set of values, such as the eight values in our eight pixel image (12), we first pair the values. Then we average the pairs, and take the difference of the first value in the pair and the average. Thus the original set can be represented by a set of averages and a set of detail coefficients [Mulcahy(1996), sec. 3].

So for our eight pixel image, we would first pair 45 with 55, 128 with 128, 5 with 10, and 50 with 50. We would then average these pairs to get the set

$$\begin{bmatrix} 50 & 128 & 7.5 & 50 \end{bmatrix}. \tag{17}$$

Then we subtract each element in (17) from the first element in each pair to get

$$\begin{bmatrix} -5 & 0 & -2.5 & 0 \end{bmatrix}. \tag{18}$$

Thus we have transformed the image from (12) into

$$\begin{bmatrix} 50 & 128 & 7.5 & 50 & -5 & 0 & -2.5 & 0 \end{bmatrix}. \tag{19}$$

The blue averages are merely an average of two adjacent pixels. So their plot, as seen in Figure 7 on page 12, is roughly the same as the original plot in Figure 4 on page 8.

Recall that we used a function in the function space $\mathcal{V}^3$ to plot our original image. Here, we have a function

$$v(x) = 50\phi_0^2(x) + 128\phi_1^2(x) + 7.5\phi_2^2(x) + 50\phi_3^2(x) \tag{20}$$

that is in the function space $\mathcal{V}^2 = span(\{\phi_0^2, \phi_1^2, \phi_2^2, \phi_3^2\})$. As a result, we have clearly lost some detail in the image. So we need some way to represent the missing detail in this lower function space. To do so, we will define the *Haar mother wavelet* [Mulcahy(1996), sec. 4] as

$$\psi(x) = \begin{cases} 1, & x \in [0, \frac{1}{2}) \\ -1, & x \in [\frac{1}{2}, 1) \\ 0, & \text{otherwise} \end{cases} \tag{21}$$

$$\psi_k^j(x) = \psi(2^j x - k), \text{ where } j \in \mathbb{N} \text{ and } k = 0, 1, \ldots (2^j - 1). \tag{22}$$

We can then define a function space $\mathcal{W}^2 = \text{span}(\{\psi_0^2, \psi_1^2, \psi_2^2, \psi_3^2\})$. By the same argument we used to support the ϕ_k^3s as a basis for the function space $\mathcal{V}^3$, we can justify the ψ_k^2s as a basis for this new function space. If we let the coefficients of $\psi_k^2(x)$ be the detail coefficients, or *wavelet coefficients*, we have the function:

$$w(x) = -5\psi_0^2(x) - 2.5\psi_2^2(x). \tag{23}$$

The plot of which can be seen in Figure 8 on page 12.

Now if we plot

$$v(x) + w(x) = 50\phi_0^2(x) + 128\phi_1^2(x) + 7.5\phi_2^2(x) + 50\phi_3^2(x) - 5\psi_0^2(x) - 2.5\psi_2^2(x) \tag{24}$$

we will get the same plot as in Figure 4.

Notice in Equation 19 that after the first transformation we have two zero entries in the image. Assuming zero entries require no storage space as we did in Section 2.1, we have already eliminated two bytes from this image. Not only have we saved two bytes of storage space, but we have not lost any information. The original image (12) can be perfectly reconstructed from

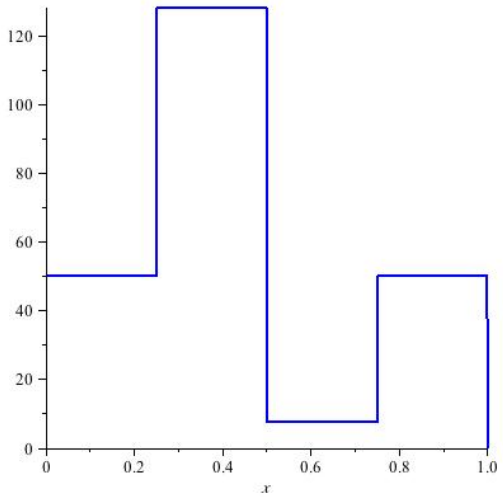


FIGURE 7. Four averages from the eight pixel image.

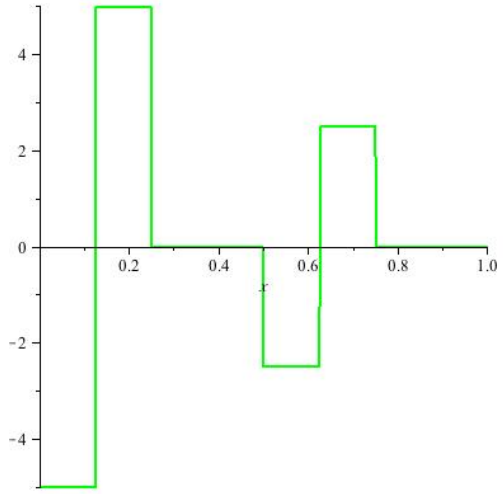


FIGURE 8. Four details from the eight pixel image.

