Image compression using the Haar wavelet transform

Colm Mulcahy, Ph.D.

ABSTRACT

The wavelet transform is a relatively new arrival on the mathematical scene. We give a brief introduction to the subject by showing how the Haar wavelet transform allows information to be encoded according to "levels of detail." In one sense, this parallels the way in which we often process information in our everyday lives. Our investigations enable us to give two interesting applications of wavelet methods to digital images: compression and progressive transmission.

The mathematical prerequesites will be kept to a minimum; indeed, the main concepts can be understood in terms of addition, subtraction and division by two. We also present a linear algebra implementation of the Haar wavelet transform, and mention important recent generalizations.

This material is suitable for undergraduate research by both mathematics and computer science majors, and has been used successfully in this way at Spelman College since 1995.

1 So much information

In the real world, we are constantly faced with having to decide how much information to absorb or impart.

Example 1: An election is impending, and you and your best friend have yet to decide for whom you will cast your votes. Your friend is content to base her decision solely on the political party affiliations of the available candidates, whereas you take the trouble to find out how the candidates stand on a variety of issues, and to investigate their records for integrity, effectiveness, and vision.

Example 2: Your younger brother is across the room, working on a project for Black History Month. From where you are sitting, you can see that he is putting a caption on a photograph. "Who is in that picture?" you ask. "A person sitting in a bus," he says evasively. "I can see that," you reply impatiently, "But who is it?" "A woman," he says, grinning impishly. "Is that all you are going to tell me?" you say in exasperation, getting up. "A young woman," he volunteers. Noticing the look of displeasure on your face, he quickly adds, "It's Rosa Parks."

What these examples all have in common is *information transmission at various levels of detail*. In the first example, each person received information up to, but not beyond, a certain level of detail. The second example illustrates *progressive information transmission*: We start with a crude approximation, and over time more and more details are added in, until finally a "full picture" emerges. Figure 1 illustrates this pictorially: looking from left to right we see progressively more recognizable images of Rosa Parks.

Figure 1: Person in bus — Woman in bus — Young woman in bus — Rosa Parks in bus

Dr. Colm Mulcahy is Associate Professor of Mathematics at Spelman College, where he has taught since 1988. In recent years, his mathematical interests have broadened to include computational and visualization issues, such as computer aided geometric design (CAGD), computer graphics, image processing, and wavelets, and he has directed undergraduate student research in all of these topics.

There are two observations worth making here. Firstly, there are circumstances under which any one of these approximations would be adequate for our immediate purposes. For instance, viewed from a sufficient distance, they all look the same. Thus, if one of these was to form a small part of a much larger picture, say a photo on a mantlepiece, or a fleeting image in a video, there would be no need to display a high quality version.

Secondly, progressive transmission of a sequence of ever-improving approximations to the "real picture" is natural: it's the way many of us relay information, and the way we learn new subjects. It's also the way the popular Netscape World Wide Web browser delivers images to users of the web: when we call up a URL (WWW address) which contains an image, that image appears in installments, starting with an approximation and working up to the final complete image. All forms of progressive information transmission have one major advantage: the receiver can halt the process and move onto something else if she decides, based on early information, that she does not want "the whole picture." This applies equally to learning about a candidate for election, listening to people recount their vacation experiences, or grabbing an image on the World Wide Web using Netscape.

Wavelets provide a mathematical way of encoding numerical information (data) in such a way that it is layered according to level of detail. This layering not only facilitates the progressive data transmission mentioned above, but also approximations at various intermediate stages. The point is that these approximations can be stored using a lot less space than the original data, and in situations where space is tight, this *data compression* is well worthwhile.

2 The wavelet transform

In this section, we introduce the simplest wavelet transform, the so-called *Haar wavelet transform*, and explain how it can be used to produce images like the first three in Figure 1, given the last, complete image of Rosa Parks (this image was extracted from a *.gif* image file downloaded from the World Wide Web.) *Matlab* numerical and visualization software was used to perform all of the calculations and generate and display all of the pictures in this manuscript.

Each of the digital images in Figure 1 is represented mathematically by a 128×128 matrix (array) of numbers, ranging from 0 (representing black) to some positive whole number (representing white). The last image uses $32 = 2^5$ different shades of gray, and as such is said to be a 5-*bit image*. The numbers in the specific matrix we used to represent this image range from 0 to 1984, in 31 increments of 64 (these exact numbers are not important; they were chosen to avoid fractions in certain calculations later on).

