Applications of Wavelets to Image Processing and Matrix Multiplication

Daniel Beatty

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Questions that drive computational science include how fast can the answer be computed? How accurate is the answer? How stable is the method for obtaining the answer? In this thesis, these questions are investigated in the context to matrix multiplication. Of course there are already conventional algorithms to compute matrix multiplication. This thesis contributes a simple analysis of how the wavelet operator can be applied to matrix multiplication.

The two most desired qualities to improve compatibility in operation on matrices are a high degree of sparseness and a value close to unity of the condition number. Sparseness for a matrix is a measure of the degree to which the majority of the elements in such a matrix are zero. The condition number indicates how accurately an operation will be performed on the system. Wavelets contribute to numerical methods by providing a stable preconditioning technique which produces a more sparse and better conditioned than the original matrix. Various forms of the wavelet transform can be used when applying wavelets as a preconditioning tool, and so the sparseness attribute is studied in this thesis.

Matrix multiplication has applications in simulations, computer vision, and almost all areas of computational science. Classic matrix multiplication has a computational complexity of order N^3 operation, which makes it costly in the number of instructions that are needed to carry out the operation. Faster matrix multiplication techniques depend on the matrix being sparse to reduce the number elements that need to be multiplied. As shown in the results section, there exists various levels of acceptable sparseness that are generated by the wavelet transform.

For matrix multiplication, wavelets provide a sparse-condition generating method that transforms a dense matrix into a sparse one. Thus sparse matrix multiplication can be applied more effectively to general class of matrices. Wavelet preconditioning may or may not help for matrices that are already sparse.

In this thesis, an overview is provided to define wavelets and how to apply them. This overview uses image processing to demonstrate the qualities of a two-dimensional wavelet transform. The next chapter describes the wavelet matrix multiplication procedure. Chapter 5 demonstrates matrix multiplication in the wavelet domain and shows the results of different threshold levels on the fidelity of the resulting product. Finally, conclusions are presented.

CHAPTER 2

OVERVIEW ON WAVELETS

Wavelets evolved from atomic function theory where they were developed as basic atoms or building blocks of all functions. This chapter provides an overview of wavelets. This is done first by providing the standard definitions and concepts of the wavelet basis function, the wavelet transform and multiresolution representation. Then a numerical example is utilized to demonstrate the concepts. The next section details the numerical implementation of the wavelet transform. The last section represents the 2D wavelet transform.

2.1 The Wavelet Basis Function

This section provides the basic properties of a wavelet function. It first describes some general properties of functions. Then it presents the translation and dilation properties which can be used to build an entire bases. Finally, it presents the wavelet function itself.

2.1.1 Function Properties

There are a number of different properties that can be used to classify functions. These include integrability, symmetry, compactness and orthogonality. The properties will be introduced now and used in the rest of the chapter.

A function, $f(\cdot)$, is *square integrable* if its L_2 norm is finite. This can be expressed by

$$f(x) \in L_2(\mathbb{R}) \quad \text{if} \quad \int\limits_{-\infty}^{\infty} |f(x)|^2 \, dx < \infty.$$

A one-dimensional function is *symmetric* about the y-axis if

$$f(x) = f(-x)$$

and it is *antisymmetric* if

$$f(x) = -f(-x).$$

The term symmetric is synonymous with "even function." Likewise, antisymmetric is synonymous with "odd functions."

A set is considered *compact* in the n-dimensional real space, $\mathbb{R}^n$ if it is both closed and bounded. A function has *compact support* if it is zero outside of a compact set. This implies that there exists n-dimensional sphere, S^n, where

$$f(x) = 0 \qquad \forall x \notin S^n.$$

Orthogonality is another concept necessary for this thesis. Two different basis functions are *orthogonal* if their inner product is zero. The inner product of two functions, f and g, can be represented by

$$\langle f, g \rangle = \int_{-\infty}^{\infty} f(x)g(x)\ dx.$$

They are *orthonormal* if they are both orthogonal and have a norm of one.

2.1.2 Translation and Dilation

Translation shifts the basis function along the variable axis. The translation of a function, f, can be represented by f_k where

$$f_k(t) = f(t - k).$$

Notice that translation does not alter the shape of the function, only the position of the function along the number space of t [19].

Dilation (also called contraction) transforms a one-dimensional function in width, and the output in height. The dilation of a function, f, can be presented by f_j where

$$f_\alpha(t) = f(\alpha t).$$

If $\alpha < 1$ then the width of the function is increased. Otherwise it is decreased.

Often, a scaling value is associated with the dilation operation. An arbitrary form of this scaling is defined with s as follows:

$$f_s(t) = sf(t)$$

If $s > 1$ then the height of the function is increased. Likewise, when $s < 1$ the height is decreased. Often the scaling component is included in either translation, dilation or both. In orthonormal-translation-dilation, there is a special case of notation as described next.

These definitions can be combined into the translation-dilation representation upon dyadic intervals. A function f can be expressed by a dilation of j and a translation of k through

$$f_{j,k}(t) = 2^{j/2} f(2^j t - k) \qquad \forall j, k \in \mathbb{Z}.$$

There is a special case for including the scaling factor of $2^{j/2}$. The scaling value $2^{j/2}$ is applied to force a dyadic orthonormal condition. A function is said to have the orthonormal translation-dilation property when for two translation-dilations, $f_{j,k}$ and $f_{l,m}$, the result can be expressed by [7]

$$\langle f_{j,k}, f_{l,m} \rangle = \delta_{j,l} \delta_{k,m}.$$

2.1.3 The Wavelet Basis Function

The two mandatory properties of a wavelet basis function are:

- it must square integrable, and
- must have a zero average, i.e.:

$$\int_{-\infty}^{+\infty} \psi(x)\ dx = 0.$$

In addition, the class of orthogonal wavelet basis functions are restricted to satisfy the orthonormal translation-dilation property. This restriction no longer applies to all wavelets by the lifting scheme [8]. An example of a strict definition of a wavelet is given by Charles Chui [7] (page 61):

> "A function $\psi \in L_2(\mathbb{R})$ is called an orthonormal wavelet if the family $\{\psi_{j,k}\}$ defined
>
> $$\psi_{j,k}(x) = 2^{j/2}\psi(2^j x - k) \forall j, k \in \mathbb{Z}$$
>
> is an orthonormal basis of $L_2(\mathbb{R})$ where $\langle \psi_{j,k}, \psi_{l,m} \rangle = \delta_{j,l}\delta_{k,m}$, $\forall j, k, l, m \in \mathbb{Z}$ and every $f \in L_2(\mathbb{R})$ can be written as
>
> $$f(x) = \sum_{j,k=-\infty}^{\infty} c_{j,k}\psi_{j,k}(x)$$

where the series convergences and is in $L_2(\mathbb{R})$ such that

$$\lim_{M_1,M_2,N_1,N_2} ||f - \sum_{j=-M_2}^{N_2} \sum_{k=-M_1}^{N_1} c_{j,k}\psi_{j,k}|| = 0$$

The simplest example of orthonormal wavelets is the Haar Transform."

There is more than one wavelet basis function. Additional classifications used include the symmetry and compactness [12]. The Haar Wavelet Basis function actually fulfills the strictest definition of a wavelet basis function, and has additional properties. The Haar Wavelet Basis Function has compact support. It is also symmetric. Furthermore, it has been stated by Walker[18], Gregory Beylkin [5, 4, 6] and Chui [7] that the Haar Wavelet Basis Function is the only wavelet basis function in $L_2(\mathbb{R})$ that is orthogonal, compact, and symmetric. The Haar wavelet basis function is defined below.

$$\psi(x) = \begin{cases} 1 & 0 \le x < \frac{1}{2} \\ -1 & \frac{1}{2} \le x < 1 \\ 0 & otherwise \end{cases} \tag{2.1}$$

2.2 The Wavelet Transform

The wavelet transform is comprised of two parts, an average sample and a difference sample. This is expressed by first looking at pairs of wavelets and then looking at the actual representation.

2.2.1 Wavelet Pairs: The Averaging Basis Function

The Wavelet Basis Function section defined the Haar Wavelet Basis Function defined the Haar Wavelet Basis function, in equation 2.1. However, there exists a concept of a wavelet pair. These pairs exist as averaging and differencing basis. The wavelet basis function and differencing basis are synonymous. The averaging basis concept was derived from classic multi-resolution which is described in section 2.3.

This section only defines the wavelet basis pair in terms of a wavelet averaging basis and shows an example with the Haar Averaging Basis Function. A wavelet pair is a set of two basis functions containing one wavelet basis function and one averaging basis function which meet the following criteria:

- both must be square integrable,
- both must satisfy the orthonormal dyadic translation-dilation property, and
- the wavelet basis function must be orthogonal with the average basis function.

The simplest form of the averaging filter is the Haar Averaging Filter, and it is a pair to the Haar Wavelet Basis Function. Like the Haar Wavelet Basis Function, the Haar Averaging Filter also satisfies the dyadic-orthogonal translation property and is in $L_2(\mathbb{R})$ The mother [18, 20, 7, 17, 10] function for the Haar Averaging Filter is defined by

$$\phi(x) = \begin{cases} 1 & 0 \leq x < 1 \\ 0 & otherwise. \end{cases}$$

2.2.2 Transform Representation

This derivation uses the wavelet pair to define the wavelet transform. Any wavelet transform uses a satisfactory wavelet pair to transform an array into two halves which constitute the average filter sub-array and the difference filtered sub-array. This definition commonly refers to the average filtered sub-array as the average terms and difference terms, respectively. The concatenated array of average and difference terms constitutes a similar array to the original array, and the same properties as similar vectors. Also, for every wavelet transform there is a straightforward method to restore the original array from the wavelet transformed array called the inverse wavelet transform.

One form of the wavelet transform is the integral wavelet transform (IWT) described by Chui[7]. Another two are the discrete wavelet transform (DWT) and the fast wavelet transform (FWT)[18, 9]. The convolution wavelet transform (CWT) is a general form of the FWT. In the case of the CWT, any proper wavelet pairs can be used to generate corresponding average and difference terms. Application of the CWT is described in section 2.5.

Convolution of the wavelet pairs with an array form the starting point to the convolution wavelet transform. The following constitute the steps of the CWT:

1. Convolute the original array S with the wavelet pair. The results of this are the average filtered array, A, and the difference filtered array, D.

2. Selectively filter every other element of A and D into new arrays half the size of S. The results map $A \rightarrow A'$ and $D \rightarrow D'$ respectively, where A' and D' are the selectively filtered average filter array, and selectively filtered difference filtered array, respectively.

3. Concatenate A' and D' to form $W(S) = (A'|D')$, which is the wavelet transform of S.

In the CWT used in this thesis, a modified version of the Haar Wavelet Basis Pair is used. In particular, $\psi_{1,k}$ and its average basis function $\phi_{1,k}$ is the wavelet pair. To assist the reader, this pair is defined for $x \in \mathbb{Z}$ in equations 2.3 and 2.4. This basis pair is derived from the orthogonal dilation-translation equation for $j = 1$, which is stated in equation 2.2.

$$\psi_{j,k}(x) = 2^{j/2}\psi(2^j x - k)\forall j, k \in \mathbb{Z} \tag{2.2}$$

$$\phi_{1,0}(x) = 2^{1/2}\phi(2^1 x) = \sqrt{\frac{1}{2}}\begin{cases} 1 & x = 0 \ and \ x = 1 \\ 0 & otherwise \end{cases} \tag{2.3}$$

$$\psi_{1,0}(x) = 2^{1/2}\psi(2^1 x) = \sqrt{\frac{1}{2}}\begin{cases} 1 & x = 0 \\ -1 & x = 1 \\ 0 & otherwise \end{cases} \tag{2.4}$$

2.3 Multi-resolution Representation

In one-dimensional, multi-resolution analysis provides a method to measure averages, differences of the original array or signal and of the sub-arrays generated in application of multi-resolution. In this section, techniques of multi-resolution are defined in terms of one-dimensional wavelet transforms. two-dimensional versions covered in section 2.6. Three methods for decomposing the signal are considered — multiresolution analysis (MRA), multiresolution expansion (MRE) and the ψ^n expansion.

2.3.1 MRA

MRA schemes generates averaging estimates of some signal such that the average representation is smaller than the original by some integer amount. MRA reapplies this averaging process until the process produces a small enough size for the analysis being conducted. MRA may maintain estimate correction factors for the purpose of translating an averaged array to the next larger average array to recover that particular function. The elements of average functions are averaging terms. Likewise, the elements of the estimate correction factor function are called differencing terms. To acquire each estimate by wavelets, the wavelet averaging basis is used. The wavelet basis function happens to be the basis function for the differencing function

There are simple ways to visualize this concept. A classic one dimensional form method is to consider a triangle. In this triangle, layers of functions are stacked up from the base to the apex. The original function is always placed on the bottom.

Subsequent average terms are placed between the base and the apex. For every level between the apex and the base, there is an averaging method to map that level to the adjacent level closer to the apex. Also, there exist a set of difference elements such that when the average is expanded it can be mapped to the adjacent level toward the base. The apex has no adjacent level. The base has no difference elements to map to a larger level.

The way to consider MRA with one-dimensional wavelet transforms is with a binary tree. At the root of the tree, the entire original array is represented. Two branches exist on this tree, the average and difference branch. Each node in the tree represents a sub-array of the original array. Branches are generated on a node if and only if a wavelet transform is performed on the node. In case a wavelet transform is performed, both an average and difference branch are generated. Otherwise the node is a leaf. The following are rules for the one-dimensional wavelet transform binary tree in MRA:

- Transforms are only applied to leaves.
- Transforms are only allowed on the root or average leaves.
- If a node is not a leaf then a wavelet transformation has already been performed and is not permitted to be reapplied.
- If the leaf is a difference term, then a transformation is not allowed.

- The maximum height of this tree is $n = \lceil \log_2 |S| \rceil$ where $|S|$ is the cardinality of the signal S.

2.3.2 MRE

MRE extends this idea by considering the difference leaves, also. The rules for one-dimensional wavelet transform binary tree in MRA are as follows:

- Transforms are only applied to leaves.
- If a node is not a leaf then a wavelet transformation has already been performed and is not permitted to be reapplied.
- The maximum height of this tree is $n = \lceil \log_2 |S| \rceil$ where $|S|$ is the cardinality of the signal S.

Like in MRA, there is an analogues to time and frequency signal representation. Every sub-array represented at each node with in the wavelet transform binary tree represents activity in the time domain within a certain octave and note of frequencies. That octave's frequency range is specified by its position within the array defined here:

$$\left[\frac{\pi x_1}{2 \cdot |S|}, \frac{\pi(x_2 + 1)}{2 \cdot |S|}\right]$$

where x_1 is the starting index of the sub-array and x_2 is the ending index of the sub-array.

2.3.3 ψ^n Expansion

There is another expansion similar to MRE which places its sections in a different arrangement. This form is called the ψ^n expansion. It is a rather simple expansion of the wavelet transform. Each resolution, reapplies the wavelet transform to the whole array again. While it is simple to conceive and implement, the results are little more complicated to analyze. A mapping procedure does exist for mapping the ψ^n expansion to the MRE and vice versa.

2.4 Numerical Example

To illustrate the concepts of the wavelet transform, four simple numerical examples are provided. First example shows the application of the CWT. The second one utilities MRA and is shown in Figure 2.1. The third one utilizes MRE and is demonstrated in Figure 2.2. In these examples, the wavelet transform is applied to the source array f defined in equation 2.5. The CWT applied to source array f is the Haar Convolution Wavelet Transform in the discrete domain. More specifically, the Haar Convolution Wavelet Transform is defined for the Haar Wavelet Basis Pair $\psi_{1,0}$ as in equation 2.4 and $\phi_{1,0}$ in equation 2.3. Lastly, the Haar Convolution Wavelet Transform in this case is the discrete form defined in terms of discrete convolution defined in equation 2.6 in terms of the two operand arrays f and g, and the result array h.

$$f(i) = \{5, 5, 5, 3, 4, 6, 8, 5\} \tag{2.5}$$

$$h_i = \langle f(l), g(-l+i) \rangle \tag{2.6}$$

2.4.1 Wavelet Transform Example

This first example shows the CWT applied to the source array f. The first step is to acquire A and D defined:

$$A = S * \phi_1$$

$$D = S * \psi_1$$

where ϕ_1 and ψ_1 are a dilated Haar Wavelet Basis Pair defined in equations 2.3 and 2.4. Each value A_i and D_i computed by these convolutions are simply the inner products of ϕ_1 and ψ_1 translated by i and are shown in equations 2.7 and 2.8.

$$A_i = \langle S, \phi_{1,i} \rangle \tag{2.7}$$

$$D_i = \langle S, \psi_{1,i} \rangle \tag{2.8}$$

Each value of A and D are computed out in equations 2.9 and 2.10.

$$S * \phi_1 = \begin{pmatrix} A_0 = & \langle S, \phi_{1,0} \rangle = & \frac{5}{\sqrt{2}} \\ A_1 = & \langle S, \phi_{1,1} \rangle = & 5\sqrt{2} \\ A_2 = & \langle S, \phi_{1,2} \rangle = & 5\sqrt{2} \\ A_3 = & \langle S, \phi_{1,3} \rangle = & 4\sqrt{2} \\ A_4 = & \langle S, \phi_{1,4} \rangle = & \frac{7}{\sqrt{2}} \\ A_5 = & \langle S, \phi_{1,5} \rangle = & 5\sqrt{2} \\ A_6 = & \langle S, \phi_{1,6} \rangle = & 7\sqrt{2} \\ A_7 = & \langle S, \phi_{1,7} \rangle = & \frac{13}{\sqrt{2}} \end{pmatrix} \tag{2.9}$$

$$S * \psi_1 = \begin{pmatrix} D_0 = & \langle S, \psi_{1,0} \rangle = & \frac{5}{\sqrt{2}} \\ D_1 = & \langle S, \psi_{1,1} \rangle = & 0 \\ D_2 = & \langle S, \psi_{1,2} \rangle = & 0 \\ D_3 = & \langle S, \psi_{1,3} \rangle = & -\sqrt{2} \\ D_4 = & \langle S, \psi_{1,4} \rangle = & \sqrt{\frac{1}{2}} \\ D_5 = & \langle S, \psi_{1,5} \rangle = & \sqrt{2} \\ D_6 = & \langle S, \psi_{1,6} \rangle = & \sqrt{2} \\ D_7 = & \langle S, \psi_{1,7} \rangle = & \frac{-3}{\sqrt{2}} \end{pmatrix} \tag{2.10}$$

