

The Haar Wavelet Transform: Compression and Reconstruction

Damien Adams and Halsey Patterson

December 14, 2006

Abstract

The Haar Wavelet Transformation is a simple form of compression involved in averaging and differencing terms, storing detail coefficients, eliminating data, and reconstructing the matrix such that the resulting matrix is similar to the initial matrix. The Haar Wavelet Transform is a cornerstone in compressing computer images.

Introduction

Have you ever wondered why computers are able to transmit images so quickly? Why computers rarely run out of space? Why internet pages with so much information transmit information progressively, rather than immediately? These answers lie in the knowledge of image compression.

[Introduction](#)

[Images as Matrices](#)

[Averaging and . . .](#)

[The Averaging and . . .](#)

[Elimination Time](#)

[Reconstruction](#)

[Conclusion](#)

[Special Thanks](#)

[Home Page](#)

[Title Page](#)



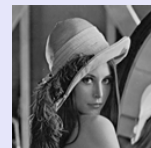
[Page 1 of 19](#)

[Go Back](#)

[Full Screen](#)

[Close](#)

[Quit](#)



[Introduction](#)

[Images as Matrices](#)

[Averaging and ...](#)

[The Averaging and ...](#)

[Elimination Time](#)

[Reconstruction](#)

[Conclusion](#)

[Special Thanks](#)

[Home Page](#)

[Title Page](#)



[Page 2 of 19](#)

[Go Back](#)

[Full Screen](#)

[Close](#)

[Quit](#)

Images on computers are made up of pixels — small squares of an image that, when thousands, sometimes millions, are placed very closely to each other, look like a continuous image. Mathematically, each image is represented by three matrices. Each of these matrix are the same in size but represent three colors, red, blue, and green. This is what is known as the RGB format. But inside each of these matrices are lots of numbers. These numbers represent the pixels.

The RGB format deals with three matrices stacked on top of each other, but the gray scale format deals with only a single matrix with each number representing a different shade between black and white. The value 0 represents purely black while the highest number represents the white. This paper will deal with gray scale images, for gray scale deals only with a single matrix rather than three times as many.

Images as Matrices

The first step in compressing an image is to be able to view an image as a matrix. Matlab, or other mathematics programs, can do this for us with a few simple commands (not discussed here). The images worked with here have been worked in Matlab.

An example of looking at an image as a matrix can be seen here. Look at the woman in Figure 1.

What you see here is a computer generated image of an interpretation of contrasts as described by a gray scale pixel matrix. Throughout this paper, we will be working with this image to demonstrate how images are compressed using the Haar Wavelet Transform. We will also be working with another image,



Figure 1: Woman



- [Introduction](#)
- [Images as Matrices](#)
- [Averaging and ...](#)
- [The Averaging and ...](#)
- [Elimination Time](#)
- [Reconstruction](#)
- [Conclusion](#)
- [Special Thanks](#)

[Home Page](#)

[Title Page](#)

[◀](#) [▶](#)

[◀](#) [▶](#)

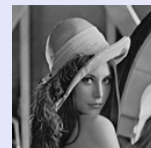
Page 3 of 19

[Go Back](#)

[Full Screen](#)

[Close](#)

[Quit](#)



an 8×8 matrix to demonstrate how this works in detail. We will let this matrix A be given below, along with an image interpretation in Figure 2.

$$A = \begin{bmatrix} 210 & 215 & 204 & 225 & 73 & 111 & 201 & 106 \\ 169 & 145 & 245 & 189 & 120 & 58 & 174 & 78 \\ 87 & 95 & 134 & 35 & 16 & 149 & 118 & 224 \\ 74 & 180 & 226 & 3 & 254 & 195 & 145 & 3 \\ 87 & 140 & 44 & 229 & 149 & 136 & 204 & 197 \\ 137 & 114 & 251 & 51 & 108 & 164 & 15 & 249 \\ 186 & 178 & 69 & 76 & 132 & 53 & 154 & 254 \\ 79 & 159 & 64 & 169 & 85 & 97 & 12 & 202 \end{bmatrix}$$

Once being able to extract a matrix from an image, what do we do with it? Well, the most basic idea, and the one that the Haar Wavelet Transform deals with, is to average and difference the rows and columns. For illustration, we'll just do this to the rows.