(19). Simply reverse the steps performed in the transformation. And, as we have seen, we can reconstruct the original image from the compressed state by plotting Equation 24.

We can continue transforming (12) until we have only one average and seven detail coefficients. After performing averaging and differencing once to obtain (19), we repeat the process. Only this time, we average and difference only the first four entries. This will retain the original details and give us two additional detail coefficients. We would then have

$$\begin{bmatrix} 89 & 28.75 & -39 & -21.25 & -5 & 0 & -2.5 & 0 \end{bmatrix}. \tag{25}$$

One final application of averaging and differencing gives

$$\begin{bmatrix}58.875 & 30.125 & -39 & -21.25 & -5 & 0 & -2.5 & 0\end{bmatrix}. \tag{26}$$

We have still only managed two bytes of compression after the second and third transformations. But if we set all wavelet coefficients whose absolute values are less than some threshold, say 10, equal to zero, we get

$$\begin{bmatrix}58.875 & 30.125 & -39 & -21.25 & 0 & 0 & 0 & 0\end{bmatrix}. \tag{27}$$

and two extra bytes of compression. However, we have now lost some of the detail in the original image. Therefore we cannot perfectly reconstruct the image; we have introduced lossy compression. If we reconstruct this lossily compressed image, we get

$$\begin{bmatrix}50 & 50 & 128 & 128 & 7.5 & 7.5 & 50 & 50\end{bmatrix} \tag{28}$$

where the red entries are different from the originals in (12). Clearly this is not the same image, but it is a very close approximation. Each wrong entry only differs from the original by, at the worst, ± 5.

3.3. **Theory.** We have seen in the previous two sections how functions in a higher function space, say $\mathcal{V}^3$, can be represented by a sum of functions from other spaces ($\mathcal{V}^2$ and $\mathcal{W}^2$ were shown in the example). In this section we will attempt to show this fact in a more general way.

Recall that we defined $\mathcal{V}^3 = \text{span}(\{\phi_0^3, \phi_1^3, \ldots, \phi_7^3\})$. In more general terms, let $\mathcal{V}^j = \text{span}(\{\phi_0^j, \phi_1^j, \ldots, \phi_k^j\})$ where $k = 2^j - 1$. Then $\mathcal{V}^j$ is the set of all functions that are piecewise continuous on disjoint subintervals of $[0, 1)$ with length 2^{-j}. Let $j \in \mathbb{N}$ and $m, n = 0, 1, \ldots, k$. Then for $m \neq n$, $\phi_m^j(x)$ and $\phi_n^j(x)$ are nonzero on mutually disjoint intervals. Thus $< \phi_m^j | \phi_n^j >= 0$. If $m = n$ then

$$< \phi_m^j | \phi_n^j >= \int_0^1 \phi_m^j \phi_n^j \, dx = \int_0^1 \left[\phi_m^j\right]^2 \, dx = 2^{-j} \neq 0.$$

Thus the ϕ_m^j form an orthogonal set of functions. Since the ϕ_m^j are mutually orthogonal, and they span $\mathcal{V}^j$, they necessarily form an orthogonal basis for that function space.

We will similarly show that $\mathcal{W}^j$ is a function space with an orthogonal basis $\{\psi_k^j\}$. Now let $\mathcal{W}^j = \text{span}(\{\psi_0^j, \psi_1^j, \ldots, \psi_k^j\})$, where $j \in \mathbb{N}$ and $k = 0, 1, \ldots, 2^j - 1$ as before, and $m, n = 0, 1, \ldots, k$. If $m \neq n$, $< \psi_m^j | \psi_n^j >= 0$ since $\psi_m^j(x)$ and $\psi_n^j(x)$ are nonzero on mutually disjoint intervals. If $m = n$ then

$$< \psi_m^j | \psi_n^j >= \int_0^1 \left[\psi_m^j\right]^2 \, dx = 2^{-j} \neq 0.$$

Therefore the ψ_m^j also form an orthogonal set of functions. Since the ψ_m^j are mutually orthogonal and span $\mathcal{W}^j$, they form a basis for that function space.

Notice that we have a set of nested subspaces $\mathcal{V}^j \supset \mathcal{V}^{j-1} \supset \cdots \supset \mathcal{V}^1 \supset \mathcal{V}^0$, where each consecutive subspace contains fewer and fewer basis functions. To show that this is true, let $u \in \mathcal{V}^{j-1}$. Then, for some constants $c_0^{j-1}, c_1^{j-1}, \ldots, c_k^{j-1}$, u can be written as a sum of the basis functions:

$$u = c_0^{j-1}\phi_0^{j-1} + c_1^{j-1}\phi_1^{j-1} + \cdots + c_k^{j-1}\phi_k^{j-1},$$

where, as usual, $k = 0, 1, \ldots, 2^j - 1$. But for example, $\phi_0^2 = \phi_0^3 + \phi_1^3$, $\phi_1^2 = \phi_3^3 + \phi_4^3$, so in general, $\phi_k^{j-1} = \phi_{2k}^j + \phi_{2k+1}^j$. Thus we have

$$u = c_0^{j-1}\phi_0^j + c_0^{j-1}\phi_1^j + c_1^{j-1}\phi_2^j + c_1^{j-1}\phi_3^j + \cdots + c_k^{j-1}\phi_{2k}^j + c_k^{j-1}\phi_{2k+1}^j,\ k = (2^j - 1).$$