Next, the selection filter removes all of the even indexed elements to produce A' and D'. Last, A' and D' are concatenated to produce equation 2.4.1, which is the

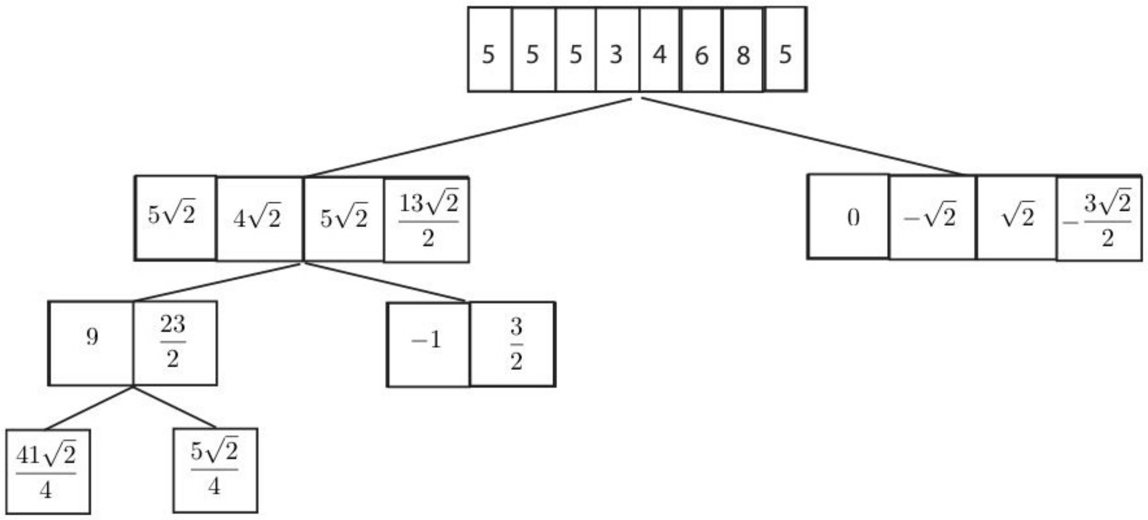
5 5 5 3 4 6 8 5
$5\sqrt{2}$ $4\sqrt{2}$ $5\sqrt{2}$ $\frac{13\sqrt{2}}{2}$
0 $-\sqrt{2}$ $\sqrt{2}$ $-\frac{3\sqrt{2}}{2}$
9 $\frac{23}{2}$
-1 $\frac{3}{2}$
$\frac{41\sqrt{2}}{4}$
$\frac{5\sqrt{2}}{4}$

Figure 2.1: The Haar Transform performed a sample function show each step of the transform in multi-resolution analysis

wavelet transform of S.

$$W(S(x)) = \{5\sqrt{2}, 4\sqrt{2}, 5\sqrt{2}, \frac{13}{\sqrt{2}}, 0, -\sqrt{2}, \sqrt{2}, -\frac{3}{\sqrt{2}}\}$$

2.4.2 MRA Example

MRA starts with the original array. The wavelet transform is applied to the array. After the first application of the wavelet transform, an average and difference array now exist. This is shown in figure 2.1 as the first two children in the wavelet binary tree.

For MRA, the analysis is continued on the average branch. The difference branch is unchanged. This procedure is repeated on the average branch. The stopping point is determined by the size of array. The finite limit, n, is

$$n = \lceil \log_2 |S| \rceil \tag{2.11}$$

where $|S|$ is the size of the array.

Once completed, the arrays are joined into an array which is the same size as the original. The energy is generally concentrated at the beginning of the array. Also, the energy of the array tends to be ordered from strongest to weakest. Terms representing change are kept as pairs to each section.

Notice that the fine difference terms are left alone, and the dynamics of the function in the time domain are accentuated. Likewise the lower frequency components are separated allowing a closer analysis in both position and frequency.

2.4.3 MRE Example

Like MRA, MRE starts with an original array, and applies a wavelet transform to that array. The decomposition is also represented by binary tree, with an average and difference branch. The limit and height of the tree, n, remains the same as equation 2.11.

What is not the same is how the wavelet transform is reapplied. In addition to being applied on the average branch again, but the wavelet transform is also applied to the difference branch also. An example is provided in Figure 2.2. The sub arrays are reinserted into the array in order from the left branch to the right branch.

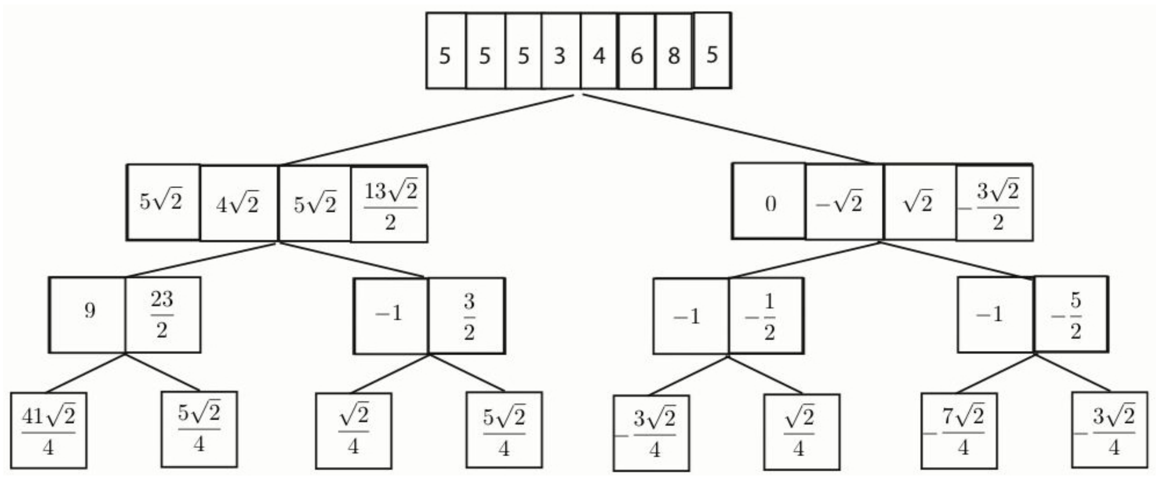
5 5 5 3 4 6 8 5
$5\sqrt{2}$ $4\sqrt{2}$ $5\sqrt{2}$ $\frac{13\sqrt{2}}{2}$
0 $-\sqrt{2}$ $\sqrt{2}$ $-\frac{3\sqrt{2}}{2}$
9 $\frac{23}{2}$
-1 $\frac{3}{2}$
-1 $-\frac{1}{2}$
-1 $-\frac{5}{2}$
$\frac{41\sqrt{2}}{4}$ $\frac{5\sqrt{2}}{4}$ $\frac{\sqrt{2}}{4}$ $\frac{5\sqrt{2}}{4}$ $-\frac{3\sqrt{2}}{4}$ $\frac{\sqrt{2}}{4}$ $-\frac{7\sqrt{2}}{4}$ $-\frac{3\sqrt{2}}{4}$

Figure 2.2: The Haar Transform performed a sample function show each step of the transform in multi-resolution expansion

In the time and frequency analogues, each transformation filters the array with high pass and low pass filter. The frequency width of these sub arrays is proportional to length of the original array. The center frequency is relative to the position of the sub-array within the array. The lowest frequency components are on the average side of the array, and the highest frequency components are on the difference section of the array. One special case exists for arrays of length 2^n. If the array is of length 2^n, then the full MRE yields entire frequency domain. In case that the array is of odd size larger than one, then the array must be padded and then normal decomposition can continue.

2.4.4 ψ^n Expansion

One other form of the multi-resolution expansion that can be used is almost trivial by its nature, ψ^n expansion. This particular form has the same branches as the MRE.

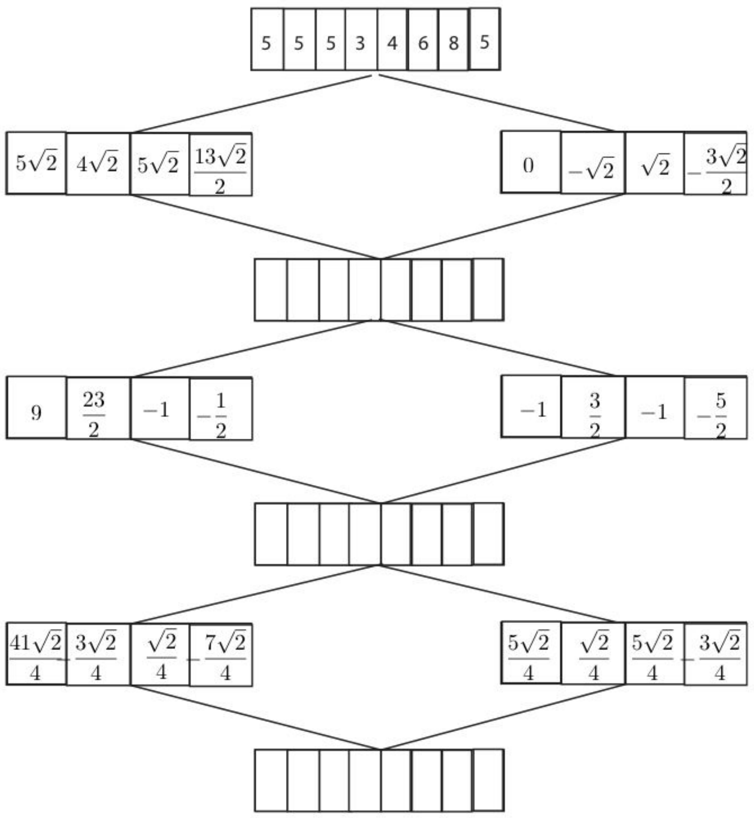
5 5 5 3 4 6 8 5
$5\sqrt{2}$ $4\sqrt{2}$ $5\sqrt{2}$ $\frac{13\sqrt{2}}{2}$
0 $-\sqrt{2}$ $\sqrt{2}$ $-\frac{3\sqrt{2}}{2}$
9 $\frac{23}{2}$ -1 $-\frac{1}{2}$
-1 $\frac{3}{2}$ -1 $-\frac{5}{2}$
$\frac{41\sqrt{2}}{4}$ $\frac{3\sqrt{2}}{4}$ $\frac{\sqrt{2}}{4}$ $-\frac{7\sqrt{2}}{4}$
$\frac{5\sqrt{2}}{4}$ $\frac{\sqrt{2}}{4}$ $\frac{5\sqrt{2}}{4}$ $-\frac{3\sqrt{2}}{4}$

Figure 2.3: The Haar Transform performed a sample function show each step of the transform in ψ^n expansion

However, the sub arrays are place back in the array in a different order. Figure 2.3 illustrates the difference in ordering from the ψ^n form and the MRE form. This form makes for a difficult analysis of time and frequency components, but contributes advantages linear operations.

2.5 Implementation of the Wavelet Transform

As stated before, the wavelet transform is defined in terms of average and difference components. A signal is taken and transformed into the two base objects.

Typically, the transform has the form $S \rightarrow (A'|D')$ where S is the original signal, A is the average component, D is the difference component and $(A'|D')$ the signal A' concatenated with D'. This transformation can be modeled with the convolution operator. Despite the decomposition of the signal into the two components, the original signal can be reconstructed.

Many mathematicians such as Walker [18], use a form that eliminates half of the values. Thus a form can be defined which has the same number of elements as the original. The rules for choosing the member elements are dependent on the wavelet filter choice. Another useful property of wavelets via convolution is the simplicity of the operation. The general case works for all. Such an algorithm requires one nested loop as seen in Algorithm 1.

Algorithm 1 Convolution of two signals $x(\cdot)$ and $h(\cdot)$.

```
$y_i = 0 \ \forall i \in (0, M-1)$
for $i = 0$ to $M-1$ do
  for $j = 0$ to $N-1$ do
    $n = i - j$
    if $(0 \leq n < M)$ then
      $y(i) += \ x(j) \cdot h(n)$
    end if
  end for
end for
```

This filter simply equates to the mathematical function

$$y(k) = (x * h)(k) = \sum_{l} x(l)h(k-l)$$

which is the convolution operation. It is obvious that the operation is $O(NM)$. For practical use, the filter is made smaller than the actual signal being analyzed. In some cases, the filter may be much smalled than the signal. Filter size matters in extracting features from the original signal.

To perform a wavelet transform via convolution, the signal is first convolved with the average operator and then with the difference operator.This can be represented by

$$A = S * \phi \qquad \text{and} \qquad D = S * \psi.$$

where ϕ is the mother wavelet and ψ is the difference wavelet.

2.6 The 2D Wavelet Transform

For the 2D Wavelet Transform, it has to be determined how to represent average and difference components. This can be done by creating one average components and three difference component - vertical, horizontal and diagonal. This can be used in a multiresolution fashion to provide several levels of decomposition.

A complete transform method returns a result matrix which is the same size as the source matrix. The result contains the four components. Each component resides on 4 corners of the matrix. Given a matrix B, the transform is to yield the following form:

$$B \Rightarrow \begin{pmatrix} H & D \\ A & V \end{pmatrix}$$

where A is the average component, H is the horizontal component, V is the vertical component, and D is the difference component. There is another form which is also used as an example:

$$B \Rightarrow \begin{pmatrix} A & V \\ H & D \end{pmatrix}.$$

The first version is simple in concept, but provides a few more possibilities for error and confusion. Regardless of the case, the four components have the following definitions:

1. Average component: produced by filtering the row vectors and the column vectors with the averaging filter.

2. Vertical Component: produced by applying the average filter to the column vectors and the difference filter to the row vectors.

3. Horizontal component: produced by applying the average filter to the row vectors and the difference to the column vectors.

4. Diagonal component: produced by applying the difference filter to both the row and column vectors.

2.6.1 Multi-Resolution in two-dimensions

Three methods are presented for decomposing 2D matrices — 2D MRE, 2D MRA and 2D ψ^n Expansion. The first of these methods for matrices is the 2D MRE. In this form, the wavelet transform is applied recursively to each of the sub-matrices.

The inverse of this MRE applies the inverse wavelet transform is applied to the lowest level adjacent sub-matrices.

The 2-D MRA is a special case of the 2-D MRE. The application of the wavelet transform is only applied to the average sub-matrix, only. This method is also called wavelet pyramids.

The ψ^n is rather more simple to implement, but is harder to analyze. In this case, the wavelet transform is simply reapplied to the whole matrix. The result of this method contains the same elements as the 2-D MRE (wavelet transform full decomposition). However, the arrangement of the elements is not the same. This arrangement will be used for matrix multiplication.

The method for visualizing 2D MRE and 2D MRA is similar to the one-dimensional version. Here a quadtree is used instead. The four branches represent the four sub-matrix types (average, vertical, horizontal, and diagonal). The rules for this tree is similar for the 1D wavelet transform binary tree.

The following are rules for the 2D wavelet transform quad tree in MRA.

- Transforms are only applied to leaves.
- Transforms are only allowed on the root or average leaves.
- If a node is not a leaf, then a wavelet transformation has already been performed and is not permitted to be reapplied.

- If the leaf is a vertical, diagonal or horizontal term, then a transformation is not allowed.
- The maximum height of this tree is $n = \min(\lceil N_r \rceil, \lceil N_c \rceil)$, where N_r is the number of rows in the matrix and N_c is the number of columns in the matrix.

MRE extends this idea by considering the difference leaves, also. The rules for the 2D wavelet transform quad tree in MRA are below.

- Transforms are only applied to leaves.
- If a node is not a leaf, then a wavelet transformation has already been performed and is not permitted to be reapplied.
- The maximum height of this tree is $n = \min(\lceil N_r \rceil, \lceil N_c \rceil)$, where N_r is the number of rows in the matrix and N_c is the number of columns in the matrix.

2.6.2 Methods of Implementation

Two methods of convolving a matrix are easily conceived. First is to use 1D wavelet. The other is to apply the convolution scheme straight to the matrix. Included in the wavelet experiment are both. Realistically, both can and do achieve the same result. However, the direct method achieves speed advantages by the lack of overhead. The direct method only stores a temporary vector resident in memory. Also, there are two fewer transfers per row and column.

2.6.2.1 1D to 2D Method

Both rows 1D and 2D and columns 1D and 2D transform are performed similarly. The obvious difference is the indexing of rows and columns.

Algorithm 2 Wavelet Row Transform

```
Require: Wavelet Transform, , and Wavelet Pair ($\psi$ and $\phi$)
Require: Matrix, $S \in \mathbb{R}^{N \times M}$
  for $i = 0$ to $N - 1$ do
    $R = S.getRow(i)$
    $S.row(i) = [R * \phi, R * \psi]$
  end for
```

This principle of this algorithm is simple. Only three intuitive steps are necessary per row or column. Two of these steps are array transfers (row/column transfer to an array). These arrays are fed into the 1D transforms.

However, the 1D wavelet transform itself includes a series of memory allocation and deallocation operations. Each memory call is at the minimum a system call.

2.6.2.2 Vector - Matrix Method

The principle of this algorithm is more complicated. All functionality, such as convolution, is built into the method. There are fewer calls and passing of structures to external functions to compute the transform.

This method has a few givens. The source matrix, the Haar average filter, and the Haar difference filter are given arguments. The result argument is the return argument. The transform signals sub-function row transforms and column transforms to perform the work.

The algorithm is as follows for the row transform (and is similar for the column transform):

Algorithm 3 Wavelet Transform: Vector - Matrix Method: Row Transform

```
Require: Wavelet Pair
Require: Temporary Vector $S$
Require: Matrix, $A \in \mathbb{R}^{M \times N}$
  for $i = 0$ to $M$ do
    load $S$ from $A_i$ where $A_i$ is the row vector at row $i$
    $S \xrightarrow{\psi_1} R$
    Load $R$ into result matrix $\alpha$ at $\alpha_i$
  end for
  Return $\alpha$
```

Algorithm 4 Wavelet Transform: Vector - Matrix Method: Column Transform

```
Require: Wavelet Pair
Require: Temporary Vector $S$
Require: Matrix, $A \in \mathbb{R}^{M \times N}$
  for $j = 0$ to $N$ do
    load $S$ from $A_j$ where $A_i$ is the column vector at column $j$
    $S \xrightarrow{\psi_1} R$
    Load $R$ into result matrix $\alpha$ at $\alpha_j$
  end for
  Return $\alpha$
```

To perform a wavelet transform with this method, the driving wavelet transform method calls on of the wavelet transform methods and feeds the results into the other. In the implementation used for this thesis, the row transform is called first, then its results α are fed into the column transform as the column transforms A. The results from the column transform are returned as the wavelet transform of the original matrix.