Averaging and Differencing

The process of averaging and differencing systematically averages paired entries in a matrix, entering the average in the first space of the resulting matrix. Then, the process finds the difference between the same paired entries from the average, providing a result that will be entered into the halfway point past the averaged value's spot. This may seem difficult to understand, but it is fairly simple and best shown by example.

Consider the row of a matrix

[Introduction](#)

[Images as Matrices](#)

[Averaging and ...](#)

[The Averaging and ...](#)

[Elimination Time](#)

[Reconstruction](#)

[Conclusion](#)

[Special Thanks](#)

[Home Page](#)

[Title Page](#)



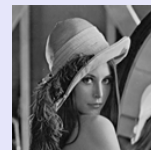
Page 4 of 19

[Go Back](#)

[Full Screen](#)

[Close](#)

[Quit](#)



Introduction

Images as Matrices

Averaging and ...

The Averaging and ...

Elimination Time

Reconstruction

Conclusion

Special Thanks

Home Page

Title Page



Page 5 of 19

Go Back

Full Screen

Close

Quit

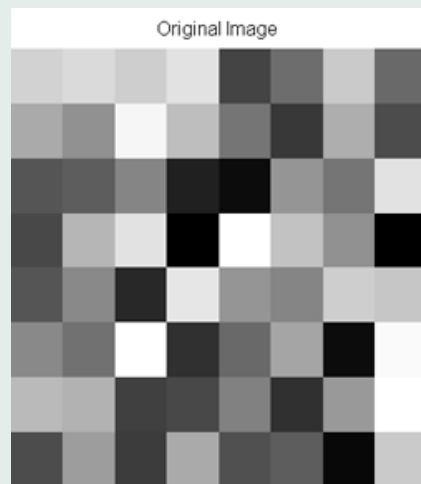
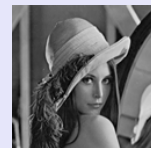


Figure 2: Matrix A as an Image Representation



$$[45 \ 11 \ 30 \ 24 \ 45 \ 38 \ 0 \ 23]$$

We want to pair the entries together, this way we have the pairs

$$\begin{array}{r} 45 \ 11 \\ 30 \ 24 \\ 45 \ 38 \\ 0 \ 23 \end{array}$$

Now, averaging each pair is simple, but differencing is the difference from the average. Thus, differencing could be thought of as a deviation from the average. Let's take a look.

Averaged	First Value – Average	Differenced
28	$45 - 28$	17
27	$30 - 27$	3
41.5	$45 - 41.5$	3.5
11.5	$0 - 11.5$	-11.5

We want to take these values and enter them into the new row that we have created from the old row. This new row will have the four averaged values in the first four spaces and the four differences in the last four spaces, respectively. Thus, we have

$$[28 \ 27 \ 41.5 \ 11.5 \ 17 \ 3 \ 3.5 \ -11.5]$$

[Introduction](#)

[Images as Matrices](#)

[Averaging and ...](#)

[The Averaging and ...](#)

[Elimination Time](#)

[Reconstruction](#)

[Conclusion](#)

[Special Thanks](#)

[Home Page](#)

[Title Page](#)



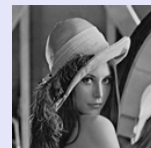
Page 6 of 19

[Go Back](#)

[Full Screen](#)

[Close](#)

[Quit](#)



from

$$[45 \ 11 \ 30 \ 24 \ 45 \ 38 \ 0 \ 23]$$

Now, we will want to repeat the process until we have only a single averaged value in the first entry. When we do this process to a matrix, we want this value in the first row, first column of our image matrix. But, let's continue with the row. This will be the stream of calculations to get to our final result.

$$\begin{array}{cccccccc} 45 & 11 & 30 & 24 & 45 & 38 & 0 & 23 \\ 28 & 27 & 41.5 & 11.5 & 17 & 3 & 3.5 & -11.5 \\ 27.5 & 26.5 & 0.5 & 15 & 17 & 3 & 3.5 & -11.5 \\ 27 & 0.5 & 0.5 & 15 & 17 & 3 & 3.5 & -11.5 \end{array}$$

This final row has our single averaged value in the leftmost entry with seven differenced numbers. These differenced numbers are called our “detail coefficients”. When we will later reconstruct the matrix, these detail coefficients will be put to work, doing exactly the opposite of what we have just done. They were found by differencing, so they will be put back to work doing just the opposite.