Thus $u \in \mathcal{V}^j$ which implies $\mathcal{V}^{j-1} \subset \mathcal{V}^j$. Clearly this holds true for any $j \in \mathbb{N}$.

Now that we have established the function spaces $\mathcal{V}^j$ and $\mathcal{W}^j$, and the nested nature of $\mathcal{V}^j$, we want to show that $\mathcal{V}^{j-1} \oplus \mathcal{W}^{j-1} = \mathcal{V}^j$. Let $f(x) \in \mathcal{V}^j$ for some $j \in \mathbb{N}$. Then, in terms of the basis functions ϕ_k^j, for $k = 0, 1, \ldots, 2^j - 1$,

$$f(x) = c_0^j\phi_0^0 + c_1^j\phi_0^1 + c_2^j\phi_1^1 + \cdots + c_k^j\phi_k^j$$

and some constants $c_0^j, c_1^j, c_2^j, \ldots, c_k^j$.

Using the method of averaging and differencing, we find

$$f(x) = a_0^{j-1}\phi_0^{j-1} + a_1^{j-1}\phi_1^{j-1} + \cdots + a_l\phi_l^{j-1} + d_0^{j-1}\psi_0^{j-1} + d_1^{j-1}\psi_1^{j-1} + \cdots + d_l^{j-1}\psi_l^{j-1}$$

where

$$a_l^{j-1} = \frac{c_{2l}^j + c_{2l+1}^j}{2} \text{ and } d_l^{j-1} = \frac{c_{2l}^j - c_{2l+1}^j}{2}$$

for $l = 0, 1, \ldots, (2^{j-1} - 1)$. Thus $f(x) \in \mathcal{V}^{j-1} \oplus \mathcal{W}^{j-1}$, which implies $\mathcal{V}^j \subseteq \mathcal{V}^{j-1} \oplus \mathcal{W}^{j-1}$.

Now let $f(x) \in \mathcal{V}^{j-1} \oplus \mathcal{W}^{j-1}$. From the previous part, we know that

$$f(x) = a_0^{j-1}\phi_0^{j-1} + a_1^{j-1}\phi_1^{j-1} + \cdots + a_l\phi_l^{j-1} + d_0^{j-1}\psi_0^{j-1} + d_1^{j-1}\psi_1^{j-1} + \cdots + d_l^{j-1}\psi_l^{j-1}$$

in terms of $\mathcal{V}^{j-1} \oplus \mathcal{W}^{j-1}$. By reversing the averaging and differencing, we have

$$c_k^j = a_l^{j-1} + d_l^{j-1} \text{ and } c_{k+1}^j = a_l^{j-1} - d_l^{j-1}$$

for $k = 0, 2, 4, \ldots, (2^j - 2)$ and $l = 0, 1, \ldots, (2^{j-1} - 1)$. Thus

$$f(x) = c_0^j\phi_0^0 + c_1^j\phi_0^1 + c_2^j\phi_1^1 + \cdots + c_k^j\phi_k^j \in \mathcal{V}^j$$

for $k = 2^j - 1$. Therefore $\mathcal{V}^{j-1} \oplus \mathcal{W}^{j-1} \subseteq \mathcal{V}^j$.

The result of this discussion is that we are able to write a function $\mathcal{V}^j$ as a direct sum of its nested subspaces and their corresponding wavelet function spaces $\mathcal{W}^j$. For example,

$$\mathcal{V}^3 = \mathcal{V}^2 \oplus \mathcal{W}^2 = \mathcal{V}^1 \oplus \mathcal{W}^1 \oplus \mathcal{W}^2 = \mathcal{V}^0 \oplus \mathcal{W}^0 \oplus \mathcal{W}^1 \oplus \mathcal{W}^2$$

and in general,

$$\mathcal{V}^j = \mathcal{V}^0 \oplus \mathcal{W}^0 \oplus \mathcal{W}^1 \oplus \cdots \oplus \mathcal{W}^{j-1}.$$

This is known as a Multiresolution analysis (MRA) [Mulcahy(1996), sec. 7].

A MRA allows us to analyze a signal at finer and finer details as j increases. When j increases by 1, the number of intervals on which the signal is analyzed is doubled. Also, each new interval is half the length of the previous intervals. This gives us the flexibility to eliminate unwanted deatils, details of such low value that they will have little impact on the reconstruction. By eliminating the details whose absolute value is less than some threshold we obtain a compressed signal.