2.6.3 2-D MRA: Wavelet Pyramids

This version of the wavelet transform (multiresolution) uses private members of the class $(hA, hD, xD/yD, xA/yA)$. Both Haar filters are maintained this way. Also both row and column transforms have average and difference myVector classes for temporary storage. All of these members are allocated and destroyed by the wavelet transform method itself. The simplified algorithm of the row transform is shown in Algorithm 5, and the column transform is shown in Algorithm 6.

Algorithm 5 Wavelet Transform: Wavelet Pyramid Method: Row Transform

```
Require: Wavelet Pair $hA$ and $hD$ of length $w_l$
Require: Temporary Vector $S$
Require: Matrix, $A \in \mathbb{R}^{M \times N}$
Require: Limits of Rows and Columns to traverse: $M'$ and $N'$
Require: Temporary vectors $xA$ and $xD$ for row Average vector and row Difference
  Vector.
  Initialize vector $xA$ and $xD$
  for $i = 0$ to $M'$ do
    Initialize $xA$ and $xD$
    for $k = 0$ to $N'$ do
      for $l = 0$ to $w_l$ do
        $n = k - l$
        if $n \in [0, N']$ then
          $xA_k = A_{i,n} \cdot hA_l$
          $xD_k = A_{i,n} \cdot hD_l$
        end if
      end for
    end for
    Transfer to $\alpha$. $\alpha_i \leftarrow xA' | xD'$
  end for
  Return $\alpha$
```

Note: A_i names the row vectors and A_j names the column vectors, and $A_{i,j}$ is the element from the i^{th} row and j^{th} column.

Algorithm 6 Wavelet Transform: Wavelet Pyramid Method: Column Transform

```
Require: Wavelet Pair $hA$ and $hD$ of length $w_l$
Require: Temporary Vector $S$
Require: Matrix, $A \in \mathbb{R}^{M \times N}$
Require: Limits of Rows and Columns to traverse: $M'$ and $N'$
Require: Temporary vectors $yA$ and $yD$ for column Average vector and column
  Difference Vector.
  for $j = 0$ to $N'$ do
    Initialize $yA$ and $yD$
    for $k = 0$ to $M'$ do
      for $l = 0$ to $w_l$ do
        $n = k - l$
        if $n \in [0, M']$ then
          $yA_k = A_{n,j} \cdot hA_l$
          $yD_k = A_{n,j} \cdot hD_l$
        end if
      end for
    end for
    Transfer to $\alpha$. $\alpha_j \leftarrow yA'|yD'$
  end for
  Return $\alpha$
```

The computational cost of this algorithm is $c \cdot C_\psi$ where C_ψ is the cost of the wavelet transform at large, and n is the number of resolutions performed. The reason for this constant is that the size of the matrix being transformed shrinks by a factor of 2 each time. There may be additional overhead for the setup of the matrices, and transfer of matrix S to T, which is N^2. Since $C_\psi \propto O(N^2)$, and c is based on $\log_2(N)$, then this a cost of another constant multiplied by $O(\sum_{k=0}^{log_2(N)} \frac{1}{k^2} N^2))$.

2.6.4 2-D MRE via Quad Tree

There are two ways to implement the 2-D MRE and it depends on if other stopping conditions are desired. The one stopping condition of size can be handled

Algorithm 7 Wavelet Transform: Wavelet Pyramid Method: Driving Transform

```
Require: Wavelet Pair $hA$ and $hD$ of length $w_l$
Require: Matrix $A$ of size $M \times N$
Require: Number of resolutions, $r$
  Initialize matrix $\alpha$ to $M \times N$ and set equal to $A$
  Initialize matrix $\beta$ to $M \times N$ and set equal to zero.
  for $k = 0$ to $r$ do
    $M' = \frac{M}{2^k}$
    $N' = \frac{N}{2^k}$
    call row transform for matrix $\alpha$, with dimension limits $M'$ , $N'$ store in $\beta$
    call column transform for matrix $\beta$, with dimension limits $M'$ , $N'$ store in $\alpha$
  end for
  Return $\alpha$ as result
```

in a simple loop. To enforce an energy limit stop, then a means to stop any further transforms on that branch must be used. Two such means exist. One way is to mark that branch and do so to its children. Another way is to use a queue structure to designate nodes to be transform. If the node represents a stopping condition, simply do not enter that node into the queue.

In either case, the matrix information can be stored in this quad-tree. Either by referencing position within the working matrix, or by storing a matrix in the node itself. In this thesis, references were used for space efficiency. However, the storage of the matrices in the nodes is just as valid. Also a hybrid of only storing the final matrices in the leaves is acceptable as well.

As stated, a straightforward loop can acquire the full decomposition. The limit of the times required for is defined $v_l = 4^{r-1}$ where r is the height of the quad-tree. In this implementation, the quad tree is implemented in an array, and those properties are exploited in using a simple for loop to perform the 2-D MRE.

Algorithm 8 Wavelet Transform: MRE Quad Tree Loop

```
Require: Wavelet Transform, $\psi$, and Wavelet Pair
Require: Matrix, $S \in \mathbb{R}^{N \times M}$
  for $i = 0$ to $v_l$ do
    draw $S$ from quad-tree at index $i$
    $S \xrightarrow{\psi_r} T$
    $T \xrightarrow{\psi_c} X$
    Wavelet Split and Store $X$ in branch children.
  end for
```

Note on Algorithm 8 that the Wavelet Split and Store operation splits the matrix X into its four sub-matrices (average, horizontal, vertical, and diagonal) and stores them in their appropriate spot in the tree. Thus every-time that S is loaded a leaf branch of the tree is loaded to be transformed.

The queue based 2-D MRE with a quad-tree is a bit more complicated. The queue must be loaded with the root, then operation may be allowed to proceed. The end condition is when the queue is empty. Rules for loading the queue is must force the leaves on the last level to be ineligible for loading. Otherwise, the loop will never stop and segmentation fault is likely to arise. Energy rules and other arbitrary rules may also be imposed, but they are not as critical as the leaf limit rule. For this example, refer to Algorithm 9. The computational cost of this algorithm $n \cdot C_\psi$ where C_ψ is the cost of the wavelet transform at large, and n is the number of resolutions performed. In some special cases, the cost may be slightly to significantly less if additional termination rules are applied. The reduced size of the sub-matrices is canceled by quantity of sub-matrices to transform. There may be additional overhead

Algorithm 9 Wavelet Transform: MRE with queue controlled visits of the Quad Tree

```
Require: Wavelet Transform, MRE, and wavelet pair ($\psi$ and $\phi$)
Require: Matrix $S$
  Insert root node representing the whole matrix in the queue.
  while as long as queue is not empty do
    Get the next element from the queue, and load matrix $S$
    $S \xrightarrow{\psi_r} T$
    $T \xrightarrow{\psi_c} X$
    Check $X$ for terminating conditions
    if $X$ does not have a terminating condition store $X$'s four sections into the branch
      children of node $S$. Store theses children nodes in the queue in order of average,
      vertical, horizontal, and difference.
  end while
```

for the setup of the matrices. Since $C_\psi \propto O(N^2)$, and n is based on $\log_2(N)$ then this has a cost on the order of $N^2 \log_2(N)$ also.

2.6.5 2-D ψ^n Implementation

As stated the ψ^n expansion is simpler to implement, but produces a matrix that is harder to analyze for features. However, it is scientific merit since it is literally a wavelet transform of a wavelet transform. Also, the limits imposed on the MRA and MRE for tree height are not required, but are rather a good idea. A good practical limit would be for the resolution limit to be defined $n = \log_2 |S| - 1$ where $|S|$ is the size of the the matrix S. For this example, refer to Algorithm 10.

The computational cost of this algorithm $n \cdot C_\psi$ where C_ψ is the cost of the wavelet transform at large, and n is the number of resolutions performed. There may be additional overhead for the setup of the matrices, and transfer of matrix S to T, which is N^2. Since $C_\psi \propto O(N^2)$ then this has a cost on the order of $O(N^2 \log_2(N))$.

Algorithm 10 Wavelet Transform: MRE with queue controlled visits of the Quad Tree

```
Require: Wavelet Transform, MRE, and wavelet pair ($\psi$ and $\phi$)
Require: Matrix $S$
  Load $S$ into temporary matrix $T$
  for $i = 1$ to $n$ do
    $T \xrightarrow{\psi_c} X$
    $X \xrightarrow{\psi_r} T$
  end for
  return $T$ as the transformed matrix
```

CHAPTER 3

IMAGE PROCESSING EXAMPLE OF WAVELETS

This chapter is for showing graphically examples of the one-dimensional and two-dimensional wavelet transform. For one-dimensional wavelet transform, a simple sinusoid signal is used as the source input to measure correctness of the implementation. Likewise, a simple pictorial image is used as the test input to show correctness of these implementations of the two-dimensional wavelet transform.

3.1 Results — 1D Wavelet Transform

Testing of the 1D wavelet was performed on a sinusoidal wave form of 128 elements:

$$y(n) = 10\sin\left(\frac{n}{128}\right) - 5\sin\left(\frac{n}{64}\right) + 2\sin\left(\frac{3n}{128}\right) - \sin\left(\frac{n}{32}\right). \tag{3.1}$$

This function is the input function used for the one-dimensional implementations and is shown graphically in Figure 3.1. The first test of the 1D transform used the even elements of both convolutions to generate the wavelet transform. These even elements came from the over-complete form and naturally allow the potential to have complete information. However, in doing so, a fundamental flaw appears.

In order to evaluate the effectiveness of the wavelet transform three tests have been devised. First, energy equivalence is used to determine how much energy is retained in the transform from the original. The general shape is used on a the first

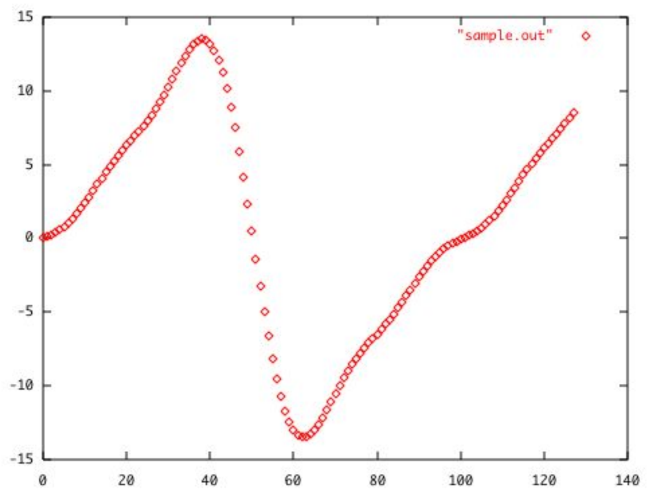

Figure 3.1: Sample function. The x-axis is the array index (index n). The y value is simple – the value $y(n)$.

resolution to test if the average signal has the same general shape as the original. Lastly, the inverse transform is used to recover the original signal. A comparison is made between the original and the recovered signal.

After one resolution, the transformed signal has the same energy as the original. This is good since it allows the original to be recovered from the transform. Also, the average component of the transform has the same shape as the original, which is good. However, the recovered signal is missing the last element. Refer to Figure 3.3. The secret is in which elements are used from the over-complete to make the complete. The over-complete form is defined from the average and difference components which are simply the result of convolution.

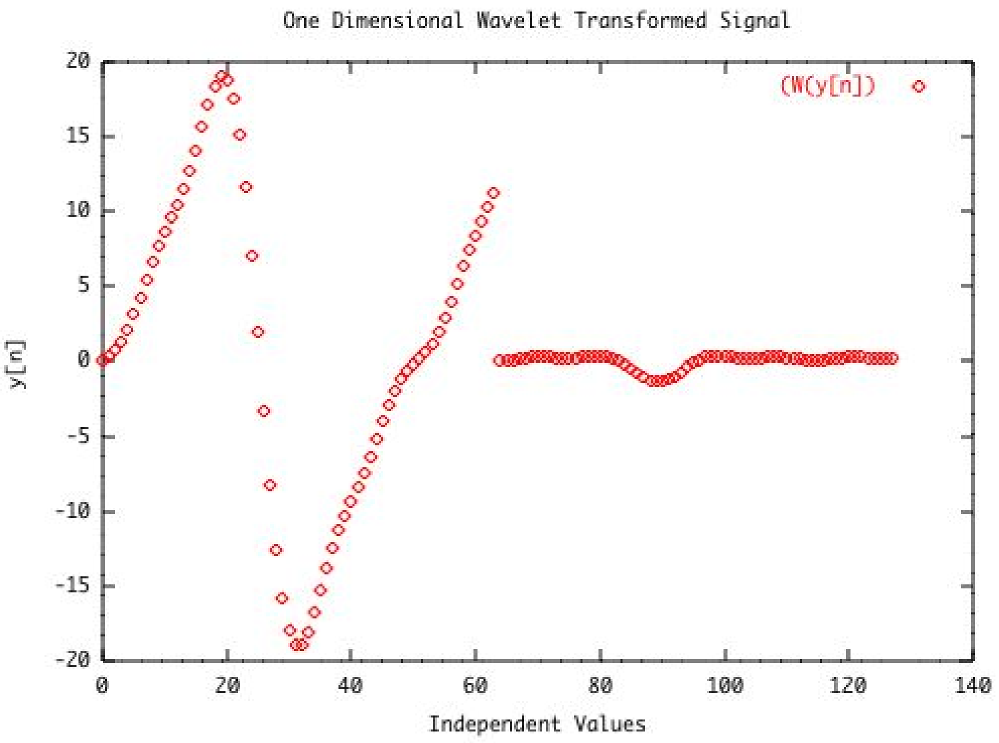

Figure 3.2: Signal after the wavelet transformation.

The convolution means is at the heart of the issue. The convolution operator in this case starts with the first element of the filter against the first element of the signal. In the simple Haar Wavelet case, there is a transformation pairing

$$(S_i, S_{i-1}) \rightarrow A_i, \text{ and } (S_i, S_{i-1}) \rightarrow D_i.$$

In this pairing with zero indexed signals, the odd indexed elements from the over-complete must be used to have all elements of the original accounted for.

Also this produces a functional difference between wavelet inverse transform for odd and even versions. The difference is slight; however, the last element is lost in the even indexed form.

$$\begin{array}{lll} \text{Odd:} & R_{2i} = (A_i - D_i)\sqrt{1/2}, & R_{2i+1} = (A_i + D_i)\sqrt{1/2} \\ \text{Even:} & R_{2i} = (A_i + D_i)\sqrt{1/2}, & R_{2i-1} = (A_i - D_i)\sqrt{1/2} \end{array}$$

An odd indexed wavelet transform yields the same energy. However, all of the values are accounted for. Refer to Figure 3.4.

3.2 Results: 2D Wavelet Transform

A simple room picture shows the difference that correct indexing produces in the wavelet transform and its inverse. The 1D to 2D method shows the incorrectly indexed case. A correctly indexed version is shown in the vector-matrix method.

The 1D to 2D implementation has a serious issue with memory leak errors. Memory is allocated and deallocated quickly, and on some platforms shows up as an error. Some platforms are not forgiving of this error and will force the program to terminate (Macintosh OSX 10.2, using gcc 3.1). On other platforms, the error is tolerated and performance is degraded (IRIX, SGI Octane2 using gcc 2.9). An example image

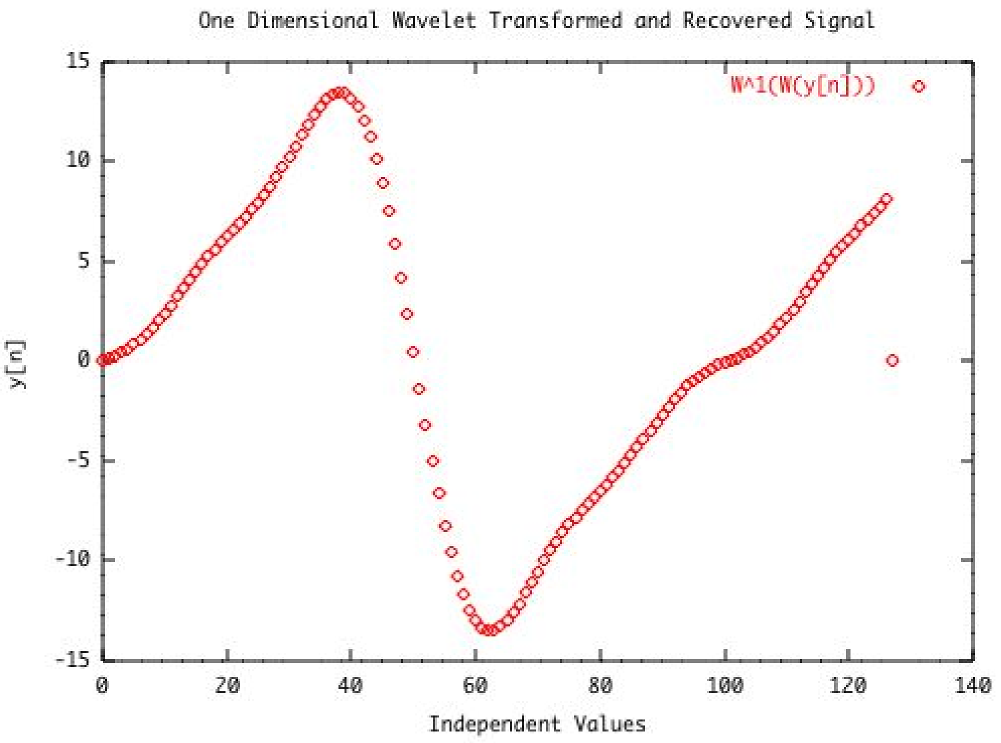

Figure 3.3: Recovered function. The x-axis is the array index (index n). The y value is simply the value $y(n)$. The function was recovered from an even indexed wavelet transform.

of 720 x 486 required nearly 10 minutes to compute the wavelet transform by this method on an SGI Octane2. However, it does eventually return a correct result.