But this process isn't over. What we have just done is average and difference the rows. This will essentially created an overall average column in the first column of our matrix. What we want is an overall average *value* in the first entry, not a column, so we have to average and difference the columns. So we want to do the same process on the columns that we just did on the rows.

This process of averaging and differencing is called Wavelet Transforming a matrix. It is like applying a wave of matrices to your matrix.

- Introduction
- Images as Matrices
- Averaging and ...
- The Averaging and ...
- Elimination Time
- Reconstruction
- Conclusion
- Special Thanks

[Home Page](#)

[Title Page](#)

⏪ ⏩

◀ ▶

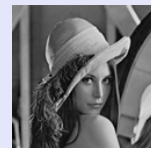
Page 7 of 19

[Go Back](#)

[Full Screen](#)

[Close](#)

[Quit](#)

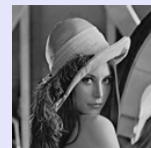
[Introduction](#)[Images as Matrices](#)[Averaging and . . .](#)[The Averaging and . . .](#)[Elimination Time](#)[Reconstruction](#)[Conclusion](#)[Special Thanks](#)[Home Page](#)[Title Page](#)[Page 8 of 19](#)[Go Back](#)[Full Screen](#)[Close](#)[Quit](#)

The Averaging and Differencing Matrix

Linear Algebra is an extremely powerful tool, and we have this tool at our disposal. Knowing this, why would we want to average and difference everything by hand? Especially if we have to do it many times. What we just did with a row with eight columns took four steps. So imagine if we had a large-sized image taken by a digital camera — say, a 2000×2000 pixel image. This represents a 2000 row by 2000 column image matrix, a total of 4,000,000 pixels to deal with in a matrix! And what if it was an RGB image? We don't even want to go there.

So what would our matrix look like? It's fairly simplistic, considering what we have to do. Think back to elementary matrices and block multiplication. When we want to add row one to row two in a 3×3 matrix, we take I and stick a 1 in the second row, first column. That 1 will add row one to row two (thus, entering it in row 2, column 1).

For our case, we don't want to add row one to row two; we want to add half of row one to half of row two (and half of column one to half of column two), half of row three to half of row four, and on down the line. When we think of elementary matrices, we will end up with this matrix, which we will denote as W_1 , for an 8×8 .



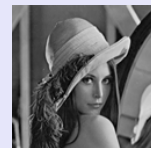
$$W_1 = \begin{bmatrix} 1/2 & 0 & 0 & 0 & 1/2 & 0 & 0 & 0 \\ 1/2 & 0 & 0 & 0 & -1/2 & 0 & 0 & 0 \\ 0 & 1/2 & 0 & 0 & 0 & 1/2 & 0 & 0 \\ 0 & 1/2 & 0 & 0 & 0 & -1/2 & 0 & 0 \\ 0 & 0 & 1/2 & 0 & 0 & 0 & 1/2 & 0 \\ 0 & 0 & 1/2 & 0 & 0 & 0 & -1/2 & 0 \\ 0 & 0 & 0 & 1/2 & 0 & 0 & 0 & 1/2 \\ 0 & 0 & 0 & 1/2 & 0 & 0 & 0 & -1/2 \end{bmatrix}$$

This matrix will average and difference the first step. We can see that, with block multiplication (as described above), the matrix A times the first column (thus we will be computing AW_1) will produce the first and second values of each row's average. The matrix A times the second column will average the third and fourth values, et cetera. But we need matrices for the next step, also. So, with block multiplication, again, we want to get the average and deviations of the first four values, but the last four differences remain the same. So, obviously, we will have a block matrix that will look exactly like this:

$$W_2 = \begin{bmatrix} W_{2 \times 2} & 0 \\ 0 & I \end{bmatrix}$$

So, our W_2 should obviously look like this after expanding our block.