Furthermore, if we redefine $\phi_k^j(x) = 2^{j/2}\phi(2^j x - k)$ and $\psi_k^j(x) = 2^{j/2}\psi(2^j x - k)$ we achieve normalization. That is, for $m, n = 0, 1, \ldots, 2^j - 1$, $< \phi_m^j | \phi_n^j >= 1$ and $< \psi_m^j | \psi_n^j >= 1$ when $m = n$. When $m \neq n$, $< \phi_m^j | \phi_n^j >= 0$ and $< \psi_m^j | \psi_n^j >= 0$. Therefore the ϕ_m^j and the ψ_m^j are orthonormal. We will continue using the previous definitions for the numerical examples in Section 3.4 to keep the calculations simple, but normalization gives better compression results [Eric J. Stollnitz(1995), sec. 6]. The program used to compress the images, **Wavelet** (Appendix A), uses the normalized functions.

3.4. **Haar Wavelet Application.** So far, we have only considered images to be a 2^j row vector. We can extend the Haar system defined in Section 3.3 to work with $2^j \times 2^j$ matrices by first performing the operations on the rows, and then the columns. So if we have the 4×4 image matrix

$$\begin{bmatrix} 10 & 20 & 30 & 40 \\ 20 & 40 & 60 & 80 \\ 30 & 60 & 90 & 120 \\ 40 & 80 & 120 & 160 \end{bmatrix}.$$

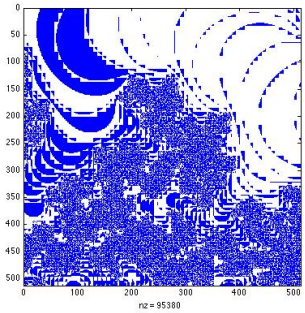
We can average and difference the rows until we get

$$\begin{bmatrix} 25 & -10 & -5 & -5 \\ 50 & -20 & -10 & -10 \\ 75 & -30 & -15 & -15 \\ 100 & -40 & -20 & -20 \end{bmatrix}$$

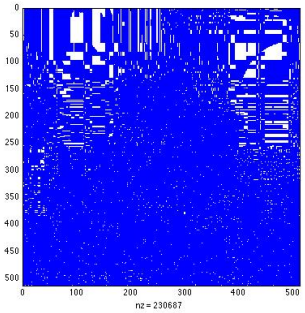
with the detail coefficients highlighted in blue and the averages in black. Finally, we transform the columns to get

$$\begin{bmatrix} 62.5 & -25 & -12.5 & -12.5 \\ -25 & 10 & 5 & 5 \\ -12.5 & 5 & 2.5 & 2.5 \\ -12.5 & 5 & 2.5 & 2.5 \end{bmatrix}.$$

Notice that, as before, we now have only one average. Again, this allows us to apply thresholding in such a fashion that we only alter the detail coefficients. Also of note is that we have not yet reduced the amount of storage space required to store this image. We have a $1:1$ transform of the original "image" to the transformed "image." Thus we can completely restore the original from the transformation. If we set the entries less than



(a) DCT Compressed Error Image



(b) Wavelet Compressed Error Image

FIGURE 9. A side-by-side comparison of the error image plots.

$|5|$ to 0 we achieve a possible four bytes of compression:

$$\begin{bmatrix} 62.5 & -25 & -12.5 & -12.5 \\ -25 & 10 & 5 & 5 \\ -12.5 & 5 & 0 & 0 \\ -12.5 & 5 & 0 & 0 \end{bmatrix}.$$

Reversing the averaging and differencing column-wise and then row-wise, we get an image that is approximately the same as the original:

$$\begin{bmatrix} 7.5 & 22.5 & 27.5 & 42.5 \\ 22.5 & 37.5 & 62.5 & 77.5 \\ 27.5 & 62.5 & 87.5 & 122.5 \\ 42.5 & 77.5 & 122.5 & 157.5 \end{bmatrix} \approx \begin{bmatrix} 10 & 20 & 30 & 40 \\ 20 & 40 & 60 & 80 \\ 30 & 60 & 90 & 120 \\ 40 & 80 & 120 & 160 \end{bmatrix}.$$

If we apply this process to our 512×512 pixel fractal image, and apply a threshold value of 48, we get the restored image as seen in Figure 12 on page 20. This image is clearly different from the original. The compression ratio for this image is 5.9 : 1, and it has a mean-squared error of 100.30. Recall that the CR for Figure 11 (page 19) was also 5.9 : 1, but its MSE was 95.97. If we look at the error plots for each image, as seen in Figure 9, we get a clear picture of how this small difference in the MSE affects the accuracy of the restored image.

We can easily see that the Haar wavelet, at an equivalent level of compression, alters the image much more than the DCT. But what happens if we increase the compression level?