The matrix-vector method also yields the correct result. However, there is less memory overhead in this method as compared to the 1D to 2D method. As a result, both the row wavelet transform and column wavelet transforms performed more quickly, with fewer memory transfers and allocations in the 256×256, 512×512 and

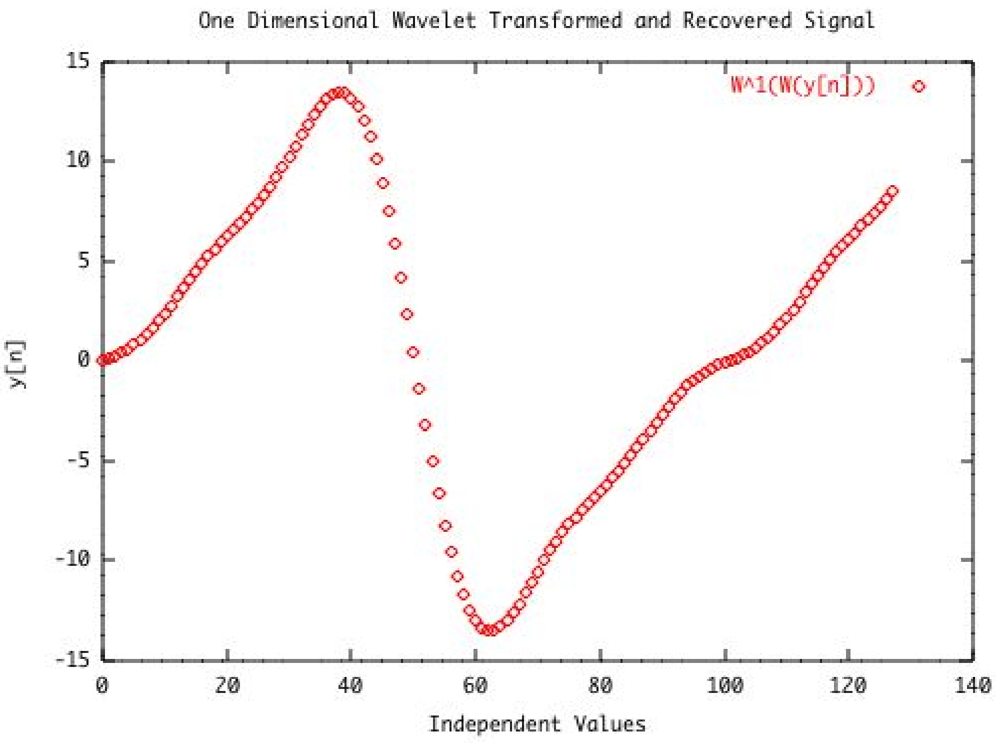

Figure 3.4: Recovered function. The x-axis is the array index (index n). The y value is simply the value y[n]. The function was recovered from an odd indexed wavelet transform.

768×768 cases. Obviously, this also allows for the operation to be conducted almost entirely in cache memory on both the SGI Octane2 and Macintosh G4 and G5 based machines. A Macintosh G3 based machine still requires main memory at a minimum to execute the same operation.

A correct result must also be matched to a correct inverse method. The indexing order matters. The inverse transform method is a forward inverse transform method.

In the case of 1D to 2D transform, the ordering was reverse indexed (Figure 3.7). As a result of an error in indexing, ringing is seen on edges in this method(Figure 3.6) for a case in point. Caution is incredibly important when matching both forward and reverse indexing, since matching the mathematics to the actual ordering can be obscure and tricky.

A correct result is shown in Figure 3.9. In this case, the indexing was matched up and ringing is not present. It is clear that the recovered image and the original (Figure 3.5) are nearly indistinguishable.

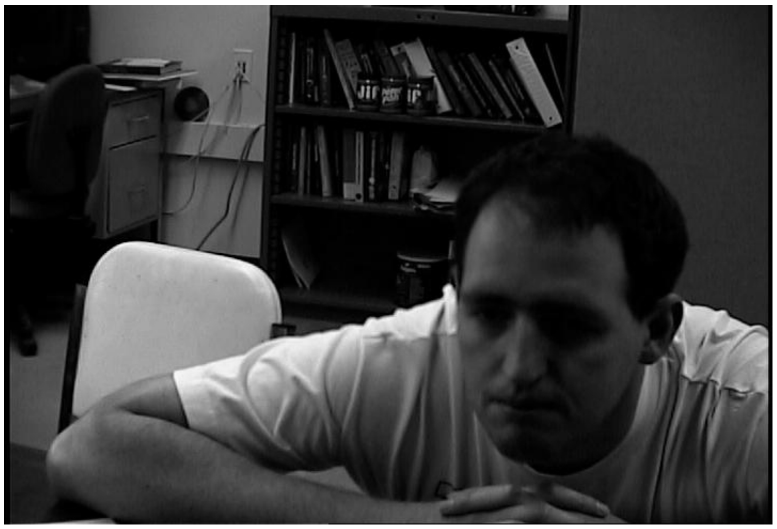

Figure 3.5: Original Image. This image is the original image.

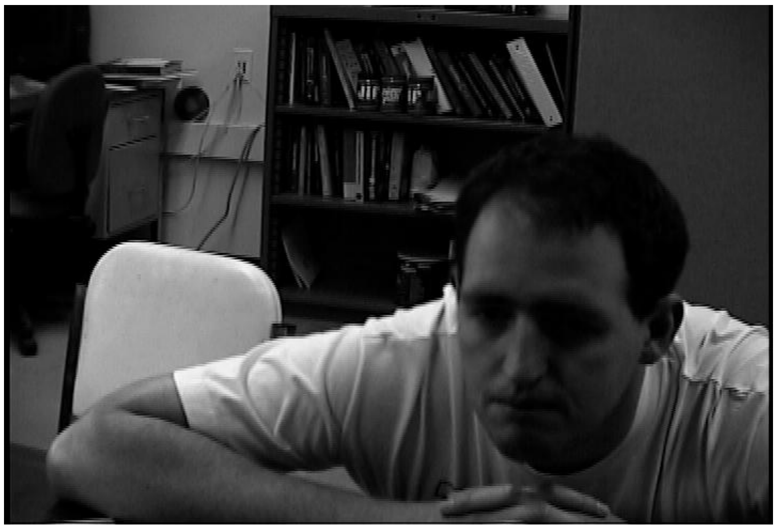

Figure 3.6: Recovered Image. This image is the recovered image. Depending on whether the image was saved as a picture first can affect the white spots in the picture. Ringing is also an issue.

3.2.1 Multiresolution Results

The expected result is a picture within a picture. Each average component has a further transform on it. The three resolution transform has the form:

$$\begin{pmatrix} A_3 & V_3 & V_2 & V_1 \\ H_3 & D_3 & & \\ H_2 & & D_2 & \\ & & & \\ H_1 & & & D_1 \\ & & & \end{pmatrix}$$

Refer to Figure 3.10 for the image transform results.

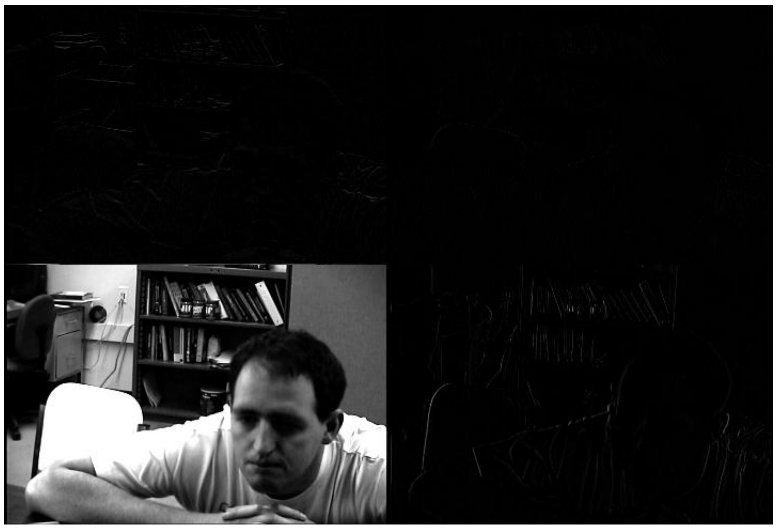
Figure 3.7: Wavelet Transform Image. This image is divided in to average, horizontal, vertical and diagonal components.

To obtain the inverse, an exact reverse procedure is necessary, otherwise the distortion is hideous. The first attempt of the wavelet inverse transform was out of order, refer to Figure 3.11. A correct picture was obtained during the second attempt. Correct order yielded correct results, refer to Figure 3.6.

3.2.2 Threshold Filtering

After a triple resolution, a 0.02 threshold will eliminate 81.1706 percent of elements in the original sample picture. Also at this point, the effects of removing these elements becomes visually evident (Figure 3.12). At a 0.01 threshold, 66.0205 percent of the elements are removed. Visually, the recovered sample and the original appear to be the same (Figure 3.15). At a threshold of 0.1, 92.9987 percent of the elements are reduced to zero. However, the distortions are clearly visible at this level

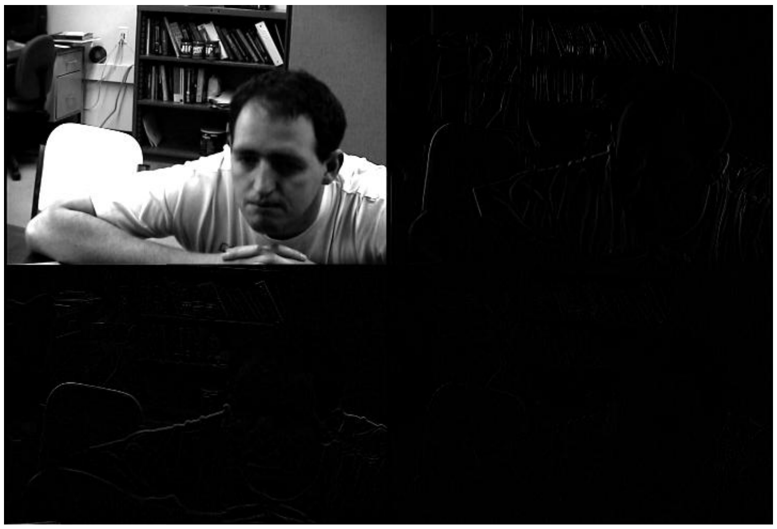
Figure 3.8: Wavelet Transform Image. This image is divided in to average, horizontal, vertical and diagonal components, using the vector-matrix version. The difference between this transform and that in Figure 3.7 is positional. Both have the same components, and strictly a matter of orientation. In this form the indexing for each element is simpler, and that is the preferable quality.

of thresholding (Figure 3.13). Even at a threshold of 0.001 which is below the numerical precision of the original, 16.0814 elements are reduced to zero. At a threshold of 0.002, 28.9683 percent is removed.

Consequently after a triple resolution, nearly 29% of the data was irrelevant for the image's brightness resolution (which also applies to color). Subjective examination reveals that removing 60% to 85% of the data was not noticeable to human perception. Which leaves only 15% to 40% of the data actually contributing or being necessary to reconstruct the image.

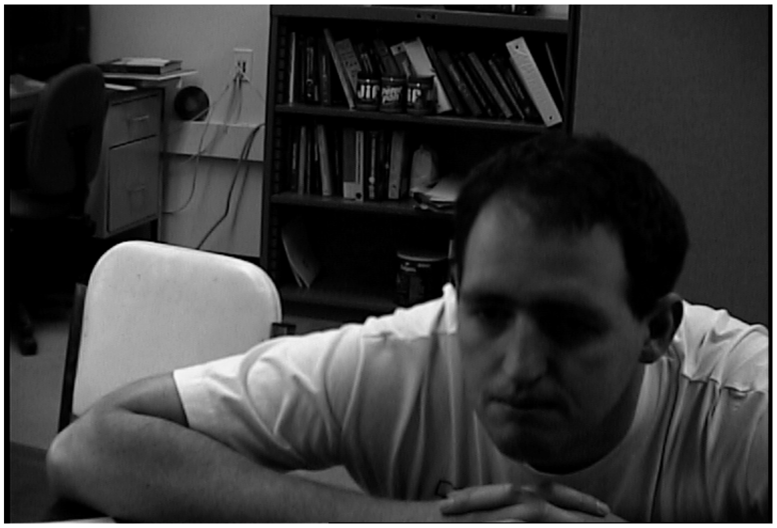

Figure 3.9: Recovered Image (Vector-Matrix Method). This image is the recovered image. This version avoids the ringing by using the vector-matrix version which is more aligned for the inverse wavelet transform.

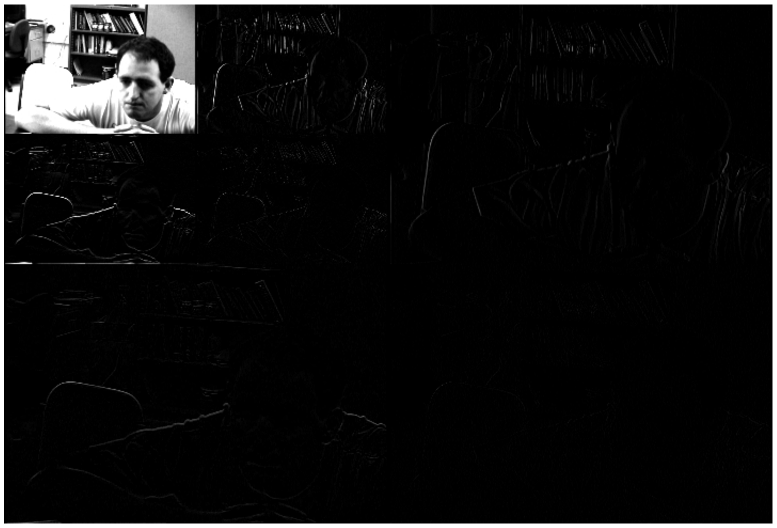

Figure 3.10: Wavelet Transform Image. This image is divided in to average, horizontal, vertical and diagonal components using multiresolution wavelet transform. Note the the average component was transformed one step further.

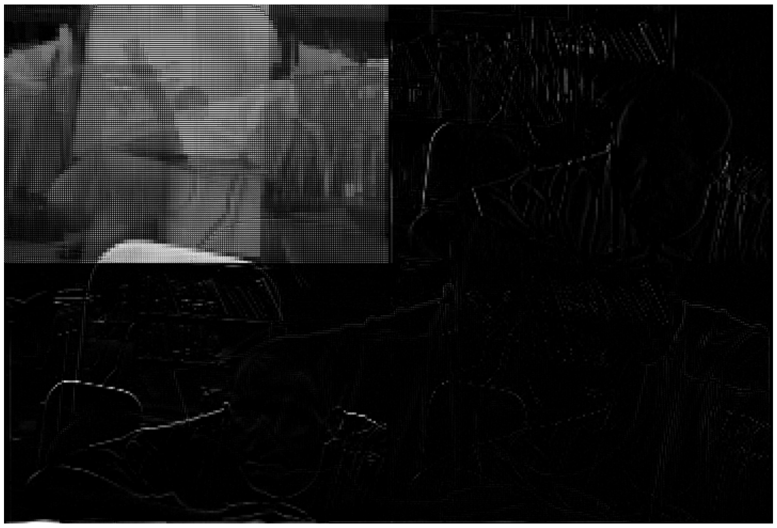

Figure 3.11: Recovered Image - Wrong Order (Multiresolution). This image shows a 2D wavelet transform after it was recovered out of order. Obviously, the distortion is hideous.

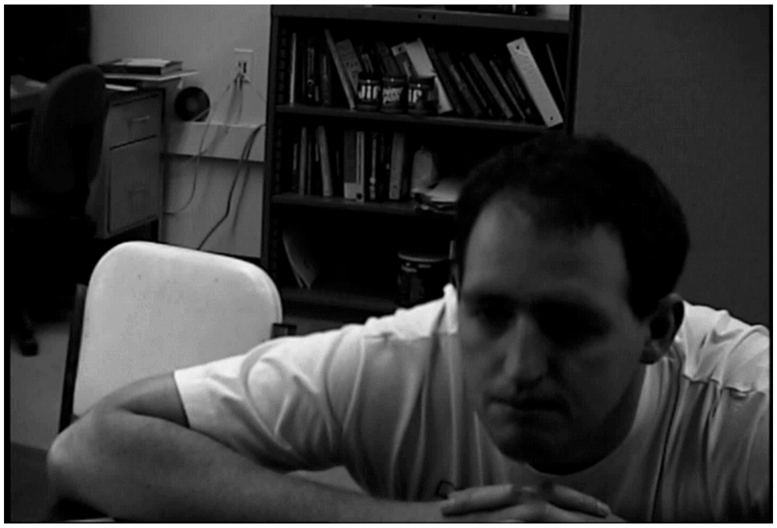

Figure 3.12: Recovered Image - 2% threshold (Multiresolution). This image had nearly 83% of its elements removed in the triple resolution wavelet transform.

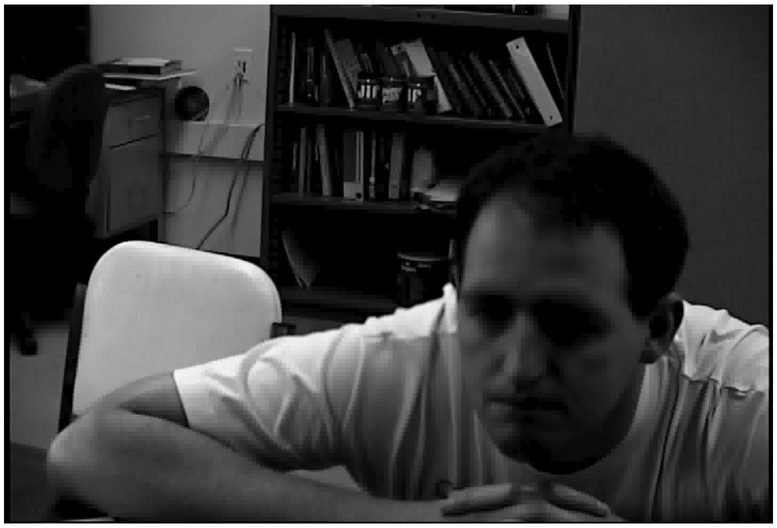

Figure 3.13: Recovered Image - 10% threshold (Multiresolution). This image had nearly 93% of its elements removed in the triple resolution wavelet transform.