Each matrix entry gives rise to a small square which is shaded a constant gray level according to its numerical value. We refer to these little squares as *pixels*; they are more noticable as individual squares when we view the images on a larger scale, such as in Figure 2(a). Given enough of them on a given region of paper, as in the 256×256 pixel 8-bit image of Nelson Mandela in Figure 2(b), we experience the illusion of a continuously shaded photograph.

Figure 2: Rosa Parks (1955) and Nelson Mandela (1990)

The matrices which specify these images have $128^2 = 16,384$ and $256^2 = 65,536$ elements, respectively. This fact poses immediate storage problems. Color images are even bigger, though in one sense they can be handled by decomposing into three "grayscale-like" arrays, one for each of the colors red, green and blue. A standard 1.44MB high density floppy disc can only accommodate a handful of large (say, 1024×1024 pixel) high quality color images; furthermore, as many of us

can attest from personal experience, such images are very time consuming to download on the World Wide Web. We only consider grayscale images here.

We describe a scheme for transforming such large arrays of numbers into arrays that can be stored and transmitted more efficiently; the original images (or good approximations of them) can then be reconstructed by a computer with relatively little effort. For simplicity, we first consider the $8 \times 8 = 64$ pixel image in Figure 3(b), which is a blow up of a region around the nose in Figure 2. The region extracted is blacked out in Figure 3(a).

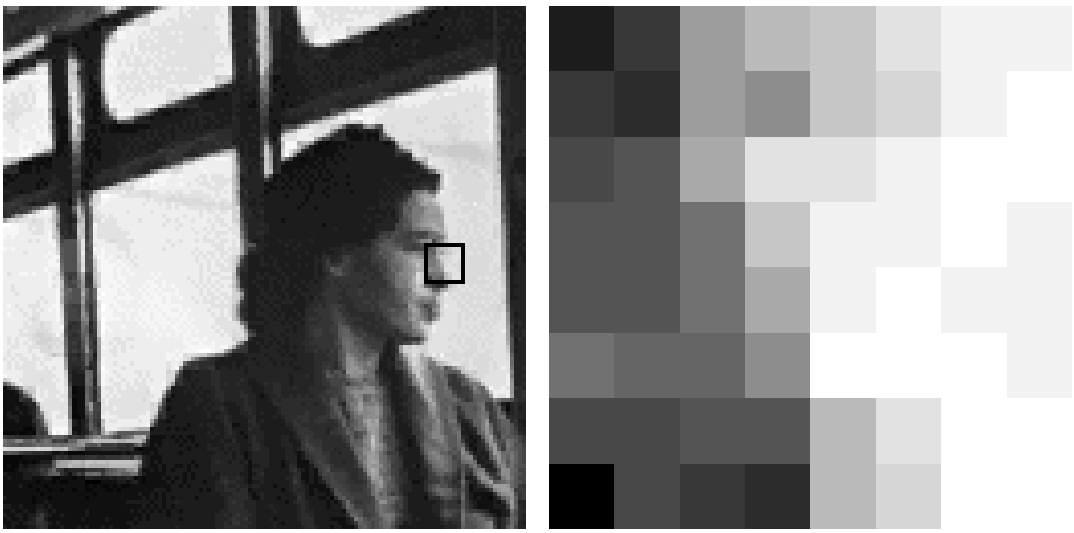

Figure 3: Rosa Parks - extracting a sub-region for inspection

This image is represented by rows 60 to 67 and columns 105 to 112 of the matrix which defines Figure 2. We now display and name this submatrix:

$$P = \begin{pmatrix} 576 & 704 & 1152 & 1280 & 1344 & 1472 & 1536 & 1536 \\ 704 & 640 & 1156 & 1088 & 1344 & 1408 & 1536 & 1600 \\ 768 & 832 & 1216 & 1472 & 1472 & 1536 & 1600 & 1600 \\ 832 & 832 & 960 & 1344 & 1536 & 1536 & 1600 & 1536 \\ 832 & 832 & 960 & 1216 & 1536 & 1600 & 1536 & 1536 \\ 960 & 896 & 896 & 1088 & 1600 & 1600 & 1600 & 1536 \\ 768 & 768 & 832 & 832 & 1280 & 1472 & 1600 & 1600 \\ 448 & 768 & 704 & 640 & 1280 & 1408 & 1600 & 1600 \end{pmatrix}.$$

(We have *contrast stretched* in Figure 3(b) to highlight the often subtle variations among the various shades of gray: the smallest and largest matrix entries, 448 and 1600, are depicted as black and white, respectively, which is not the way they appeared in Figure 2(a).)