If we set the threshold level to 70 for the DCT and to 150 for the Haar wavelet, we get a CR of approximately 41 and MSEs of 224.92 and 1286.51, respectively. The resulting images for the DCT compression can be seen in Figure 13 on page 21 and the wavelet compression in Figure 14 on page 22.

Notice that the DCT causes blurring at all points of the image, but the wavelet seems to keep some of the finer details clear. This leads to the conclusion that the wavelet method has more potential than the DCT method. The restored wavelet image may have more distortion in less active areas of the image than the DCT restored image, but the wavelet image retains more information in the active areas. So finer, possibly more important, details are retained at higher levels of compression with the DWT than with the DCT.

4. Conclusion

In the preceding sections, we define the Discrete Cosine Transform and how it is used in the JPEG picture compression standard. We then explored the theory behind the Discrete Wavelet Transform, specifically the Haar wavelet, and compared using it as an image compressor to the DCT.

We learned that the DCT does a good job of compressing images while retaining overall visual image quality. At comparable compression ratios, the DWT was less visually appealing but retained finer details. We concluded the discussion with an extreme comparison between the DCT and the DWT. In this extreme case, we learned that the DWT does a better job of retaining details at higher compression ratios than does the DCT.

Appendix A. Software Used

Several pieces of software were used in the development of this article. Namely:

- **Mandelbrot on Cocoa** [`http://hp.vector.co.jp/authors/VA026611/index_e.html`]: This excellent program was used to generate the fractal image seen throughout this article. The `mdl` file for the specific fractal location can be found on this article's web site.
- **The Gimp** [`http://www.gimp.org/`]: An open source photo editing application. This was used to convert the image output from *Mandelbrot on Cocoa* to TIFF from PNG format. It was also used to re-convert the TIFF images back into PNG images for inclusion in this article.
- **DCT Lab** [`http://www.sprljan.com/nikola/matlab/dctlab.html`]: An easy to use, open source, Matlab toolbox designed specifically for examining the error inherent in the JPEG file format. A modified version of this toolbox is available on this article's website. The modified version exports the compressed image Z to the current workspace so that the nonzero entries can be counted.
- **Maple 11** [`http://www.maplesoft.com/`]: Maple 11 was used to implement the FDCT and IDCT in 2.1. The worksheet is available on the website for this article.
- **Wavelet** [`http://student.claytonstate.net/~jsumners/wavelet/`]: Written specifically for this article. This software reads an eight bits