[Introduction](#)[Images as Matrices](#)[Averaging and ...](#)[The Averaging and ...](#)[Elimination Time](#)[Reconstruction](#)[Conclusion](#)[Special Thanks](#)[Home Page](#)[Title Page](#)[Page 9 of 19](#)[Go Back](#)[Full Screen](#)[Close](#)[Quit](#)

[Introduction](#)[Images as Matrices](#)[Averaging and ...](#)[The Averaging and ...](#)[Elimination Time](#)[Reconstruction](#)[Conclusion](#)[Special Thanks](#)

$$W_2 = \begin{bmatrix} 1/2 & 0 & 1/2 & 0 & 0 & 0 & 0 & 0 \\ 1/2 & 0 & -1/2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1/2 & 0 & 1/2 & 0 & 0 & 0 & 0 \\ 0 & 1/2 & 0 & -1/2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

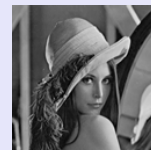
Thus, repeating this same block for W_3 should produce yet another obvious matrix.

$$W_3 = \begin{bmatrix} 1/2 & 1/2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1/2 & -1/2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Thus, the Averaging and Differencing Matrix is $W = W_1 W_2 \dots W_n$, and in our case with our 8×8 , $W = W_1 W_2 W_3$.

When we put our matrix W to work, we will use the matrix multiplication to multiply the columns of W with the rows of A . But again, what about the columns of A ?

[Home Page](#)[Title Page](#)[Page 10 of 19](#)[Go Back](#)[Full Screen](#)[Close](#)[Quit](#)



[Introduction](#)

[Images as Matrices](#)

[Averaging and ...](#)

[The Averaging and ...](#)

[Elimination Time](#)

[Reconstruction](#)

[Conclusion](#)

[Special Thanks](#)

[Home Page](#)

[Title Page](#)



Page 11 of 19

[Go Back](#)

[Full Screen](#)

[Close](#)

[Quit](#)

To average and difference the rows, we multiply on the left by W . With the rules of matrix multiplication, the “college algebra method” says that we multiply the rows of the left matrix with the columns of the right matrix. This means that if AW averages and differences the rows of A , then $W^T A$ should average and difference the columns of A .

We end up with the multiplication on the left and right resulting in $T = W^T AW$ where T is our wavelet transformed matrix. So here is our equation:

$$T = W^T AW$$

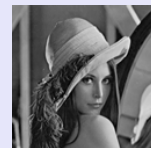
Later on, we will be taking the inverse of our W . Since taking the inverse of a matrix is pretty tedious, and because orthogonal matrices are easy to invert, then let’s go ahead and make our matrix W orthogonal! And this will be simple in that all we must do is change all of our $1/2$ to $1/\sqrt{2}$ s!

But what good is this, really? We are still storing the same amount of information as before, so we haven’t done anything. So let’s eliminate some data!

Elimination Time

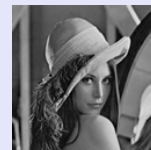
What good is averaging and differencing? Sure, it knocks the image matrix down to a bunch of detail coefficients and a lone average value, but we haven’t gotten rid of any data! So, let’s get rid of some stuff.

What would be the best way to efficiently eliminated a bunch of information? Well, we have an average value that we will use to reconstruct our original

[Introduction](#)[Images as Matrices](#)[Averaging and ...](#)[The Averaging and ...](#)[Elimination Time](#)[Reconstruction](#)[Conclusion](#)[Special Thanks](#)[Home Page](#)[Title Page](#)[Page 12 of 19](#)[Go Back](#)[Full Screen](#)[Close](#)[Quit](#)

matrix A out of using our detail coefficients. Our detail coefficients represent the difference in darkness of our pixels that are directly next to each other. This means that large detail coefficients signify a large change between the pixels directly next to each other, and small detail coefficients represent little difference from the pixels next to each other (large and small meaning the absolute value of the detail coefficient being large or small). So why not just let the pixels next to each other with little difference be the same? By this, we say that we should let the absolute value of the detail coefficients close to zero become zero. But what values should we eliminate? What constitutes large or small? Well, that depends upon how much data we want to eliminate. So let's designate a threshold value that we will set all values with absolute value equal to or less than our threshold value to zero. We will call this threshold value ϵ .