To demonsrate how to wavelet transform such a matrix, we first describe a method for transforming strings of data, called *averaging and differencing*. Later, we'll use this technique to transform an entire matrix as follows: Treat each row as a string, and perform the averaging and differencing on each one to obtain a new matrix, and then apply exactly the same steps on each column of this new matrix, finally obtaining a row and column transformed matrix.

To understand what averaging and differencing does to a data string, for instance the first row in the matrix P above, consider the table below. Successive rows of the table show the starting, intermediate, and final results.

576	704	1152	1280	1344	1472	1536	1536
640	1216	1408	1536	**-64**	**-64**	**-64**	**0**
928	1472	**-288**	**-64**	**-64**	**-64**	**-64**	**0**
1200	**-272**	**-288**	**-64**	**-64**	**-64**	**-64**	**0**

There are 3 steps in the transform process because the data string has length $8 = 2^3$. The first row in the table is our original data string, which we can think of as four pairs of numbers. The first four numbers in the second row are the

averages of those pairs. Similarly, the first two numbers in the third row are the averages of those four averages, taken two at a time, and the first entry in the fourth and last row is the average of the preceding two computed averages.

The remaining numbers, shown in bold, measure deviations from the various averages. The first four bold entries, in the second half of the second row, are the result of subtracting the first four averages from the first elements of the pairs that gave rise to them: subtracting $640, 1216, 1408, 1536$ from $576, 1152, 1344, 1536$, element by element, yields $\mathbf{-64}, \mathbf{-64}, \mathbf{-64}, \mathbf{0}$. These are called *detail coefficients*; they are repeated in each subsequent row of the table. The third and fourth entries in the third row are obtained by subtracting the first and second entries in that row from the first elements of the pairs that start row two: subtracting $928, 1472$ from $640, 1408$, element by element, yields $\mathbf{-288}, \mathbf{-64}$. These two new detail coefficients are also repeated in each subsequent row of the table. Finally, the second entry in the last row, $\mathbf{-272}$, is the detail coefficient obtained by subtracting the overall average, 1200, from the 928 that starts row three.

We have transformed our original string into a new string in 3 steps. Moreover, *the averaging and differencing process is reversible:* we can work back from any row in the table to the previous row—and hence to the first row—by means of appropriate additions and subtractions. In other words, we have lost nothing by transforming our string.

(Clearly, the process can be generalized to strings of any length: strings of length 2^k require k rounds of averaging and differencing, and any string can be padded at the end, say with zeros, until it has length equal to a power of two.)

To apply the scheme to an 8×8 matrix, we simply do the averaging and differencing three times on each row separately, and then three times on the columns of the resulting matrix. Averaging and differencing columns can also be achieved by transposing the row-transformed matrix, doing row transformations to the result of that transposition, and transposing back. The final result is a new 8×8 matrix T, called the *Haar wavelet transform* of P.

Applying this technique to the matrix P as above, we obtain, after a great deal of calculation, this transformed matrix:

$$T = \begin{pmatrix} 1212 & -306 & -146 & -54 & -24 & -68 & -40 & 4 \\ 30 & 36 & -90 & -2 & 8 & -20 & 8 & -4 \\ -50 & -10 & -20 & -24 & 0 & 72 & -16 & -16 \\ 82 & 38 & -24 & 68 & 48 & -64 & 32 & 8 \\ 8 & 8 & -32 & 16 & -48 & -48 & -16 & 16 \\ 20 & 20 & -56 & -16 & -16 & 32 & -16 & -16 \\ -8 & 8 & -48 & 0 & -16 & -16 & -16 & -16 \\ 44 & 36 & 0 & 8 & 80 & -16 & -16 & 0 \end{pmatrix}.$$

This matrix has one overall average value in the top left hand corner, and 63 detail elements. (The first row is not the same as the last row in the table we saw before, since this time, column as well as row transformations have been done).