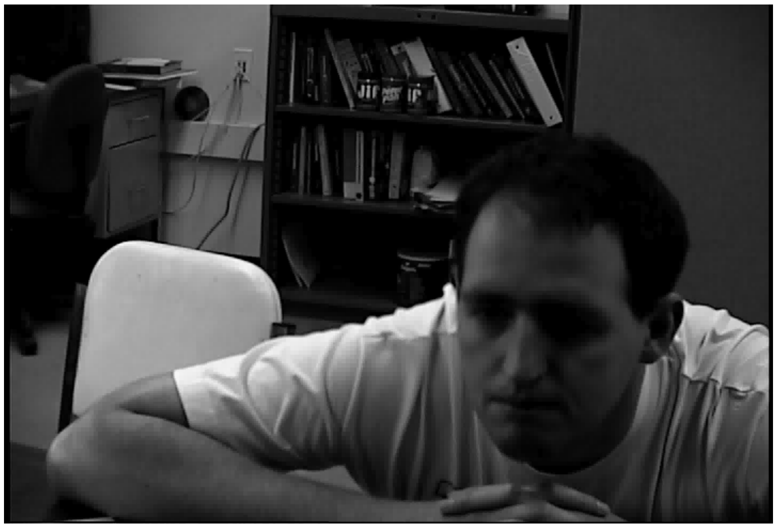

Figure 3.14: Recovered Image - 5% threshold (Multiresolution). This image had nearly 85% of its elements removed in the triple resolution wavelet transform.

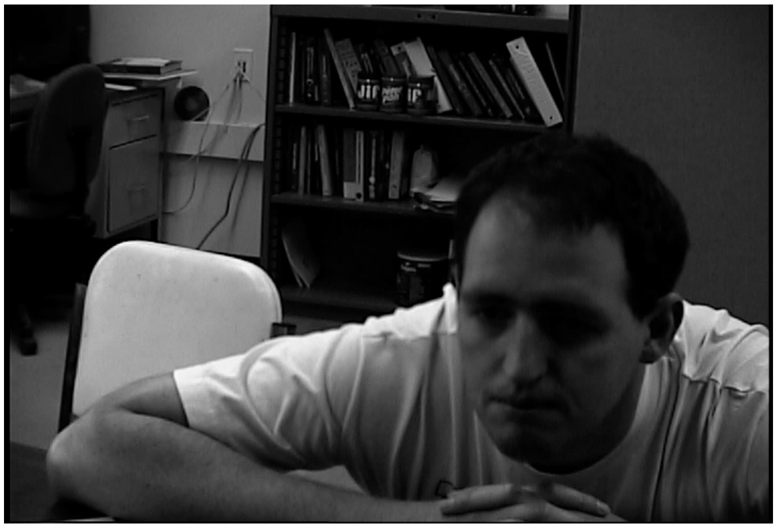

Figure 3.15: Recovered Image - 1% threshold (Multiresolution). This image had nearly 60% of its elements removed in the triple resolution wavelet transform.

CHAPTER 4

MATRIX MULTIPLICATION VIA WAVELETS

In this chapter, the general concept of matrix multiplication via wavelets is introduced. First, matrix multiplication and sparse matrix multiplication itself is reviewed for completeness. Then the ψ^n transform with the Haar basis is shown to provide a precodnitioned matrix which can be used to compute the matrix product. The correctness of this method is demonstrated through a formal proof.

4.1 Matrix Multiplication

Matrix multiplication is one of the fundamental operations in linear algebra. It is defined for two matrices A and B, denoted $C = A \cdot B$. Matrix multiply requires that the row length of A to be the same as the column length of B. The value of element, $c_{i,j}$ is defined by

$$c_{i,j} = \sum_k a_{i,k} b_{k,j}.$$

Matrix multiplication also obeys the following properties [14].

- Associative law: $(AB)C = A(BC)$
- Left Distributive Law $A(B + C) = AB + AC$
- Right Distributive Law $(B + C)A = BA + CA$
- Distribution of a constant

For two generic 2×2 matrices, A and B, the resulting products is

$$A \cdot B = \begin{pmatrix} a_1^1 & a_1^2 \\ a_2^1 & a_2^2 \end{pmatrix} \begin{pmatrix} b_1^1 & b_1^2 \\ b_2^1 & b_2^2 \end{pmatrix} = \begin{pmatrix} a_1^1 b_1^1 + a_1^2 b_2^1 & a_1^1 b_1^2 + a_1^2 b_2^2 \\ a_2^1 b_1^1 + a_2^2 b_2^1 & a_2^1 b_1^2 + a_2^2 b_2^2 \end{pmatrix}. \tag{4.1}$$

There is a fast matrix multiplication which was devised in 1969 by Strassen. It achieves a computational complexity of $O(N^{2.807})$ for large N representing one dimension of the matrix. This works for square matrices. There is a variation of the Strassen called the Winograd which obtains slightly better performance [11].

Another view on matrix multiplication is sparse multiplication. The whole point of sparse matrix multiplication is to take a matrix such that the matrix composition is mostly zero, and exploit that composition to reduce the number of multiplications required. Such multiplication schemes optimal limit for matrix multiplication is $O(N^2)$. One method is included in the Sparse Matrix Multiplication Package, called SYMBMM [3]. Another is contained in the Basic Linear Algebra Subprogram for sparse matrix to dense matrix multiply.

4.2 Wavelet Matrix Multiplication

The point of this thesis is not show the efficiencies of these above algorithms. Rather, it is to show a pre-conditioner using the wavelet transform that can be used in each of them. Wavelet based matrix multiply is sound for the Haar Wavelet Transform and the ψ^n expansion. If the Haar Wavelet Transform produces a sparse and well-conditioned matrix, then the Haar Wavelet Transform proves itself as a useful preconditioner. Here, the general concept of matrix multiplication via wavelets is introduced, and the linearity principle is shown. The key point for wavelet matrix multiplication is the proof that $W(\alpha) \times W(\beta) = W(\alpha \times \beta)$ If this is the case, then it is obvious that $W(\alpha) \times W(\beta) = W(\Gamma) = W(\alpha \times \beta = \Gamma)$.

4.2.1 A 2×2 example

The feasibility of multiplication in the wavelet domain is demonstrated directly using a 2×2 matrix. The coefficients of the matrices are multiplied both according to normal matrix multiplication and the modified wavelet multiplication operator. In the end the resulting coefficients are seen to be the same.

4.2.1.1 Wavelet Transforms of the Matrices

For a wavelet transform, the result on matrix α is

$$W(\alpha) = \frac{1}{2}\begin{pmatrix} (\alpha_1^1 + \alpha_2^1 + \alpha_1^2 + \alpha_2^2) & (\alpha_1^1 + \alpha_2^1 - \alpha_1^2 - \alpha_2^2) \\ (\alpha_1^1 - \alpha_2^1 + \alpha_1^2 - \alpha_2^2) & (\alpha_1^1 - \alpha_2^1 - \alpha_1^2 + \alpha_2^2) \end{pmatrix}, \tag{4.2}$$

and for matrix β it is

$$W(\beta) = \frac{1}{2}\begin{pmatrix} (\beta_1^1 + \beta_2^1 + \beta_1^2 + \beta_2^2) & (\beta_1^1 + \beta_2^1 - \beta_1^2 - \beta_2^2) \\ (\beta_1^1 - \beta_2^1 + \beta_1^2 - \beta_2^2) & (\beta_1^1 - \beta_2^1 - \beta_1^2 + \beta_2^2) \end{pmatrix}. \tag{4.3}$$

4.2.1.2 Product of A and B in wavelet space

The conventional product of A and B (equation 4.1) can be transformed into wavelet space. The wavelet transform of this matrix is represented by

$$W(\alpha \cdot \beta) = \frac{1}{2} \begin{pmatrix} \psi(A) & \psi(V) \\ \psi(H) & \psi(D) \end{pmatrix}$$

where

$$\psi(A) = (\alpha_1^1\beta_1^1 + \alpha_1^2\beta_2^1 + \alpha_1^1\beta_1^2 + \alpha_1^2\beta_2^2) + (\alpha_2^1\beta_1^1 + \alpha_2^2\beta_2^1 + \alpha_2^1\beta_1^2 + \alpha_2^2\beta_2^2) \quad (4.4)$$

$$\psi(V) = (\alpha_1^1\beta_1^1 + \alpha_1^2\beta_2^1 - \alpha_1^1\beta_1^2 - \alpha_1^2\beta_2^2) + (\alpha_2^1\beta_1^1 + \alpha_2^2\beta_2^1 - \alpha_2^1\beta_1^2 - \alpha_2^2\beta_2^2) \quad (4.5)$$

$$\psi(H) = (\alpha_1^1\beta_1^1 + \alpha_1^2\beta_2^1 + \alpha_1^1\beta_1^2 + \alpha_1^2\beta_2^2) - (\alpha_2^1\beta_1^1 + \alpha_2^2\beta_2^1 + \alpha_2^1\beta_1^2 + \alpha_2^2\beta_2^2) \quad (4.6)$$

$$\psi(D) = (\alpha_1^1\beta_1^1 + \alpha_1^2\beta_2^1 - \alpha_1^1\beta_1^2 - \alpha_1^2\beta_2^2) - (\alpha_2^1\beta_1^1 + \alpha_2^2\beta_2^1 - \alpha_2^1\beta_1^2 - \alpha_2^2\beta_2^2). \quad (4.7)$$

4.2.1.3 The product of the waveletized matrices

Straightforward multiplication of $W(A) \cdot W(B)$ represented by equations 4.2 and 4.3 works out as follows:

$$W(\alpha) \cdot W(\beta) = \frac{1}{4} \begin{pmatrix} W_A & W_V \\ W_H & W_D \end{pmatrix}$$

where

$$\begin{aligned}
W_A &= (\alpha_1^1 + \alpha_2^1 + \alpha_1^2 + \alpha_2^2)(\beta_1^1 + \beta_2^1 + \beta_1^2 + \beta_2^2) + (\alpha_1^1 + \alpha_2^1 - \alpha_1^2 - \alpha_2^2)(\beta_1^1 - \beta_2^1 + \beta_1^2 - \beta_2^2) \\
W_V &= (\alpha_1^1 + \alpha_2^1 + \alpha_1^2 + \alpha_2^2)(\beta_1^1 + \beta_2^1 - \beta_1^2 - \beta_2^2) + (\alpha_1^1 + \alpha_2^1 - \alpha_1^2 - \alpha_2^2)(\beta_1^1 - \beta_2^1 - \beta_1^2 + \beta_2^2) \\
W_H &= (\alpha_1^1 - \alpha_2^1 + \alpha_1^2 - \alpha_2^2)(\beta_1^1 + \beta_2^1 + \beta_1^2 + \beta_2^2) + (\alpha_1^1 - \alpha_2^1 - \alpha_1^2 + \alpha_2^2)(\beta_1^1 - \beta_2^1 + \beta_1^2 - \beta_2^2) \\
W_D &= (\alpha_1^1 - \alpha_2^1 + \alpha_1^2 - \alpha_2^2)(\beta_1^1 + \beta_2^1 - \beta_1^2 - \beta_2^2) + (\alpha_1^1 - \alpha_2^1 - \alpha_1^2 + \alpha_2^2)(\beta_1^1 - \beta_2^1 - \beta_1^2 + \beta_2^2)
\end{aligned}$$

which simplifies to

$$\begin{aligned}
W_A &= \alpha_1^1\beta_1^1 + \alpha_2^1\beta_1^1 + \alpha_1^2\beta_2^1 + \alpha_2^2\beta_2^1 + \alpha_1^1\beta_1^2 + \alpha_2^1\beta_1^2 + \alpha_1^2\beta_2^2 + \alpha_2^2\beta_2^2 \\
W_V &= \alpha_1^1\beta_1^1 + \alpha_2^1\beta_1^1 + \alpha_1^2\beta_2^1 + \alpha_2^2\beta_2^1 - \alpha_1^1\beta_1^2 - \alpha_2^1\beta_1^2 - \alpha_1^2\beta_2^2 - \alpha_2^2\beta_2^2 \\
W_H &= \alpha_1^1\beta_1^1 - \alpha_2^1\beta_1^1 + \alpha_1^2\beta_2^1 - \alpha_2^2\beta_2^1 + \alpha_1^1\beta_1^2 - \alpha_2^1\beta_1^2 + \alpha_1^2\beta_2^2 - \alpha_2^2\beta_2^2 \\
W_D &= \alpha_1^1\beta_1^1 - \alpha_2^1\beta_1^1 + \alpha_1^2\beta_2^1 - \alpha_2^2\beta_2^1 - \alpha_1^1\beta_1^2 + \alpha_2^1\beta_1^2 - \alpha_1^2\beta_2^2 + \alpha_2^2\beta_2^2
\end{aligned}$$

These can then be compared to the coefficients of $W(A \cdot B)$ in equations 4.4-4.5 and seen to be identical. This asserts that $W(A) \cdot W(B) = W(A \cdot B)$ in the case of 2×2 matrices.

4.2.2 Haar Wavelet Multiplication

Given that the multiplication of the transformed coefficients is the same as the transformation of the multiplied coefficients, an algorithm can be developed to take advantage of this. The Haar Wavelet Multiplication Algorithm is presented in Algorithm 11. This produces a correct solution.

Algorithm 11 Haar Wavelet Multiplication

```
Require: Matrices $A$ and $B$
  $\hat{A} \leftarrow \psi^n(A)$
  $\hat{B} \leftarrow \psi^n(B)$
  $\hat{C} \leftarrow \hat{A} \cdot \hat{B}$
  $C \leftarrow \psi^{-n}\left(\hat{C}\right)$
```

However, the preconditioned matrix has not been sparsified. In order to sparsify the matrix, it is thresholded. Then, only nonzero elements are retained and sparse matrix matrix multiplication can be employed on the previously dense matrix. This is presented in Algorithm 12.

Algorithm 12 Sparse Haar Wavelet Multiplication

```
Require: Matrices $A$ and $B$
  $\hat{A} \leftarrow \psi^n(A)$
  $\hat{B} \leftarrow \psi^n(B)$
  $l_A \leftarrow \text{sparsify}\left(\hat{A}\right)$
  $l_B \leftarrow \text{sparsify}\left(\hat{B}\right)$
  $l_C \leftarrow \text{sparse multiply}\,(l_A, l_B)$
  $\hat{C} \leftarrow \text{densify}\left(\hat{C}\right)$
  $C \leftarrow \psi^{-n}\left(\hat{C}\right)$
```

4.3 Formal Proof of the Haar Wavelet Multiplication

The formal proof of the correctness of the haar wavelet multiplication is presented below.

4.3.1 Haar Wavelets and Vector Inner Products

One of the other crucial keys for the Haar Wavelet Transform Matrix Multiply to work is vector inner product. The issue is whether or not

$$\langle f', g' \rangle = \langle f, g \rangle \tag{4.8}$$

is true for vectors f and g which both of length p, and the wavelet transformed versions of f and g denoted f' and g' respectively. To show this, $\langle f', g' \rangle$ is expanded algebraically to establish its identity.

$$\langle f', g' \rangle = \sum_{k=0}^{p/2-1} (\frac{f_{2k} + f_{2k+1}}{\sqrt{2}} \cdot \frac{g_{2k} + g_{2k+1}}{\sqrt{2}}) + \sum_{k=0}^{p/2-1} (\frac{f_{2k} - f_{2k+1}}{\sqrt{2}} \cdot \frac{g_{2k} - g_{2k+1}}{\sqrt{2}}) \tag{4.9}$$

When equation 4.9 is expanded further, the terms that emerge expose the identity of the inner product.

$$\begin{aligned} \langle f', g' \rangle = \frac{1}{2} \sum_{k=0}^{p/2-1} (& f_{2k}g_{2k} + f_{2k+1}g_{2k} + f_{2k}g_{2k+1} + f_{2k+1}g_{2k+1} \\ & + f_{2k}g_{2k} - f_{2k+1}g_{2k} - f_{2k}g_{2k+1} + f_{2k+1}g_{2k+1} \end{aligned} \tag{4.10}$$

When simplified the $f_{2k}g_{2k+1}$ and $f_{2k+1}g_{2k}$ terms cancel. The $f_{2k}g_{2k}$ and $f_{2k+1}g_{2k+1}$ terms combine to yield equation 4.11 and further simplifies to equation vectinwtsimple2.

$$\langle f', g' \rangle = \frac{1}{2} \sum_{k=0}^{p/2-1} (f_{2k}g_{2k} + f_{2k+1}g_{2k+1}) \tag{4.11}$$

$$\langle f', g' \rangle = \frac{1}{2}\sum_{k=0}^{p-1}(f_k g_k) = \langle f, g \rangle \tag{4.12}$$

4.3.2 Matrix Multiply with the Haar Transform: Formal Proof

Given two arbitary matrices A of size $m \times p$, B of size $p \times n$, the Haar Wavelet Pair, and the Haar Wavelet Transform. The wavelet pair to be used here is

$$\psi_H(x) = \sqrt{\frac{1}{2}} \begin{cases} 1 & x = 0 \\ -1 & x = 1 \\ 0 & otherwise \end{cases}$$

$$\phi_H(x) = \sqrt{\frac{1}{2}} \begin{cases} 1 & x = 0 \; and \; x = 1 \\ 0 & otherwise \end{cases} .$$

Also, this notation is used for this proof.

- $\psi_{1R}(S)$ is the row transform of matrix S.
- $\psi_{1C}(S)$ is the column transform of matrix S.
- ψS is the 2-D wavelet transform of matrix S.
- $\langle f, g \rangle = \langle \psi_1(f), \psi_1(g) \rangle$
- $A' = \psi(A)$
- $B' = \psi(B)$
- $A^R = \psi_{1R}(A)$

- $B^C = \psi_{1C}(B)$
- A^R_{ri} is the ith row vector of the row transform of A. This is the same as saying $A_{ri} = A(i, :)$
- A^R_{ri+1} is the $i + 1$th row vector of the row transform of A. This is the same as saying $A_{ri+1} = A(i + 1, :)$
- B^C_{cj} is the jth column vector of the column transform of B. This is the same as saying $B_{cj} = B(:, j)$
- B^C_{cj+1} is the $j + 1$th column vector of the column transform of B. This is the same as saying $B_{cj+1} = B(:, j + 1)$

Required show that

$$\psi(A)\psi(B) = \psi(AB)$$

is a true statement.