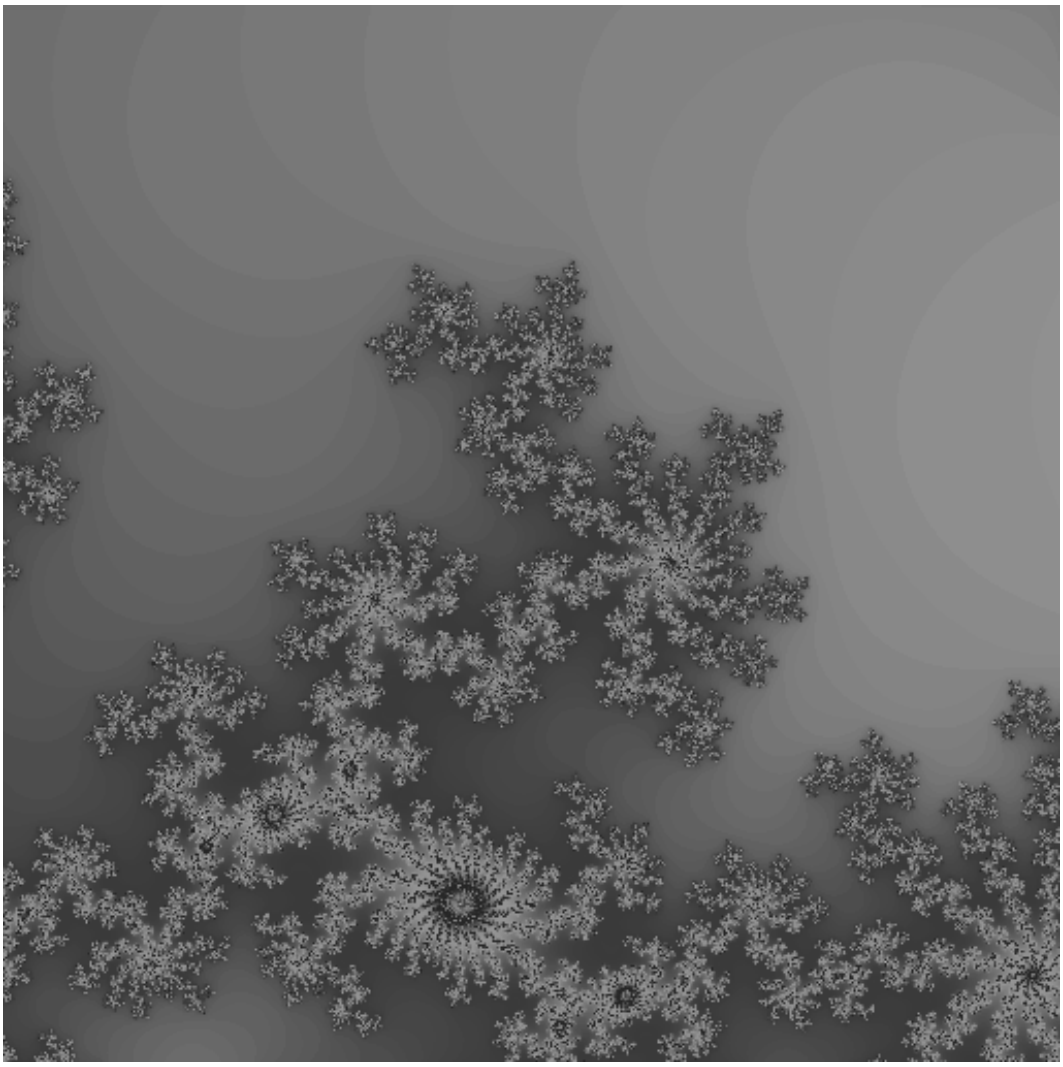


FIGURE 10. Uncompressed 512x512 image.

per pixel, greyscale, TIFF of $2^n \times 2^n$ dimension and performs the desired Haar wavelet compression.

APPENDIX B. IMAGES

FIGURE 11. DCT Lab compressed 512x512 image.

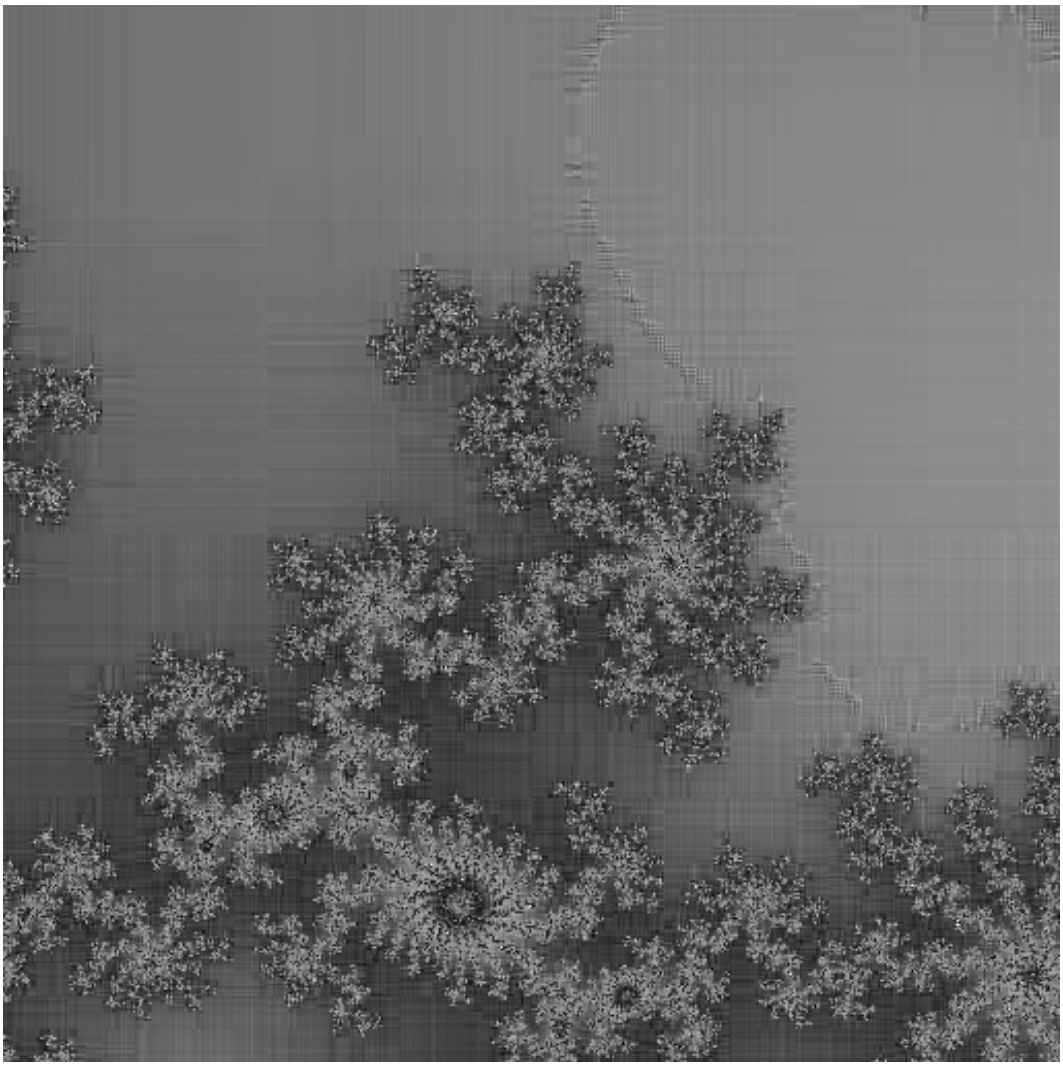


FIGURE 12. Wavelet compressed 512x512 image.