The point of the wavelet transform is that *regions of little variation in the original data manifest themselves as small or zero elements in the wavelet transformed version.* The 0's in T are due to the occurrences of identical adjacent elements in P, and the $-2, -4$, and 4 in T can be explained by some of the nearly identical adjacent elements in P.

A matrix with a high proportion of zero entries is said to be *sparse*. For many image matrices, their corresponding wavelet transformed versions are much sparser than the originals. Very sparse matrices are easier to store and tramsmit than ordinary matrices of the same size. This is because sparse matrices can be specified in a data file solely in terms of the locations and values of their nonzero entries: just think how much more efficient it is to say "a 100×100 matrix which is all 0's except for 7's in the $(23, 44)$ and $(65, 2)$ positions" than it is to show all $10,000$ entries.

To benefit more fully from the wavelet transform, we must think big: 24% of the entries of the wavelet transform of the 256×256 matrix defining the picture of Nelson Mandela in Figure 2(b) are 0's.

3 Compression

The real pay-off in the wavelet transmission game is not so much the expectation of sparsity of the transformed matrices, it's the fact that we can fiddle with the "mostly detail" versions to make lots of entries zero: we can alter the transformed matrices, taking advantage of "regions of low activity," and then apply the inverse wavelet transform to this doctored version, to obtain an approximation of the original data.

Thus we arrive at the door of *wavelet compression*: Fix a nonnegative *threshold value* ϵ, and decree that any detail coefficient in the wavelet transformed data whose magnitude is less than or equal to ϵ will be reset to zero (hopefully, this leads to a relatively sparse matrix), then rebuild an approximation of the original data using this doctored version of the wavelet transformed data. The surprise is that in the case of image data, we can throw out a sizable proportion of the detail coefficients in this way and obtain visually acceptable results. This process is called *lossless compression* when no information is lost (e.g., if $\epsilon = 0$); otherwise it's referred to as *lossy compression* (in which case $\epsilon > 0$). In the former case we can get our original data back, and in the latter we can build an approximation of it.

For instance, consider the 8×8 image matrix P and its wavelet transformed version T from before. If we take $\epsilon = 20$, i.e., reset to zero all elements of T which are less than or equal to 20 in absolute value, we obtain the doctored matrix:

$$D = \begin{pmatrix} 1212 & -306 & -146 & -54 & -24 & -68 & -40 & 0 \\ 30 & 36 & -90 & 0 & 0 & 0 & 0 & 0 \\ -50 & 0 & 0 & -24 & 0 & 72 & 0 & 0 \\ 82 & 38 & -24 & 68 & 48 & -64 & 32 & 0 \\ 0 & 0 & -32 & 0 & -48 & -48 & 0 & 0 \\ 0 & 0 & -56 & 0 & 0 & 32 & 0 & 0 \\ 0 & 0 & -48 & 0 & 0 & 0 & 0 & 0 \\ 44 & 36 & 0 & 0 & 80 & 0 & 0 & 0 \end{pmatrix}.$$

Applying the inverse wavelet transform to D, we get this reconstructed approximation R:

$$R = \begin{pmatrix} 582 & 726 & 1146 & 1234 & 1344 & 1424 & 1540 & 1540 \\ 742 & 694 & 1178 & 1074 & 1344 & 1424 & 1540 & 1540 \\ 706 & 754 & 1206 & 1422 & 1492 & 1572 & 1592 & 1592 \\ 818 & 866 & 1030 & 1374 & 1492 & 1572 & 1592 & 1592 \\ 856 & 808 & 956 & 1220 & 1574 & 1590 & 1554 & 1554 \\ 952 & 904 & 860 & 1124 & 1574 & 1590 & 1554 & 1554 \\ 776 & 760 & 826 & 836 & 1294 & 1438 & 1610 & 1610 \\ 456 & 760 & 668 & 676 & 1278 & 1422 & 1594 & 1594 \end{pmatrix}.$$