Proof

There are two formulae required for transforming either A or B into the wavelet domain.

$$\psi(A) = \psi_{1R}(\psi_{1C}(A)) = \psi_{1C}(\psi_{1R}(A)) \tag{4.13}$$

$$\psi(B) = \psi_{1R}(\psi_{1C}(B)) = \psi_{1C}(\psi_{1R}(B)) \tag{4.14}$$

Thus, this is simply a re-write of the definition of the wavelet transform by the CWT. Next, define a matrix Γ' so that

$$\Gamma' = \psi(A)\psi(B).$$

From, the definition Γ', a series of re-writes and the properties of the Haar Wavelet Transform can expose the solution for every element of Γ' and Γ'. In general, the elements of Γ' defined as follows:

$$\Gamma'_{i,j} = \langle \psi_{1C}(\psi_{1R}(A))_{ri}, \psi_{1R}(\psi_{1C}(B))_{ci} \rangle \tag{4.15}$$

$$\Gamma'_{i,j} = \langle \psi_{1R}(\psi_{1C}(A))_{ri}, \psi_{1C}(\psi_{1R}(B))_{ci} \rangle \tag{4.16}$$

In this case, each element of Γ' is defined as nothing more than the inner product of row i of $\psi(A)$ and column j of $\psi(B)$. This is in agreement with standard matrix multiplication.

Another identity that is important what the wavelet pair for transform will do in either a row or column transform.

$$\psi_{1C}(A) = \begin{cases} \frac{A_{i,j}+A_{i,j+1}}{\sqrt{2}} & j < \frac{n}{2} \\ \frac{A_{i,j}-A_{i,j+1}}{\sqrt{2}} & j \geq \frac{n}{2} \end{cases}$$

$$\psi_{1R}(A) = \begin{cases} \frac{A_{i,j}+A_{i+1,j}}{\sqrt{2}} & i < \frac{m}{2} \\ \frac{A_{i,j}-A_{i+1,j}}{\sqrt{2}} & i \geq \frac{m}{2} \end{cases}$$

In this case, there are four cases to show for:

1. $i < \frac{m}{2}$ and $j < \frac{n}{2}$
2. $i \geq \frac{m}{2}$ and $j < \frac{n}{2}$
3. $i < \frac{m}{2}$ and $j \geq \frac{n}{2}$
4. $i \geq \frac{m}{2}$ and $j \geq \frac{n}{2}$

Case 1 $i < \frac{m}{2}$ and $j < \frac{n}{2}$. The base equation $\Gamma'_{i,j}$ in this cases is as follows:

$$\Gamma'_{ij} = \left\langle \frac{A^R_{ri} + A^R_{ri+1}}{\sqrt{2}}, \frac{B^C_{cj} + B^C_{cj+1}}{\sqrt{2}} \right\rangle \tag{4.17}$$

This re-write must be expanded to expose an equality. This expansion is valid for vectors sums within inner products. Thus $\Gamma'_{i,j}$ in this case is expanded to:

$$\Gamma'_{i,j} = \frac{1}{2}(\langle A^R_{ri}, B^C_{cj}\rangle + \langle A^R_{ri+1}, B^C_{cj}\rangle + \langle A^R_{ri}, B^C_{cj+1}\rangle + \langle A^R_{ri+1}, B^C_{cj+1}\rangle).$$

Next, $\Gamma'_{i,j}$ must be compared to $\psi(\Gamma)_{i,j}$ where

$$\psi(AB) = \psi(\Gamma)$$

By definition, every element in C is defined:

$$\Gamma_{i,j} = \langle A_{ri}, B_{cj} \rangle$$

The next step is to show what every element within this case is for $\psi(\Gamma)$. This is done by analyzing what the column transform will do at $C_{i,j}$. To do this, the column transform is applied to both column j and $j+1$ for columns between $[0, \frac{col}{2} - 1]$. The effects of these column transforms are defined in equations 4.18 and 4.19.

$$\psi_{1C}(\Gamma)_{i,j} = \frac{1}{\sqrt{2}} \langle A_{ri}, B_{cj} \rangle + \langle A_{ri+1}, B_{cj} \rangle \tag{4.18}$$

$$\psi_{1C}(\Gamma)_{i,j+1} = \frac{1}{\sqrt{2}} \langle A_{ri}, B_{cj+1} \rangle + \langle A_{ri+1}, B_{cj+1} \rangle \tag{4.19}$$

The result of the row transform on $\psi_{1C}(\Gamma)$ in this case is a sum of $\psi_{1C}(\Gamma)_{i,j}$ and $\psi_{1C}(\Gamma)_{i,j+1}$, and specified in equation 4.20.

$$\psi(\Gamma) = \frac{1}{\sqrt{2}}(\psi_{1C}(\Gamma)_{i,j} + \psi_{1C}(\Gamma)_{i,j+1}) \tag{4.20}$$

Expanded this equation is equation 4.21.

$$\psi(\Gamma) = \frac{1}{2}(\langle A_{ri}, B_{cj} \rangle + \langle A_{ri+1}, B_{cj} \rangle + \langle A_{ri}, B_{cj+1} \rangle + \langle A_{ri+1}, B_{cj+1} \rangle) \tag{4.21}$$

Case 2 $i \geq \frac{m}{2}$ and $j < \frac{n}{2}$. The base equation $\Gamma'_{i,j}$ in this cases is as follows:

$$\Gamma'_{ij} = \left\langle \frac{A^R_{ri,j} - A^R_{ri+1}}{\sqrt{2}}, \frac{B^C_{cj} + B^C_{cj+1}}{\sqrt{2}} \right\rangle$$

The next step is to show what every element within this case is for $\psi(\Gamma)$.

$$\Gamma'_{i,j} = \frac{1}{2}(\left\langle A^R_{ri}, B^C_{cj} \right\rangle - \left\langle A^R_{ri+1}, B^C_{cj} \right\rangle + \left\langle A^R_{ri}, B^C_{cj+1} \right\rangle - \left\langle A^R_{ri+1}, B^C_{cj+1} \right\rangle)$$

How does this compare with $\psi(\Gamma)$?

$$\psi(AB) = \psi(\Gamma)$$

$$(\Gamma) = \psi \left\langle A_{ri}, B_{cj} \right\rangle$$

This equation expands with wavelet operator to:

$$\psi_{1C}(\Gamma)_{i,j} = \frac{1}{\sqrt{2}} \left\langle A_{ri}, B_{cj} \right\rangle - \left\langle A_{ri+1}, B_{cj} \right\rangle$$

$$\psi_{1C}(\Gamma)_{i,j+1} = \frac{1}{\sqrt{2}} \left\langle A_{ri}, B_{cj+1} \right\rangle - \left\langle A_{ri+1}, B_{cj+1} \right\rangle$$

The result of the row transform on $\psi_{1C}(\Gamma)$ in this case is a sum of $\psi_{1C}(\Gamma)_{i,j}$ and

$$\psi(\Gamma) = \frac{1}{\sqrt{2}}(\psi_{1C}(\Gamma)_{i,j} + \psi_{1C}(\Gamma)_{i,j+1})$$

Expanded this equation is equation 4.21.

$$\psi(\Gamma) = \frac{1}{2}(\langle A_{ri}, B_{cj}\rangle - \langle A_{ri+1}, B_{cj}\rangle + \langle A_{ri}, B_{cj+1}\rangle - \langle A_{ri+1}, B_{cj+1}\rangle)$$

Case 3 $i < \frac{m}{2}$ and $j \geq \frac{n}{2}$. Like, before the base case for $\Gamma'_{i,j}$

$$\Gamma'_{ij} = \left\langle \frac{A^R_{ri,j} + A^R_{ri+1}}{\sqrt{2}}, \frac{B^C_{cj} - B^C_{cj+1}}{\sqrt{2}} \right\rangle$$

Again, the vector sums are expanded within their inner products.

$$\Gamma'_{i,j} = \frac{1}{2}(\left\langle A^R_{ri}, B^C_{cj}\right\rangle + \left\langle A^R_{ri+1}, B^C_{cj}\right\rangle - \left\langle A^R_{ri}, B^C_{cj+1}\right\rangle - \left\langle A^R_{ri+1}, B^C_{cj+1}\right\rangle)$$

Next, $\Gamma'_{i,j}$ is compared with $\psi(\Gamma)$

$$\psi(AB) = \psi(\Gamma).$$

Next, the column transform expansion is conducted:

$$(\Gamma) = \psi \langle A_{ri}, B_{cj}\rangle$$

$$\psi_{1C}(\Gamma)_{i,j} = \frac{1}{\sqrt{2}} \langle A_{ri}, B_{cj}\rangle + \langle A_{ri+1}, B_{cj}\rangle$$

$$\psi_{1C}(\Gamma)_{i,j+1} = \frac{1}{\sqrt{2}} \langle A_{ri}, B_{cj+1}\rangle + \langle A_{ri+1}, B_{cj+1}\rangle$$

Now the column transform results are combined, by subtraction in this case:

$$\psi(\Gamma) = \frac{1}{\sqrt{2}}(\psi_{1C}(\Gamma)_{i,j} - \psi_{1C}(\Gamma)_{i,j+1})$$

Lastly, the expanded form is expanded.

$$\psi(\Gamma) = \frac{1}{2}(\langle A_{ri}, B_{cj}\rangle + \langle A_{ri+1}, B_{cj}\rangle - \langle A_{ri}, B_{cj+1}\rangle - \langle A_{ri+1}, B_{cj+1}\rangle)$$

Case 4 $i \geq \frac{m}{2}$ and $j \geq \frac{n}{2}$. Like, before the base case for $\Gamma'_{i,j}$

$$\Gamma'_{ij} = \left\langle \frac{A^R_{ri,j} - A^R_{ri+1}}{\sqrt{2}}, \frac{B^C_{cj} - B^C_{cj+1}}{\sqrt{2}} \right\rangle$$

Again, the vector sums are expanded within their inner products.

$$\Gamma'_{i,j} = \frac{1}{2}(\left\langle A^R_{ri}, B^C_{cj}\right\rangle - \left\langle A^R_{ri+1}, B^C_{cj}\right\rangle - \left\langle A^R_{ri}, B^C_{cj+1}\right\rangle + \left\langle A^R_{ri+1}, B^C_{cj+1}\right\rangle)$$

Next, $\Gamma'_{i,j}$ is compared with $\psi(\Gamma)$

$$\psi(AB) = \psi(\Gamma)$$

$$(\Gamma) = \psi \left\langle A_{ri}, B_{cj}\right\rangle.$$

Once the column wavelet transform expands Γ to

$$\psi_{1C}(\Gamma)_{i,j} = \frac{1}{\sqrt{2}} \langle A_{ri}, B_{cj} \rangle - \langle A_{ri+1}, B_{cj} \rangle$$

$$\psi_{1C}(\Gamma)_{i,j+1} = \frac{1}{\sqrt{2}} \langle A_{ri}, B_{cj+1} \rangle - \langle A_{ri+1}, B_{cj+1} \rangle .$$

Now the column transform results are combined, by subtraction in this case:

$$\psi(\Gamma) = \frac{1}{\sqrt{2}}(\psi_{1C}(\Gamma)_{i,j} - \psi_{1C}(\Gamma)_{i,j+1}).$$

Lastly, the expanded form is expanded.

$$\psi(\Gamma) = \frac{1}{2}(\langle A_{ri}, B_{cj} \rangle - \langle A_{ri+1}, B_{cj} \rangle - \langle A_{ri}, B_{cj+1} \rangle + \langle A_{ri+1}, B_{cj+1} \rangle)$$

Since the inner product of two vectors f, g is the same as their wavelet transformed vectors f', g', then all 4 cases produce a case where $\Gamma'_{i,j} = (\psi(\Gamma))_{i,j}\ \forall (i, j)$. Thus for the Haar Wavelet Transform (discrete mother basis),

$$\psi(A) \cdot \psi(B) = \psi(A \cdot B)$$

.

4.4 Multi Resolution Expansion Proof

In a previous section, wavelet based matrix multiplication was proven such that

$$\psi(A) \cdot \psi(B) = \psi(A \cdot B)$$

In this section the question of whether or not there is MRE form which is sound for matrix multiplication. There are few facts that are relevant and were made obvious in the empirical analysis in the results chapter.

- $\psi_{WPx}(A) \neq \psi^x(A)$ where $\psi_{WPx}(A)$ is the x resolution of wavelet transform packets (full decomposition) except for $x = 1$.
- $\psi_{Wx}(A) \neq \psi^x(A)$ where $\psi_{Wx}(A)$ is the x resolution of wavelet transform pyramids except for $x = 1$.

The next hypothesis to be answered is can the wavelet transform be applied more than once to condition a matrix for matrix multiplication? This question is answered by the following lemma.

$$\psi^2(A) \cdot \psi^2(B) = \psi^2(A \cdot B)$$

Proof:

The theorem

$$\psi(A) \cdot \psi(B) = \psi(A \cdot B)$$

is proven as fact.

$$\psi^2(A) = \psi(\psi(A))$$

$$\psi^2(A) = \cdot\psi^2(B) = \psi(\psi(A)) \cdot \psi(\psi(B))$$

$$\psi(\psi(A)) \cdot \psi(\psi(B)) = \psi(\psi(A) \cdot \psi(B))$$

$$\psi(\psi(A) \cdot \psi(B)) = \psi(\psi(A \cdot (B))$$

$$\psi(\psi(A \cdot (B)) = \psi^2(A \cdot B)$$

Therefore:

$$\psi^2(A) \cdot \psi^2(B) = \psi^2(A \cdot B)$$

CHAPTER 5

RESULTS

Each empirical result is from a series of matrix multiplications on 9 different images. These matrices were submitted to preconditioning with the wavelet transform in the 3 basic method of decomposition — MRA, MRE and the ψ^n decomposition.In the end, MRA performed poorly for any set of resolutions deeper than one in matrix multiplication. Thus MRA should not be used as a pre-conditioner to matrix multiplication. The MRE transform performs modestly in matrix multiplication tests, but to be effective must be reordered into ψ^n structure. Lastly, the ψ^n form performs well on matrix multiplication.

The matrix multiplication used in each of these cases is $A^2 = A \cdot A$, where A is a $M \times N$ matrix. The general equation for evaluating the wavelet transform pyramid method is the fidelity measurement:

$$\sqrt{E(\psi^{-1}((\psi(A))^2) - A^2)}.$$

where ψ represents the given multilevel wavelet transform and ψ^{-1} represents its inverse. Relative fidelity measurements provide an error that takes into account the scale. It is more useful for general observations. To obtain this measurement, the results are compared to standard. In this case, A^2 is the standard, and the energy difference is inversely proportional.

$$\sqrt{\frac{E(\psi^{-1}((\psi(A))^2) - A^2)}{E(A^2)}}$$

In both fidelity and relative fidelity, the energy of the function is defined.

$$E(A) = \frac{1}{m \cdot n} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} (a_{i,j})^2$$

5.1 8×8 Examples

This section demonstrates matrix multiplication on ψ^n expanded 8×8 matrix multiplication. One matrix in the set is an upper triangular matrix it is multiplied by itself. Another is a matrix of $\frac{1}{2}$s everywhere except the diagonal being $\frac{1}{4}$ and it is also multiplied by itself. In the third example, the upper triangular matrix and matrix of $\frac{1}{2}$ are multiplied together.

The values are inserted and retrieved from the program in PGM and PPM formats with their signs retained. Also, values above 256 and below zero are retained. Each value is a quantized by 256 during the input and output stages. While processing, the values are computed with double floating point precision. In these case, the only error can be accounted for by quantization and numerical round off at the output.

5.1.1 Matrix multiplication on 8×8 Upper Triangle

This first example of multiplication uses the matrix defined in equation 5.1 as the test matrix A. A^2 is provided in equation 5.2.

$$A = \frac{1}{256}\begin{pmatrix} 64 & 128 & 128 & 128 & 128 & 128 & 128 & 128 \\ 0 & 64 & 128 & 128 & 128 & 128 & 128 & 128 \\ 0 & 0 & 64 & 128 & 128 & 128 & 128 & 128 \\ 0 & 0 & 0 & 64 & 128 & 128 & 128 & 128 \\ 0 & 0 & 0 & 0 & 64 & 128 & 128 & 128 \\ 0 & 0 & 0 & 0 & 0 & 64 & 128 & 128 \\ 0 & 0 & 0 & 0 & 0 & 0 & 64 & 128 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 64 \end{pmatrix} \tag{5.1}$$

$$A^2 = \frac{1}{256}\begin{pmatrix} 17 & 65 & 129 & 193 & 258 & 322 & 386 & 450 \\ 0 & 17 & 65 & 129 & 193 & 258 & 322 & 386 \\ 0 & 0 & 17 & 65 & 129 & 193 & 258 & 322 \\ 0 & 0 & 0 & 17 & 65 & 129 & 193 & 258 \\ 0 & 0 & 0 & 0 & 17 & 65 & 129 & 193 \\ 0 & 0 & 0 & 0 & 0 & 17 & 65 & 129 \\ 0 & 0 & 0 & 0 & 0 & 0 & 17 & 65 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 17 \end{pmatrix} \tag{5.2}$$

The steps and notation for analyzing the ψ^3 expansion of A are listed here:

- $W^3(A)$ is the ψ^3 expansion of A and is shown in equation 5.3.
- $(W^3(A))^2$ is the square of $W^3(A)$ and is shown in equation 5.3.
- $W^{-3}((W^3(A))^2)$ is the ψ^-3 inverse of $(W^3(A))^2$. and is shown in equation 5.5.