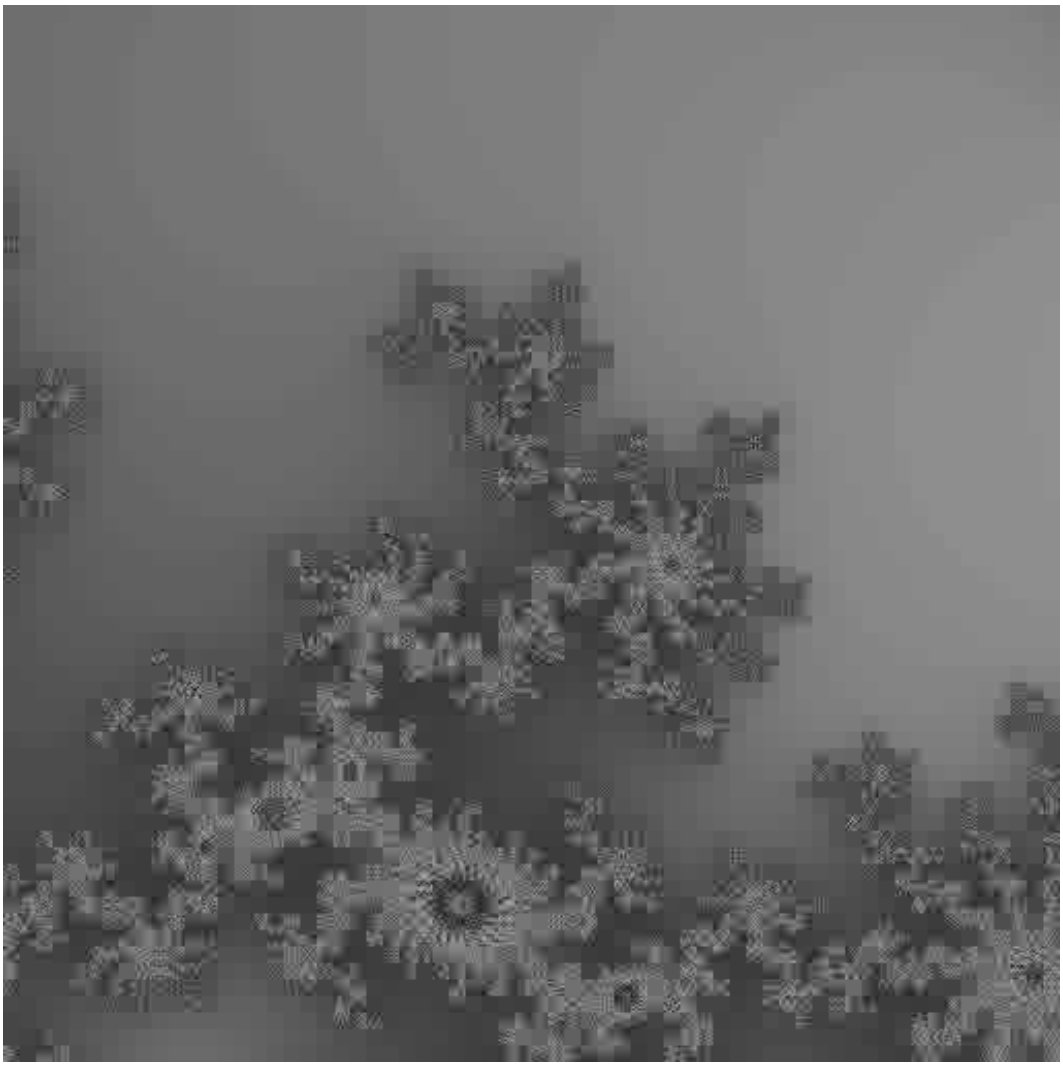


FIGURE 13. DCT compressed image with threshold level 70.