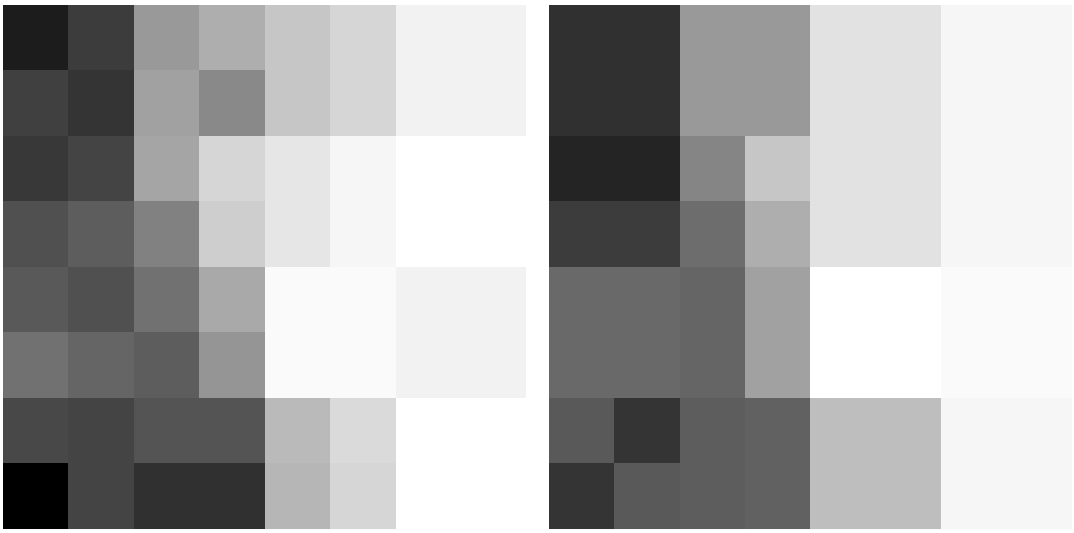

Figure 4: Compression with varying thresholds

Figure 4(a) shows the image corresponding to R; compare this with Figure 3(b). Figure 4(b) shows what happens if we repeat with $\epsilon = 50$; this clearly deviates even further from the original image. Admittedly, these approximations are not visually impressive, but this is mainly because of the scale; similar approximations of much higher resolution images (more pixels) are often quite acceptable, as we will demonstrate presently.

First we need a way to quantify the compression. The wavelet transformed matrix T—from which Figure 3(b) can be reconstructed—has 60 nonzero elements out of 64, whereas the doctored version D—from which Figure 4(b) was constructed—has a mere 29. In one sense that's a 50% saving, or a 2:1 compression ratio. The doctored matrix which results from taking $\epsilon = 50$ (not shown) only has 12 nonzero entries, so we could claim a 5:1 compression ratio in that case. In general, for a particular image matrix P and a fixed threshold ϵ, the *compression ratio* is understood to mean the ratio of the number of nonzero entries in the wavelet transformed matrix T to the number of nonzero entries in the doctored matrix D. Compression ratios of 10:1 or better give rise to matrices D which are sufficiently sparse that there is a significant saving when it comes to storage or transmission.

Now we consider more realistic, larger image matrices, such as the one in Figure 2(a). Let's take $\epsilon = 200$. We process the entire 128×128 matrix, which involves 7 rounds of row averaging and differencing followed by 7 rounds of column averaging and differencing, and then set to zero all entries which do not exceed 200 in absolute value. Finally we apply the inverse wavelet transform. We obtain the rather poor compressed image shown in Figure 5(a), and a compression ratio of 10:1.

Figure 5: 10:1 compressions of the Rosa Parks image

The second image, Figure 5(b), was obtained using a simple modification of the Haar wavelet transform, known as *normalization*. It too gives rise to a compression ratio of 10:1, and is obviously much closer to the original than Figure 5(a). Actually, we've seen a smaller version of this before, as the third of the images in Figure 1. Note the concentration of small blocks near the edges where light and dark regions meet, illustrating the adaptiveness of this process. The extreme blockiness of these images is due to the nature of averaging and differencing, which is equivalent to working with certain step functions (see [8] for details).

The averaging and differencing which is at the heart of the wavelet transform earlier involves repeated processing of certain pairs (a, b) to yield new pairs $(\frac{a+b}{2}, \frac{a-b}{2})$. In the *normalized wavelet transform*, we process each such pair (a, b) to yield $(\frac{a+b}{\sqrt{2}}, \frac{a-b}{\sqrt{2}})$ instead. Until we get to **§5**, this new form of averaging and differencing will likely seem unintuitive and unmotivated. Certainly, we have a reversible transform. We do not inflict any explicit examples of normalized wavelet transformed strings or matrices on the reader, but content ourselves with exhibiting the resulting compressed images. For a given compression ratio, normalization generally leads to better approximations, as we just saw in Figure 5(b).