The ψ^3 expansion of A has six elements of the 64 elements are below $\frac{1}{512}$. Another ten of the 64 elements are below $\frac{3}{512}$. Relative fidelity is on the order of 10^-14.

$$W^3(A) = \frac{1}{256}\begin{pmatrix} 512 & 64 & 128 & 0 & 256 & 0 & 0 & 0 \\ -64 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ -128 & 0 & 0 & 65 & 0 & 1 & 1 & 1 \\ 1 & 1 & -64 & 0 & 1 & 1 & 0 & 0 \\ -256 & 1 & 0 & 0 & 0 & 65 & 129 & 0 \\ 1 & 0 & 0 & 1 & -64 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & -128 & 0 & 0 & 65 \\ 0 & 1 & 0 & 0 & 1 & 1 & -64 & 0 \end{pmatrix} \tag{5.3}$$

$$(W^3(A))^2 = \frac{1}{256}\begin{pmatrix} 691 & 129 & 258 & 33 & 515 & 65 & 129 & 0 \\ -128 & -16 & -32 & 0 & -64 & 0 & 0 & 1 \\ -257 & -32 & -80 & 1 & -128 & 0 & 0 & 1 \\ 33 & 1 & 1 & -16 & 1 & 0 & 0 & 0 \\ -514 & -64 & -128 & 1 & -337 & 1 & 1 & 33 \\ 65 & 1 & 1 & 0 & 1 & -16 & -32 & 0 \\ 129 & 1 & 1 & 0 & 1 & -32 & -80 & 1 \\ 0 & 0 & 0 & 0 & 33 & 1 & 1 & -16 \end{pmatrix} \tag{5.4}$$

$$W^-3((W^3(A))^2) = \frac{1}{256}\begin{pmatrix} 17 & 65 & 129 & 193 & 258 & 322 & 386 & 450 \\ 0 & 17 & 65 & 129 & 193 & 258 & 322 & 386 \\ 0 & 0 & 17 & 65 & 129 & 193 & 258 & 322 \\ 0 & 0 & 0 & 17 & 65 & 129 & 193 & 258 \\ 0 & 0 & 0 & 0 & 17 & 65 & 129 & 193 \\ 0 & 0 & 0 & 0 & 0 & 17 & 65 & 129 \\ 0 & 0 & 0 & 0 & 0 & 0 & 17 & 65 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 17 \end{pmatrix} \tag{5.5}$$

5.1.2 Matrix Multiply Results for 8 by 8 Full Matrix

This next example uses a matrix defined in equation 5.6, and designated B. The square of B is denoted B^2 and is shown in equation 5.7.

$$B = \frac{1}{256}\begin{pmatrix} 64 & 128 & 128 & 128 & 128 & 128 & 128 & 128 \\ 128 & 64 & 128 & 128 & 128 & 128 & 128 & 128 \\ 128 & 128 & 64 & 128 & 128 & 128 & 128 & 128 \\ 128 & 128 & 128 & 64 & 128 & 128 & 128 & 128 \\ 128 & 128 & 128 & 128 & 64 & 128 & 128 & 128 \\ 128 & 128 & 128 & 128 & 128 & 64 & 128 & 128 \\ 128 & 128 & 128 & 128 & 128 & 128 & 64 & 128 \\ 128 & 128 & 128 & 128 & 128 & 128 & 128 & 64 \end{pmatrix} \tag{5.6}$$

$$B^2 = \frac{1}{256}\begin{pmatrix} 466 & 450 & 450 & 450 & 450 & 450 & 450 & 450 \\ 450 & 466 & 450 & 450 & 450 & 450 & 450 & 450 \\ 450 & 450 & 466 & 450 & 450 & 450 & 450 & 450 \\ 450 & 450 & 450 & 466 & 450 & 450 & 450 & 450 \\ 450 & 450 & 450 & 450 & 466 & 450 & 450 & 450 \\ 450 & 450 & 450 & 450 & 450 & 466 & 450 & 450 \\ 450 & 450 & 450 & 450 & 450 & 450 & 466 & 450 \\ 450 & 450 & 450 & 450 & 450 & 450 & 450 & 466 \end{pmatrix} \tag{5.7}$$

The ψ^3 expansion of B is denoted $W^3(B)$. The procedure for computing B^2 in a ψ^3 expansion involves the follow:

- Compute the ψ^3 expansion denoted $(W^3(B))$ which is shown in equation 5.8.

- Square $W^3(B)$, which is denoted $(W^3(B))^2$ and shown in equation 5.9.
- Compute the inverse of $(W^3(B))^2$, which is denoted as $W^{-3}((W^3(B))^2)$ and shown in equation 5.10.

$$W^3(B) = \frac{1}{256}\begin{pmatrix} 961 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & -64 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & -63 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & -64 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & -63 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & -64 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & -63 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 & -64 \end{pmatrix} \tag{5.8}$$

$$(W^3(B))^2 = \frac{1}{256}\begin{pmatrix} 3615 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 17 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 17 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 17 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 17 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 17 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 17 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 17 \end{pmatrix} \tag{5.9}$$

$$W^{-3}((W^3(B))^2) = \frac{1}{256}\begin{pmatrix} 466 & 450 & 450 & 450 & 450 & 450 & 450 & 450 \\ 450 & 466 & 450 & 450 & 450 & 450 & 450 & 450 \\ 450 & 450 & 466 & 450 & 450 & 450 & 450 & 450 \\ 450 & 450 & 450 & 466 & 450 & 450 & 450 & 450 \\ 450 & 450 & 450 & 450 & 466 & 450 & 450 & 450 \\ 450 & 450 & 450 & 450 & 450 & 466 & 450 & 450 \\ 450 & 450 & 450 & 450 & 450 & 450 & 466 & 450 \\ 450 & 450 & 450 & 450 & 450 & 450 & 450 & 466 \end{pmatrix} \tag{5.10}$$

$W^3(B)$ has 14 of its elements with an epsilon of $\frac{1}{512}$. Another, seven within an epsilon of $\frac{3}{512}$. Most of the energy is in the second, third, fifth, and last rows. The remaining energy is located in the diagonal. Again nearly, $\frac{7}{32}$ of elements are not likely to contribute anything significant to this multiplication. As for precision, the relative fidelity of $W^{-3}((W^3(B))^2)$ and B^2 is on the order of 10^{-14}.

5.1.3 Matrix Multiplication between Upper Triangular and Full Matrix

This final 8×8 example , the upper triangular matrix A and a full matrix B are multiplied together both in conventional space and in wavelet space. Their results are compared for fidelity, and any notable discrepancies. The upper triangular matrix is the same A used in section 5.1.1 . The full matrix is the same B used in section 5.1.2. Their wavelet transforms are the same as in those sections as well. The results of $(W^3(A) \cdot W^3(B))$ and $W^{-3}(W^3(A) \cdot W^3(B)))$ are described for completeness. The product of A and B is designated C, and is provided for comparison in equation 5.11.

$$A \cdot B = \frac{1}{256} \begin{pmatrix} 466 & 450 & 450 & 450 & 450 & 450 & 450 & 450 \\ 418 & 402 & 386 & 386 & 386 & 386 & 386 & 386 \\ 354 & 354 & 338 & 322 & 322 & 322 & 322 & 322 \\ 290 & 290 & 290 & 274 & 258 & 258 & 258 & 258 \\ 225 & 225 & 225 & 225 & 209 & 193 & 193 & 193 \\ 161 & 161 & 161 & 161 & 161 & 145 & 129 & 129 \\ 97 & 97 & 97 & 97 & 97 & 97 & 81 & 65 \\ 33 & 33 & 33 & 33 & 33 & 33 & 33 & 17 \end{pmatrix} \tag{5.11}$$

The multiplication of A and B in the ψ^3 expansion has a relative fidelity of $2.3 \cdot 10^{-14}$. The product of A and B in wavelet space is denoted $W^3(A) \cdot W^3(B)$, and its inverse is $W^{-3}(W^3(A) \cdot W^3(B))$. The matrix $W(A) \cdot W(B)$ is shown in equation 5.12. Its inverse, $W^{-3}(W^3(A) \cdot W^3(B))$, is shown in equation 5.13.

$$W(A) \cdot W(B) = \frac{1}{256} \begin{pmatrix} 1928 & -16 & -32 & 1 & -64 & 1 & 1 & 0 \\ -240 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ -481 & 1 & 1 & -16 & 1 & 0 & 0 & 1 \\ 1 & 0 & 17 & 0 & 0 & 1 & 0 & 0 \\ -963 & 1 & 1 & 0 & 1 & -16 & -32 & 1 \\ 1 & 0 & 0 & 0 & 17 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 33 & 0 & 1 & -16 \\ 0 & 1 & 0 & 0 & 0 & 0 & 17 & 0 \end{pmatrix} \tag{5.12}$$

$$W^{-1}(W(A) \cdot W(B)) = \frac{1}{256} \begin{pmatrix} 466 & 450 & 450 & 450 & 450 & 450 & 450 & 450 \\ 418 & 402 & 386 & 386 & 386 & 386 & 386 & 386 \\ 354 & 354 & 338 & 322 & 322 & 322 & 322 & 322 \\ 290 & 290 & 290 & 274 & 258 & 258 & 258 & 258 \\ 225 & 225 & 225 & 225 & 209 & 193 & 193 & 193 \\ 161 & 161 & 161 & 161 & 161 & 145 & 129 & 129 \\ 97 & 97 & 97 & 97 & 97 & 97 & 81 & 65 \\ 33 & 33 & 33 & 33 & 33 & 33 & 33 & 17 \end{pmatrix} \tag{5.13}$$

Table 5.1: Peppers 512 × 512 by MRA.

Resolution	Original Energy	Estimate Energy	Fidelity
1	12972.4	12972.4	$1.26464 \cdot 10^{-13}$
2	12972.4	12960.3	3.11211
3	12972.4	12947.8	5.44678
4	12972.4	12939.7	8.43738
5	12972.4	12926	13.1662
6	12972.4	12904.3	19.9257
7	12972.4	12877.9	25.4429

Table 5.2: Waterfall 512 × 512 by MRA.

Resolution	Original Energy	Estimate Energy	Fidelity
1	6630.76	6630.76	$9.37836e - 14$
2	6630.76	6629.92	1.20049
3	6630.76	6626.6	2.37529
4	6630.76	6619.14	3.9194
5	6630.76	6595.14	6.63615
6	6630.76	6579.62	9.48774
7	6630.76	6567.17	12.1211

5.2 Empirical Analysis: MRA

Several standard images were used for measuring the effects of matrix multiplication. Amongst them were the artificial images such as the waterfall and the clock. Others were actual images such as an aerial view of the Pentagon, an in flight photo of an F-16, a fishing boat, an opera house, a river, and a set of bell peppers.

In first round analysis, matrix multiplication was measured against the wavelet pyramids. These showed some undesirable results. As the decomposition proceeded deeper, the fidelity quick became unstable, and made the method unusable for matrix multiplication. The results are shown in Tables 5.1 and 5.2.

Table 5.3: Peppers 512×512 by MRE.

Resolution	Original Energy	Estimate Energy	Fidelity
1	12972.4	12972.4	$1.26464e-13$
2	12972.4	12972.4	0.059622
3	12972.4	12972.4	0.072957
4	12972.4	12972.4	0.0849579
5	12972.4	12972.4	0.09721
6	12972.4	12972.4	0.104528
7	12972.4	12972.4	0.119789

5.3 Empirical Analysis: MRE

The primary example shown is from the Peppers image. Although, all nine of the images used for this thesis could have been used, all of them exhibit similar qualities. After, two resolutions the 2-D MRE becomes unstable for matrix multiplication. It misplaces sections to where random noise in those sections would be just useful as the actual information contained in those sections. The results are shown in Table 5.3.

5.4 Empirical Results: ψ^n

The ψ^n transform clearly produces better results as shown in all examples of the wavelet transform. Trade off issues start to appear when comparing the number of resolution levels to apply as opposed to a spare filter threshold.

In these next examples, resolution levels 1 through 7 are examined by removing data based on energy level. The method used is to strictly remove the lower threshold of energy from the matrix at large. The range of energy thresholds vary from 0.0 to 0.9.

The reasoning for any of these schemes is to establish sparse representations of the matrix. In these sparse representation, only the above threshold elements are used. These representations are useful for sparse matrix multiplication since only the above threshold elements need be multiplied thus reducing the number of multiplications (operations at large) performed.

Lastly, two methods exist to find the threshold value in the array of values. One is to sum up values until the threshold energy level has been reached. Another is to take element relative to the percentage threshold desired. The relative percentage method is simpler, by performing one division on the energy array size and a look up for the value in the energy array.

Once found, the filter threshold is used to filter out elements whose squared value is less than the threshold. Those elements that are less are made to be zero. Those that are above the threshold are left alone.

For this experiment both matrix operands have the filter applied. It should be noted that for the BLAS level three matrix multiplication, this filtering scheme need only be applied to the first matrix operand. Applying this filter to the second is simply unnecessary since the method states that the second matrix can be dense.

To test the ψ^n expansion on a semi-large scale matrices, eight images have been selected. The intensity values of the images used as the values of each matrix element.

Some of the images were produced from photographic images. As such these images contain some noise components which is reflected in the matrices which they populate. The images that are part of this collection are a picture of an US Air Force F-16 fighter jet, a fishing boat, the Opera House at Lyon, and the Pentagon. For the sake of this thesis, these images form a class of images called the real world images. This images are shown in Figures 5.2 ,5.4 , 5.6. 5.8, and 5.10

Other images contributing to the test matrices were part of the International Ray-tracing Competition. As such, they have very little noise from the scene which they represent. Furthermore, such noise is not part of the matrices which these images populate. These images include "A Gold Watch on a Chain," "Always running, never the same...", and M.C. Escher's "Waterfall" rendered by Roger Penrose. For the sake of this thesis, these images form a class of images called the Ray-Traced Images. These images are shown in Figures 5.12, 5.14, and 5.16.

Measurements on each A^2 test compared relative fidelity to general sparseness. For each matrix, test were conducted on seven ψ^n expansions. Tests of each expansion are shown in the results graph.

For both real world image and ray traced image categories, there are few regions of the hold interest. The first section is for low values of relative fidelity, on the order of 10^{-6}. The second section is for moderate values of relative fidelity, on the order of 10^{-3} to the lower 10^{-1}. Last section is for high values of relative fidelity from the upper order of 10^{-1} on up. This is shown in Figures 5.3 ,5.5, 5.7, 5.9, 5.11, 5.13, 5.14, and 5.17.

Both classes of images stay within the low values of relative fidelity for low values of sparseness (zero to five percent sparseness). Ray-Traced images tended to keep relative fidelity in the low values for higher order ψ^n expansions. However, both classes were in moderate values of relative fidelity for fifteen to about sixty percent sparseness.

After sixty percent sparseness, relative fidelity rose at different rates for the two classes of matrices (images). The ray traced class showed sharp rises from the moderate relative fidelity to high relative fidelity between sixty to seventy percent sparseness for ψ^1. The higher order ψ^n expansions the the relative fidelity climbed more gracefully. The most graceful matrix was produced from "Gold Watch on a Chain." Where as matrices produced from "Waterfall' and "Always running, never the same" became high after eighty percent of the matrix was made sparse. This was shown in Figures 5.14 and 5.17.

The above sixty percent sparseness for real world images increased the relative fidelity values smoother than ray-traced images. However, all ψ^n expansions had high values of relative fidelity for eighty percent sparseness and more sparse. This is shown in Figures 5.3, 5.5, 5.7, 5.9, and 5.11.

It is not clear whether gracefulness of these curves has anything to do with how much noise was present in the source. It is clear that a trade off exists between the number of ψ^n expansions versus sparseness levels that determines relative fidelity. In the lower sparseness levels, lower ψ^n expansions have lower fidelity. For moderate to high sparseness values, the moderate to higher order ψ^n expansions have lower relative fidelity values.

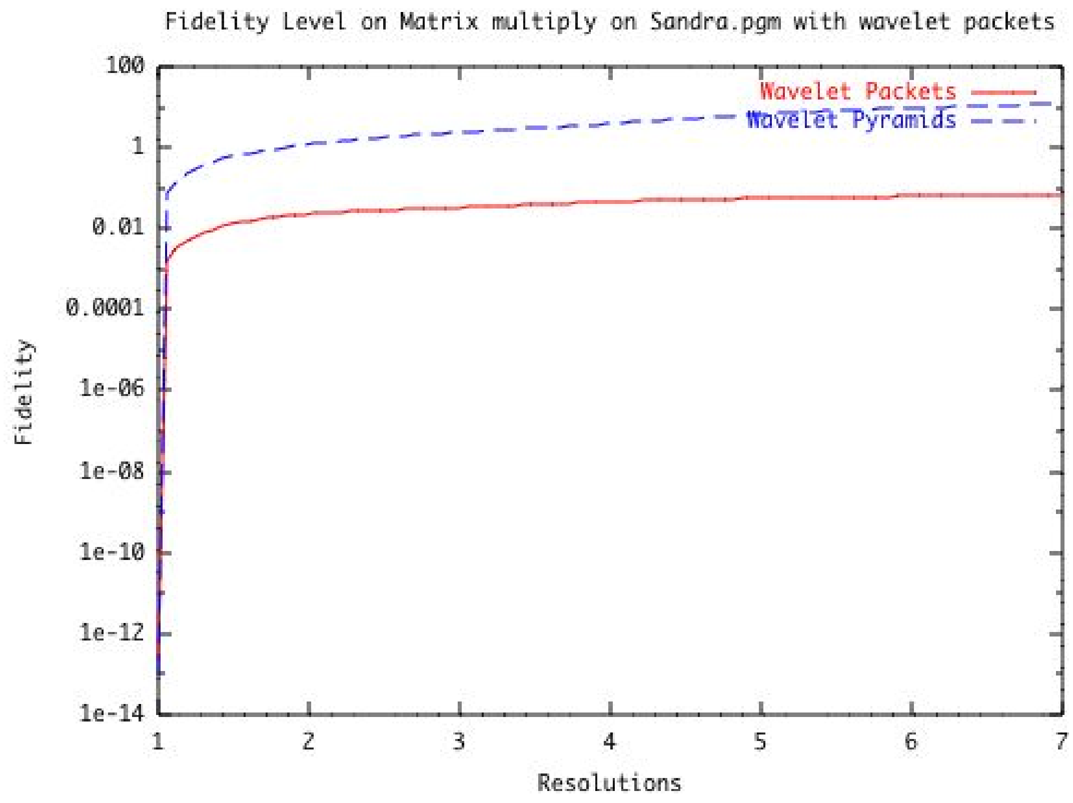

Figure 5.1: This fidelity level graph shows that the wavelet transform packet method as a preconditioner to matrix multiplication is not reliable past the first resolution. The wavelet packet produces marginal results mostly due sections being placed out of order.

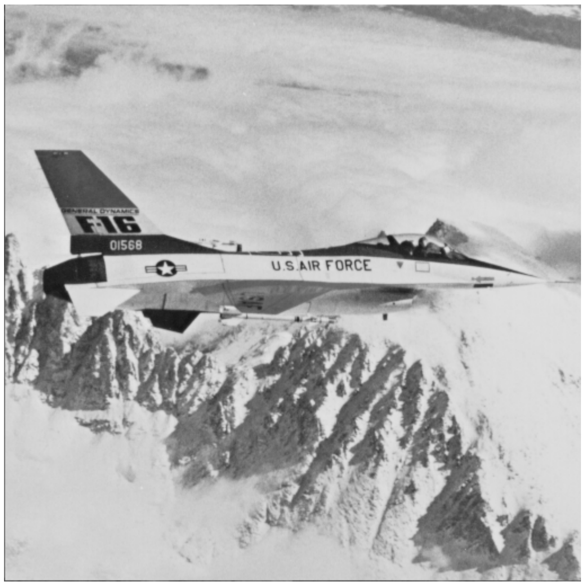

Figure 5.2: This image is a photo of an F-16 fighter jet.