After having averaged and differenced an image, we will have a bunch of differences close to zero. The phrase “close to zero” is almost completely interpretive at this point — although there are algorithms that dictate an appropriate threshold value, those algorithms are beyond our current level of understanding, so our strategy is simply “trial and error.” So (after we’ve messed with it a bit and decided upon this value) let’s let ϵ be equal to, simply, 1. We will eliminate all of the values in the entire matrix, not just this section, that have values of less than or equal to 1, or in our case, all values of 1. Now, we can make this a sparse matrix. When making a sparse matrix, one will eliminate all of the zeros in the matrix, but it will add a little “caption” type message that says, “the remaining entries go in these spaces of the matrix.” For example, if you have a 20×20 identity matrix and make it sparse, it will result in a 20×1 column vector full of ones with captions that tell the user to place the ones on

[Introduction](#)[Images as Matrices](#)[Averaging and . . .](#)[The Averaging and . . .](#)[Elimination Time](#)[Reconstruction](#)[Conclusion](#)[Special Thanks](#)[Home Page](#)[Title Page](#)[Page 13 of 19](#)[Go Back](#)[Full Screen](#)[Close](#)[Quit](#)

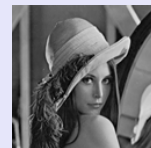
the diagonal. This makes it a whole lot easier to store information — storing a 20×20 matrix versus storing a 20×1 column vector, you do the thinking. This is why we choose a threshold value to eliminate entries. We want to make our matrix sparse! This new sparse matrix is a doctored matrix that we will denote as S .

Reconstruction

The Haar Wavelet Transform reconstructs images using a technique that progressively reconstructs the image. Reconstructing the image is not instantaneous — it takes time. You may not realize it, but you can see this happening all of the time on the internet. If you watch carefully (or if you have a slow enough internet connection), you can see images that load from a lesser quality image to its precise high quality that results at the end of this “progressive transform.”

Why does this happen? Well, the answer is right in front of you. When the image is compressed, it is hit with a myriad of matrices. When it is reconstructed, the same this happens. Each of the detail coefficients is used to reconstruct this image to as close as it can to its original form, but the reason it takes time is due to re-entering the zeros back into the matrix from the scarce matrix. Once the image is “reformed” with the zeros, we can again reconstruct the image.

Reconstructing the image is simple. Since we already have a matrix W that compresses the image down to its detail coefficients and average value, we can simply reconstruct the image by taking the inverse of our matrix W , and multiply our W^{-1} on the left by our sparse matrix, S , to progressively transform

[Introduction](#)[Images as Matrices](#)[Averaging and ...](#)[The Averaging and ...](#)[Elimination Time](#)[Reconstruction](#)[Conclusion](#)[Special Thanks](#)[Home Page](#)[Title Page](#)[Page 14 of 19](#)[Go Back](#)[Full Screen](#)[Close](#)[Quit](#)

the matrix back to close to its original state. It isn't exactly what it used to be, but that's okay. We don't want our original, we want a compressed version. What we want is an approximation of what it used to be, and we want our approximation to be sparse.

Recall that we have eliminated all values of ϵ or lesser from our matrix to obtain a sparse matrix S . To reconstruct our image, we need to find W^{-1} . This is simple. Since we know that $W = W_1W_2W_3$, we can easily find the inverse, especially knowing that W is orthogonal.

The defining property of orthogonal matrices is that $I = Q^TQ$. Thus, if $Q^TQ = I$, then $Q^T = Q^{-1}$. In our context, we have made W an orthogonal matrix, so it retains all properties of orthogonal matrices. Since this is true, can you see how easy it would be to invert W ? It should be pretty obvious that $W^{-1} = W^T$. Thus, to reconstruct our compressed image, we want to simply invert it, and to do that, we simply need to multiply our compressed image on the left by W^T . This gives us this string of equalities:

$$T = W_n^{-1} \dots W_3^{-1}W_2^{-1}W_1^{-1}AW_1W_2W_3 \dots W_n = W^{-1}AW$$

Now we want to compute. So let's look at the woman in Figure 1, again (for a later comparison in Figure 3. We've doctored the woman's image to get rid of a bunch of information that we don't want. We just need to invert W to get our reconstruction matrix W^{-1} . Our case is simple:

$$W^{-1} = W^T = (W_1W_2W_3)^T = W_3^TW_2^TW_1^T$$

Thus, we will use this W :