Figure 6: 10:1 compressions of Nelson Mandela image

Figure 6 shows two wavelet compressed versions of the Nelson Mandela image, the second one normalized; again, both yield compression ratios of 10:1. Here we see the real potential of wavelet compression: Figure 6(b) is visually quite close to Figure 2(b), yet can be stored in a form which only requires 10% of the space that the original does.

4 Progressive Image Transmission

In the case of real-time image retrieval, such as grabbing images on the World Wide Web, the compression technique we have discussed allows for a type of progressive image transmission: When an image P is requested electronically, a wavelet-encoded version T is brought out of storage, and bits of information about it are sent "over the wires," starting with the overall average and the larger detail coefficients, and working down to the smallest detail coefficients. As this information is received by the user, it is used to display a reconstruction of P, starting with a very crude approximation of the image that, rapidly updated and refined, looks noticeably better as more wavelet coefficients are used. Eventually (assuming the user has deemed this picture worth waiting for) all of the detail coefficients will have been transmitted and a perfect copy of P displayed. If the user loses interest or patience along the way, she can easily halt the process.

For instance, the images of Rosa Parks in Figure 1, which use normalized wavelet compression with ratios of 50:1, 20:1, 10:1, and 1:1 (i.e., the orginal), respectively, could form stages in a progressive transmission, finishing up with a perfect image. Figure 7 shows another example using the Nelson Mandela image. Again, normalized wavelet compression is used here, this time with ratios of 200:1, 100:1, 50:1, 25:1, 10:1, and 2:1 respectively. The "half Nelson" in the last image is such a good approximation that it's almost indistinguishable from the "whole Nelson" in Figure 2(b).

Figure 7: The ever progressive Nelson Mandela

The Netscape browser, which many people use to access the World Wide Web, currently makes use of a form of JPEG image compression for progressive transmission (JPEG is one of the standard image file formats in use today). Wavelet-based software for this purpose, using a generalization of the Haar wavelet transform (see **Closing Remarks**) is already available. For instance, Summus, a software company based in South Carolina (URL: http://www.summus.com/index.html), has a product called *The Wavelet Image Netscape Plugin* on the market.

5 A Linear Algebra Approach to Wavelet Transforms

We demonstrate how matrix multiplication can be used to effect averaging and differencing. This, in turn, leads to a matrix multiplication version of the Haar wavelet transform. We provide enough details to allow the curious reader to use a standard computer algebra package, such as *Matlab*, to reproduce our results and pictures. (Matrix multiplication is not necessarily the most efficient approach here; for large data sets there are much better ways to effect the wavelet transforms.)

Our first wavelet transform took us from (576,704,1152,1280,1344,1472,1536,1536) to (1200,-272,-288,-64,-64,-64,-64,0) in three steps. The first step involved averaging and differencing four pairs of numbers; this can be expressed as the single vector equation

$$(640,1216,1408,1536,\text{-}64,\text{-}64,\text{-}64,0) = (576,704,1152,1280,1344,1472,1536,1536)\ A_1,$$

where A_1 denotes the matrix

$$\begin{pmatrix} \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{2} & 0 & 0 & 0 & -\frac{1}{2} & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 & 0 & -\frac{1}{2} & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 \\ 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & -\frac{1}{2} & 0 \\ 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} \\ 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & -\frac{1}{2} \end{pmatrix}.$$

The transition from (640,1216,1408,1536,-64,-64,-64,0) to (928,1472,-288,-64,-64,-64,-64,0) to (1200,-272,-288,-64,-64,-64, -64,0) is equivalant to the two vector equations

$$(928,1472,\text{-}288,\text{-}64,\text{-}64,\text{-}64,\text{-}64,0) = (640,1216,1408,1536,\text{-}64,\text{-}64,\text{-}64,0)\ A_2,$$

$$(1200,\text{-}272,\text{-}288,\text{-}64,\text{-}64,\text{-}64,\text{-}64,0) = (928,1472,\text{-}288,\text{-}64,\text{-}64,\text{-}64,\text{-}64,0)\ A_3.$$

where A_2, and A_3, respectively, denote the matrices:

$$\begin{pmatrix} \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & 0 & -\frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & -\frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \quad \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & -\frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