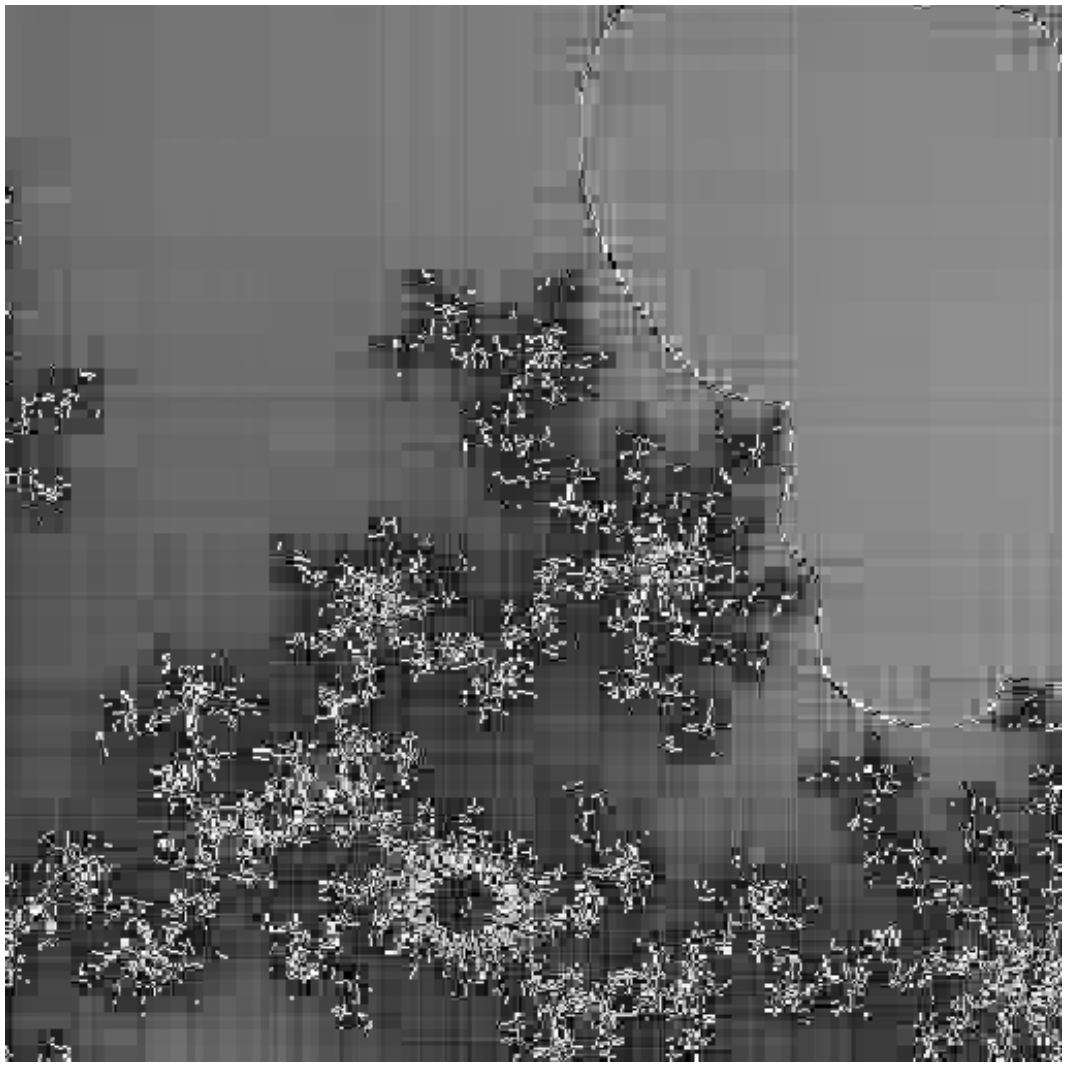


FIGURE 14. Wavelet compressed image with threshold level 150.

References

[CCI(1992)] Ccitt rec. t.81 (1992 e), 1992. URL http://www.w3.org/Graphics/JPEG/itu-t81.pdf.

[Brani Vidakovic(1994)] Peter Muller Brani Vidakovic. Wavelets for kids. Unpublished, 1994. URL http://www2.isye.gatech.edu/~brani/wavelet.html.

[Davidson and Donsig(2002)] Kenneth R. Davidson and Allan P. Donsig. *Real Analysis with Real Applications*. Prentince Hall, 2002. ISBN 0130416479.

[Eric J. Stollnitz(1995)] David H. Salesin Eric J. Stollnitz, Tony D. DeRose. Wavelets for computer graphics: A primer, part 1. *IEEE Computer Graphics and Applications*, 15 (3):76–84, May 1995.

[Graps(1995)] Amara L. Graps. An introduction to wavelets. *IEEE Computational Sciences and Engineering*, 2(2):50–61, 1995.

[Liu and Lee(1999)] Chi-Min Liu and Wen-Chieh Lee. A unified fast algorithm for cosine modulated filterbanks in current audio standards. *J. Audio Engineering*, 47(12):1061–1075, 1999.

[Mulcahy(1996)] Colm Mulcahy. Plotting and scheming with wavelets. *Mathematics Magazine*, 69(5):323–343, December 1996. ISSN 0025-570X.

[Strang(1999)] Gilbert Strang. The discrete cosine transform. *SIAM Review*, 41(1):135–147, March 1999. ISSN 0036-1445.

[Wallace(1991)] Gregory K. Wallace. The jpeg still picture compression standard. *Communications of the ACM*, 34(4):30–44, 1991. ISSN 0001-0782. doi: http://doi.acm.org/10.1145/103085.103089.

Department of Mathematics, Clayton State University, Morrow, GA
E-mail address: jsumners@clayton.edu
URL: http://student.claytonstate.net/~jsumners/wavelet/