[Introduction](#)[Images as Matrices](#)[Averaging and ...](#)[The Averaging and ...](#)[Elimination Time](#)[Reconstruction](#)[Conclusion](#)[Special Thanks](#)[Home Page](#)[Title Page](#)[◀](#) [▶](#)[◀](#) [▶](#)[Page 15 of 19](#)[Go Back](#)[Full Screen](#)[Close](#)[Quit](#)

$$W = \begin{bmatrix} 1/\sqrt{8} & 1/\sqrt{8} & 1/2 & 0 & 1/\sqrt{2} & 0 & 0 & 0 \\ 1/\sqrt{8} & 1/\sqrt{8} & 1/2 & 0 & -1/\sqrt{2} & 0 & 0 & 0 \\ 1/\sqrt{8} & 1/\sqrt{8} & -1/2 & 0 & 0 & 1/\sqrt{2} & 0 & 0 \\ 1/\sqrt{8} & 1/\sqrt{8} & -1/2 & 0 & 0 & -1/\sqrt{2} & 0 & 0 \\ 1/\sqrt{8} & -1/\sqrt{8} & 0 & 1/2 & 0 & 0 & 1/\sqrt{2} & 0 \\ 1/\sqrt{8} & -1/\sqrt{8} & 0 & 1/2 & 0 & 0 & -1/\sqrt{2} & 0 \\ 1/\sqrt{8} & -1/\sqrt{8} & 0 & -1/2 & 0 & 0 & 0 & 1/\sqrt{2} \\ 1/\sqrt{8} & -1/\sqrt{8} & 0 & -1/2 & 0 & 0 & 0 & -1/\sqrt{2} \end{bmatrix}$$

And this W^T :

$$W^T = \begin{bmatrix} 1/\sqrt{8} & 1/\sqrt{8} & 1/\sqrt{8} & 1/\sqrt{8} & 1/\sqrt{8} & 1/\sqrt{8} & 1/\sqrt{8} & 1/\sqrt{8} \\ 1/\sqrt{8} & 1/\sqrt{8} & 1/\sqrt{8} & 1/\sqrt{8} & -1/\sqrt{8} & -1/\sqrt{8} & -1/\sqrt{8} & -1/\sqrt{8} \\ 1/2 & 1/2 & -1/2 & -1/2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/2 & 1/2 & -1/2 & -1/2 \\ 1/\sqrt{2} & -1/\sqrt{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/\sqrt{2} & -1/\sqrt{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/\sqrt{2} & -1/\sqrt{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1/\sqrt{2} & -1/\sqrt{2} \end{bmatrix}$$

When we act these matrices upon the matrix S (which we remember to be our sparse matrix), we obtain the reconstructed compressed matrix R , an approximation of our original image matrix A .



Figure 3: Original vs. Compressed with $\epsilon = 1$ and $\epsilon = 5$

$$R = W^{-1}SW = W^T SW$$

After calculating our new approximation, R , let's compare the end result with our original matrix. We will do the same thing with the larger images of the woman and show the resulting images. Note that these images should NOT be the same. We are compressing the image, not recreating the same thing.

Because the images look similar in Figure 3 for $\epsilon = 1$, the compression ratio is a success. The compression of the woman is pretty ideal, because $\epsilon = 1$. But, for $\epsilon = 5$, you can hardly tell it's a woman, so it is too lossy — a term used to describe the amount of data lost. Because this we choose such a small threshold for $\epsilon = 1$, we can say that this matrix is not very lossy. We did not lose much, so the image still looks close to the original, so this is not a very large compression. As for our matrix A , we chose a threshold value of 25, a



[Introduction](#)

[Images as Matrices](#)

[Averaging and ...](#)

[The Averaging and ...](#)

[Elimination Time](#)

[Reconstruction](#)

[Conclusion](#)

[Special Thanks](#)

[Home Page](#)

[Title Page](#)



Page 16 of 19

[Go Back](#)

[Full Screen](#)

[Close](#)

[Quit](#)