Thus, the combined effect of the Haar wavelet transform can be achieved with the single equation:

$$(1200,\text{-}272,\text{-}288,\text{-}64,\text{-}64,\text{-}64,\text{-}64,0) = (576,704,1152,1280,1344,1472,1536,1536)\ W,$$

where $W = A_1A_2A_3$. So it all boils down to linear algebra! Since the columns of the A_i's are evidently orthogonal to each other with respect to the standard dot product, each of these matrices is invertible. The inverses are even easy to write down—after all, they simply reverse the three averaging and differencing steps. In any case, we can recover the original string from the transformed version by means of the inverse Haar wavelet transform:

$$(576,704,1152,1280,1344,1472,1536,1536) = (1200,\text{-}272,\text{-}288,\text{-}64,\text{-}64,\text{-}64,\text{-}64,0)\ W^{-1},$$

where $W^{-1} = A_3^{-1}A_2^{-1}A_1^{-1}$. It is a routine matter to construct the corresponding $2^r \times 2^r$ matrices $A_1, A_2, \ldots, A_r$ needed to work with strings of length 2^r, and to write down the corresponding equations. As mentioned earlier, there is no loss of generality in assuming that each string's length is a power of 2.

For image matrices, we do the same row transformations to each row, followed by corresponding column transformations. The beauty of the string transform approach is that the equations relating the "before" and "after" strings are valid applied to an image matrix and its row-transformed form. If P is a $2^r \times 2^r$ image matrix, then the equations $Q = PW$ and

$P = QW^{-1}$ express the relationships between P and its row-transformed image Q, where $W = A_1 A_2 \dots A_r$. To handle column transformations, we repeat the steps above with a few transposes (denoted by $'$) thrown in. Putting everything together gives the following equations, which express the relationship between the original P and the *row-and-column-transformed* image T:

$$T = ((PW)'W)' = W'PW \quad \text{and} \quad P = ((T')W^{-1})'W^{-1} = (W^{-1})'TW^{-1}.$$

The normalization modification mentioned earlier is not only easy to implement in this new setting, but we can get also a glimpse of why it might be desirable. First, to do it we simply replace all of the $\pm\frac{1}{2}$'s in the matrices A_j with $\pm\frac{1}{\sqrt{2}}$'s. The columns of each matrix A_j then form an orthonormal set.[1] Consequently the same is true of the matrix W, which speeds up the inverse wavelet transform, since the matrix inverses are simply transposes. There is more than mere speed at stake here: as we have already seen, such normalization tends to lead to compressed images that are more acceptable to the human eye.

In matrix terms, the image compression scheme works like this: Start with P, and compute $T = W'PW$, which (we hope) will be somewhat sparse. Choose a threshold value ϵ, and replace by zero any entries of T whose absolute value is less than or equal to ϵ. Denote the resulting doctored matrix by D; this is sparse by design, and thus easier to store and transmit than P. To reconstruct an image from D, compute $R = (W^{-1})'DW^{-1}$.

Lossless compression is the case where $D = T$ (e.g., if $\epsilon = 0$) so that $R = P$. Otherwise we have *lossy compression*, in which case the goal is to pick ϵ carefully, so as to balance the conflicting requirements of storage (the more zeros in D, the better) and visual acceptability of the reconstruction R.

6 Student Research

Undergraduate research may be directed in the area described in this survey. Familiarity with linear algebra and calculus helps, and some access to a software package such as *Matlab* is desirable. Students seem to enjoy seeing how the mathematics they have learned can be applied to such a practical and multimedia-friendly project. They have also presented their research at numerous local and mational meetings.

At Spelman College, Tamara Pearson and Jimmitria Ford spent time on the topic during the summer of 1995. Ms. Pearson, a mathematics major and a computer science minor with a strong interest in graphics, has subsequently graduated and moved on to graduate school in computer science at the University of Florida at Gainesville. Ms. Ford's interests include image processing, and she is currently pursuing a degree in environmental engineering at Georgia Institute of Technology.