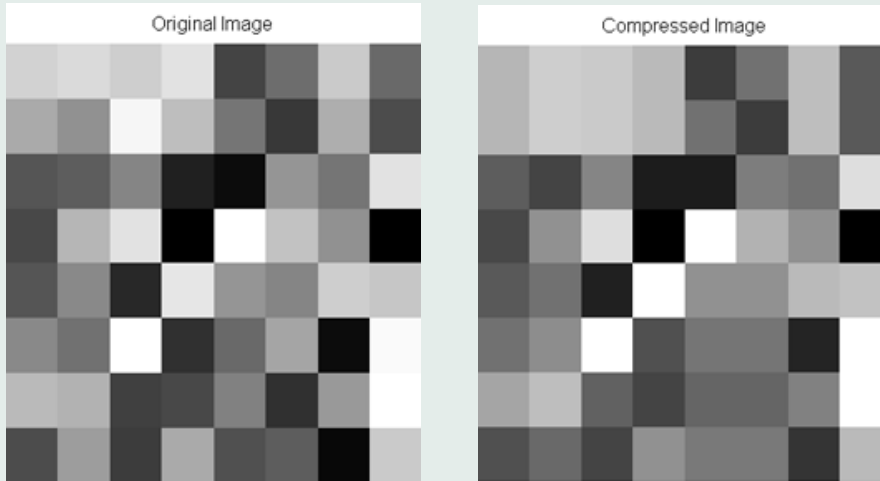


Figure 4: Image of A , Original vs. Wavelet Compressed with $\epsilon = 25$

relatively lossy compression ratio. Thus, our reconstructed matrix R from A is considered lossy.

Conclusion

To sum up the Haar Wavelet Transformation, you start with an image matrix, A . You average and difference that matrix until you have one average value in row 1, column 1 of your matrix. Multiply all of the averaging and differencing

[Introduction](#)

[Images as Matrices](#)

[Averaging and ...](#)

[The Averaging and ...](#)

[Elimination Time](#)

[Reconstruction](#)

[Conclusion](#)

[Special Thanks](#)

[Home Page](#)

[Title Page](#)



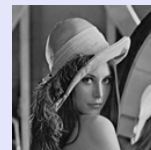
Page 17 of 19

[Go Back](#)

[Full Screen](#)

[Close](#)

[Quit](#)

[Introduction](#)[Images as Matrices](#)[Averaging and ...](#)[The Averaging and ...](#)[Elimination Time](#)[Reconstruction](#)[Conclusion](#)[Special Thanks](#)

matrices together to get your matrix W . Select an ϵ such that it zeros out all values whose absolute value is less than or equal to ϵ . Call this new matrix the sparse, doctored matrix S . Hit S with the inverse of W on the right and the transposed inverse of W , to get your reconstructed compressed matrix R , which should be an approximated image of your initial image A .

The Haar Wavelet Transformation is only a glimpse of what linear algebra is capable of with compression and reconstruction of computer images. In fact, most of the transformations are based off of the Haar Wavelet, but to understand the more complicated transformations, you have to first understand what they are based upon. That is the Haar Wavelet Transformation.

Special Thanks

Thanks to Dave Arnold for all of the help, to Colm Mulcahy for the great Haar Wavelet Transform paper and the matrices that are used in Matlab to wavelet compress these images, and to Gilbert Strang for writing an awesome book to teach us this gigantic amount of information.

References

- [1] Arnold, David, *2006 Math 45, College of the Redwoods*
- [2] Mulcahy, Colm, *1996 Image compression using the Haar wavelet transform*

[Home Page](#)[Title Page](#)[Page 18 of 19](#)[Go Back](#)[Full Screen](#)[Close](#)[Quit](#)



[Introduction](#)

[Images as Matrices](#)

[Averaging and . . .](#)

[The Averaging and . . .](#)

[Elimination Time](#)

[Reconstruction](#)

[Conclusion](#)

[Special Thanks](#)

[Home Page](#)

[Title Page](#)



[Page 19 of 19](#)

[Go Back](#)

[Full Screen](#)

[Close](#)

[Quit](#)

[3] Mulcahy, Colm, <http://www.spelman.edu/~colm/matlabwav.html> for the *Haar Wavelet Transform matrices and Matlab Programs*

[4] Strang, Gilbert, *1998 Introduction to Linear Algebra*