During each subsequent semester, several other mathematics majors have continued to explore this topic, for instance striving to come up with more efficient programs to implement the compression discussed here. At the time of writing, juniors Camille Daniel and Latoria Thomas are exploring the use of more advanced wavelet transforms, using spline wavelets, as well as the extension of the results to color images.

7 Closing Remarks

The wavelet transform we have discussed is the simplest, and crudest, member of a large class of possibilities. These Haar methods have been known and used in image processing for many years [9]. As we have seen, some quite acceptable results can be achieved with this modest little transform.

The Haar wavelet transform is usually presented in a more abstract mathematical setting using special functions called *Haar wavelets* (this approach is taken in [8]). In the mid and late 1980s some far-reaching generalizations of these wavelets were discovered [5], [4]. The corresponding transforms can often be implemented using matrix multiplication, as in the last section, where the matrices A_i have more numerous and more exotic entries. These more sophisticated wavelets produce smoother, more satisfactory compressed images than the ones that we obtained [10], [6].

Wavelets provide an alternative to classical Fourier methods for both one- and multi-dimensional data analysis and synthesis, and have numerous applications both within mathematics (e.g., to partial differential operators) and in areas as diverse as physics, seismology, medical imaging, digital image processing, signal processing, and computer graphics and video. Unlike their Fourier cousins, wavelet methods make no assumptions concerning periodicity of the data at hand. As a result, wavelets are particularly suitable for studying data exhibiting sharp changes or even discontinuities.

Good introductions to the mathematics and applications of wavelets can be found in Gilbert Strang's survey article [11] and in Barbara Burke Hubbard's remarkable book *The World According To Wavelets* [7]. Books covering applications in some detail include [2] and [1]. For an account of a recent adoption of wavelets as a standard for image compression, see [3]. There are also wavelet applications to audio and speech signals [12], and to partial differential operators and equations

[1] So that A_j is "orthogonal" according to a popular but unfortunate naming convention.

[2]. An excellent World Wide Web resource for wavelet matters is The Wavelet Digest `http://www.wavelet.org/wavelet` (email: `help@wavelet.org`). The *M-files* (Matlab programs) employed to produce all of our pictures, as well the author's paper [8], are available from `http://www.spelman.edu/~colm`.

Acknowledgements. This research was funded in part by the W.K. Kellogg Foundation, through Spelman College's Center for Scientific Applications of Mathematics (CSAM). The author would like to thank CSAM Director Sylvia Bozeman, who first stimulated his interest in wavelets. Thanks also to Tony DeRose and colleagues at the University of Washington for showing the author (and the world) that wavelet basics can be understood and appreciated without a solid background in Fourier methods; and to Rhonda Hughes of Bryn Mawr College for helpful conversations and clarifications.

References

[1] Aldroubi, Akram and Unser, Michael (editors), *Wavelets in Medicine and Biology*, CRC Press, Boca Raton FL, 1996.

[2] Benedetto, John J. and Frazier, Michael (editors), *Wavelets; Mathematics and Applications*, CRC Press, Boca Raton FL, 1996.

[3] Brislawn, Christopher M., "Fingerprints go digital," *AMS Notices* **42**(1995), 1278–1283.

[4] Chui, Charles, *An Introduction to Wavelets*, Academic Press, San Diego CA, 1992.

[5] Daubechies, Ingrid, *Ten Lectures on Wavelets*, CBMS 61, SIAM Press, Philadelphia PA, 1992.

[6] Glassner, Andrew S., *Principles of Digital Image Synthesis*, Morgan Kaufmann, San Francisco CA, 1995.

[7] Hubbard, Barbara Burke, *The World According to Wavelets*, A.K. Peters, Wellesley MA, 1996.

[8] Mulcahy, Colm, "Plotting and Scheming with Wavelets," *Mathematics Magazine* **69**, 5, (1996), 323–343.

[9] Pratt, William, *Digital Image Processing*, Wiley-Interscience, New York NY, 1978.

[10] Stollnitz, Eric, DeRose, Tony, and Salesin, David, *Wavelets for Computer Graphics*, Morgan Kaufmann, San Francisco CA, 1996.

[11] Strang, Gilbert, "Wavelets," *American Scientist* **82**(1994), 250–255.

[12] Wickerhauser, Mladen Victor, *Adapted Wavelet Analysis from Theory to Software*, A.K. Peters, Wellesley MA, 1994.