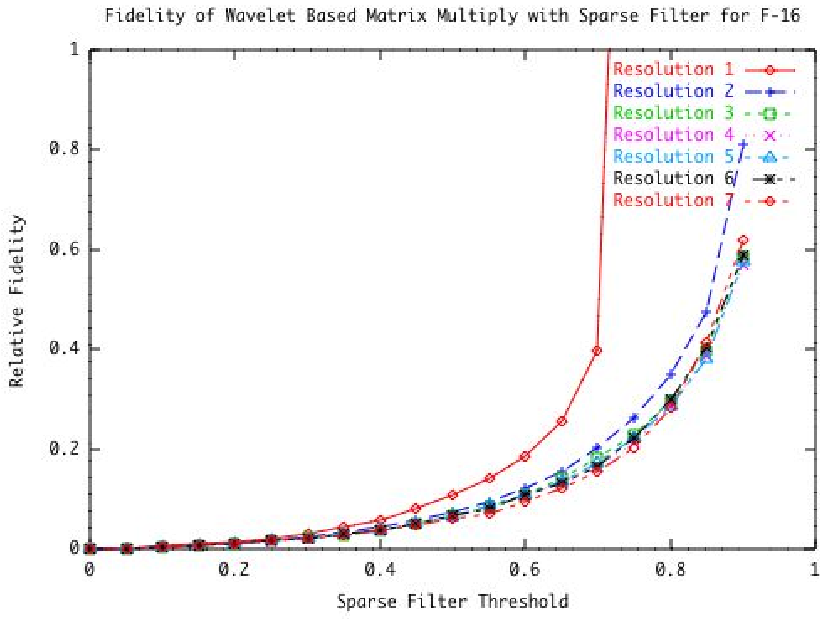

Figure 5.3: This fidelity level graph shows that Psi Wavelet Transform (full decomposition) approximation error in matrix multiplication. This graph in particular was obtained by multiplying the F-16 image by itself with various resolution levels of Wavelet Transforms applied

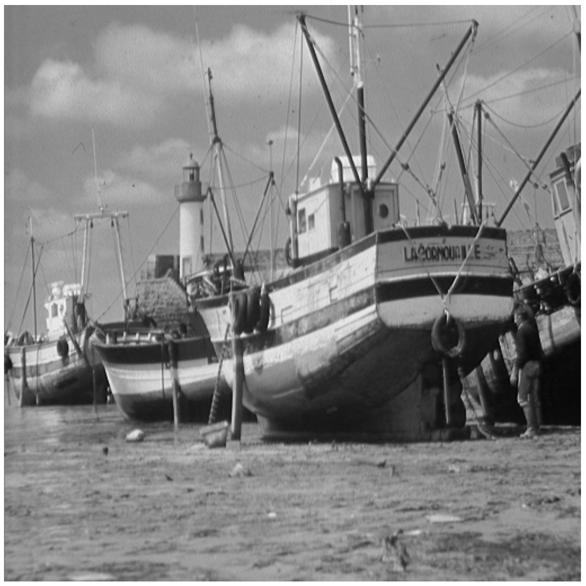

Figure 5.4: This image is a photo of an Fishing Boat which is a standard image.

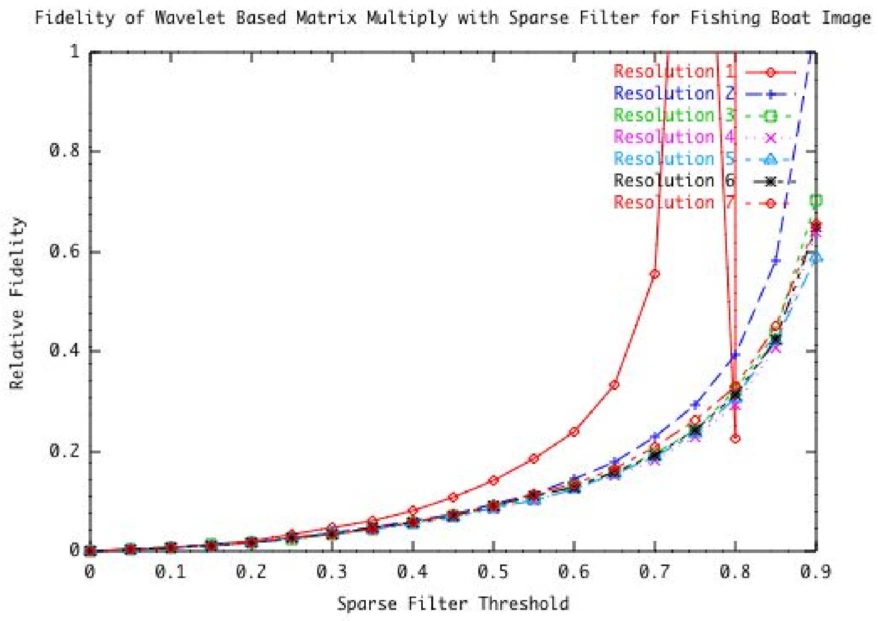

Figure 5.5: This fidelity level graph shows that Psi Wavelet Transform (full decomposition) approximation error in matrix multiplication. This graph in particular was obtained by multiplying the Fishing Boat image by itself with various resolution levels of Wavelet Transforms applied.

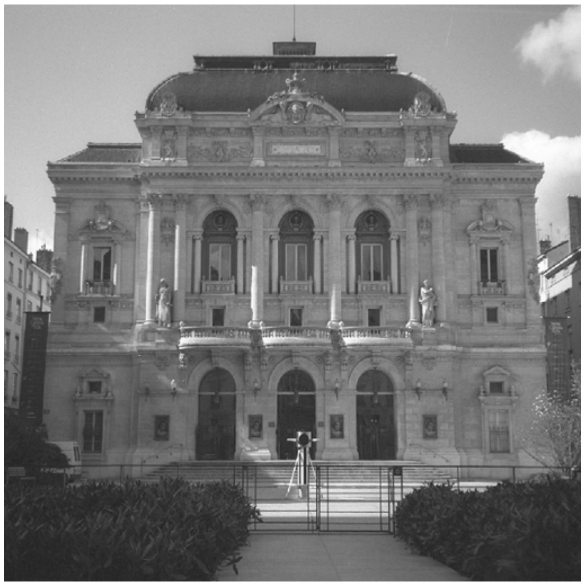

Figure 5.6: This image is a photo of the Opera House in Lyon [1].

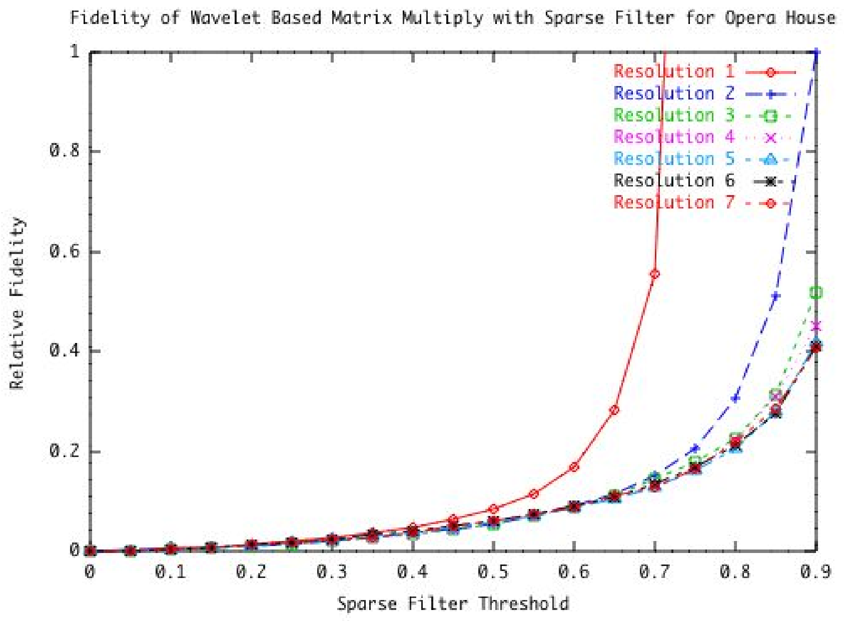

Figure 5.7: This fidelity level graph shows that Psi Wavelet Transform (full decomposition) approximation error in matrix multiplication. This graph in particular was obtained by multiplying the Opera House image by itself with various resolution levels of Wavelet Transforms applied.

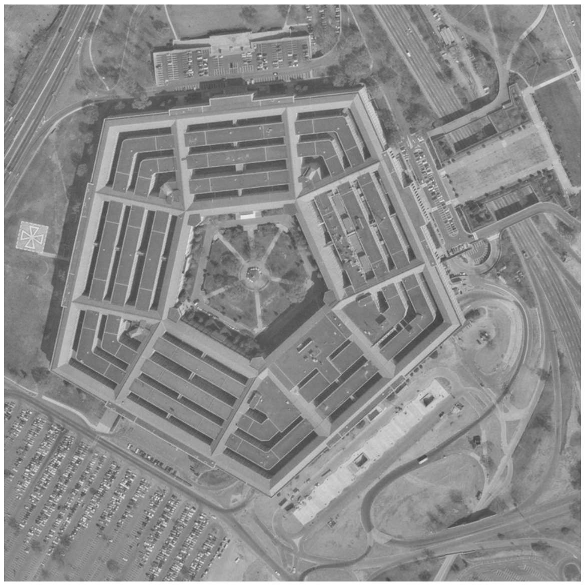

Figure 5.8: This image is an aerial photo of the Pentagon.

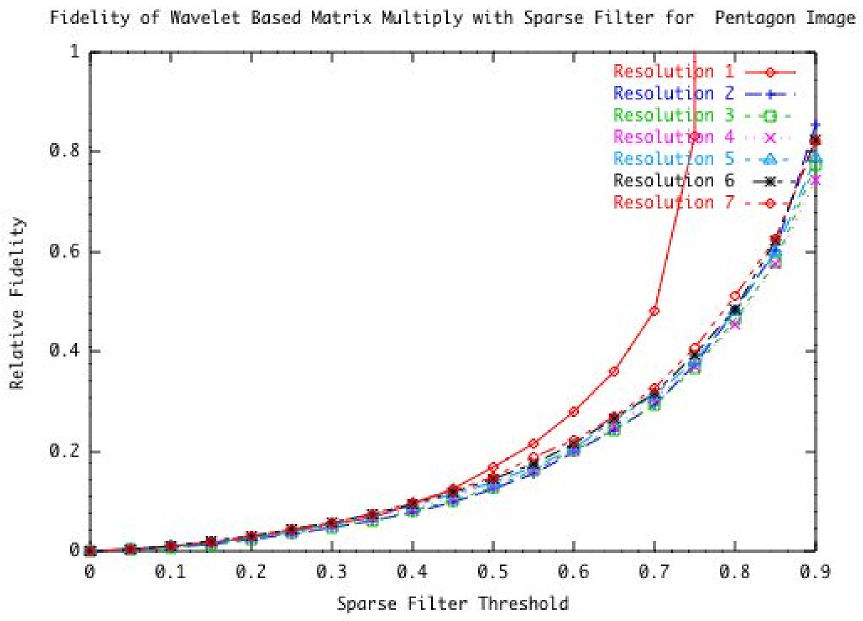

Figure 5.9: This fidelity level graph shows that Psi Wavelet Transform (full decomposition) approximation error in matrix multiplication. This graph in particular was obtained by multiplying the Pentagon image by itself with various resolution levels of Wavelet Transforms applied.

Figure 5.10: This image is of peppers was used to populate a sample matrix used to test matrix multiplication.

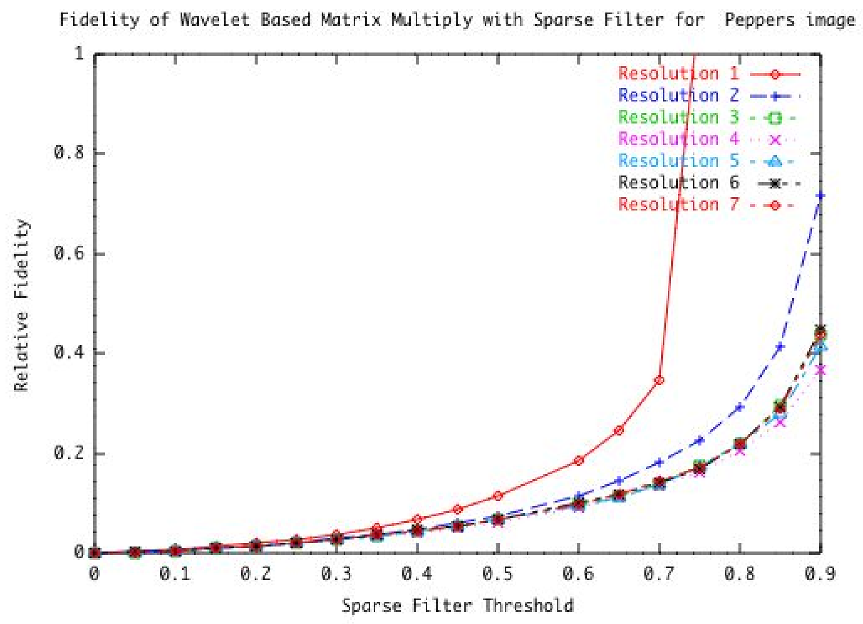

Figure 5.11: This fidelity level graph shows that Psi Wavelet Transform (full decomposition) approximation error in matrix multiplication. This graph in particular was obtained by multiplying the Peppers image by itself with various resolution levels of Wavelet Transforms applied[2].

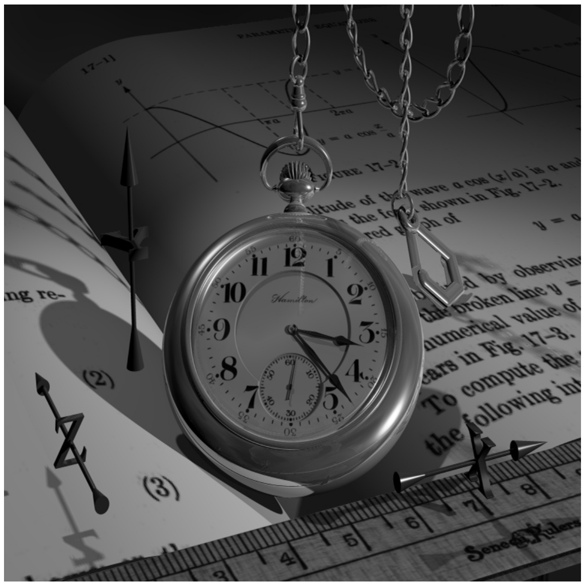

Figure 5.12: “Pocket Watch on a Gold Chain.” Copyright image courtesy of Kevin Odhner[15].

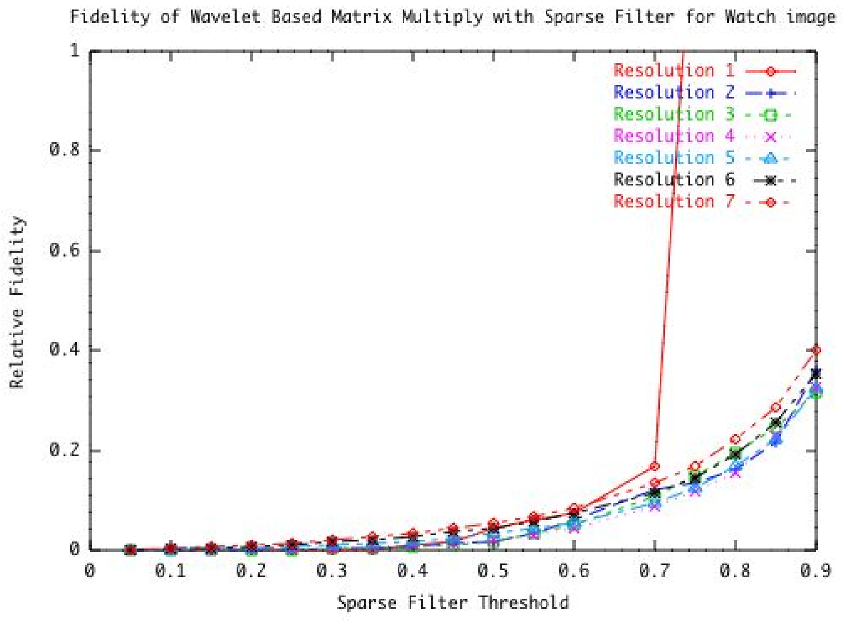

Figure 5.13: This fidelity level graph shows that Psi Wavelet Transform (full decomposition) approximation error in matrix multiplication. This graph in particular was obtained by multiplying the Watch image by itself with various resolution levels of Wavelet Transforms applied [15].

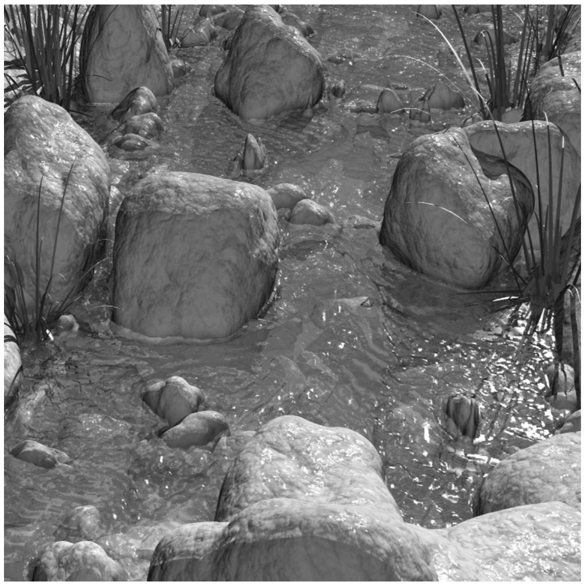

Figure 5.14: This image is an aerial photo of “Always running, never the same....” provided Courtesy of the Jaime Vives Piqueres[16].

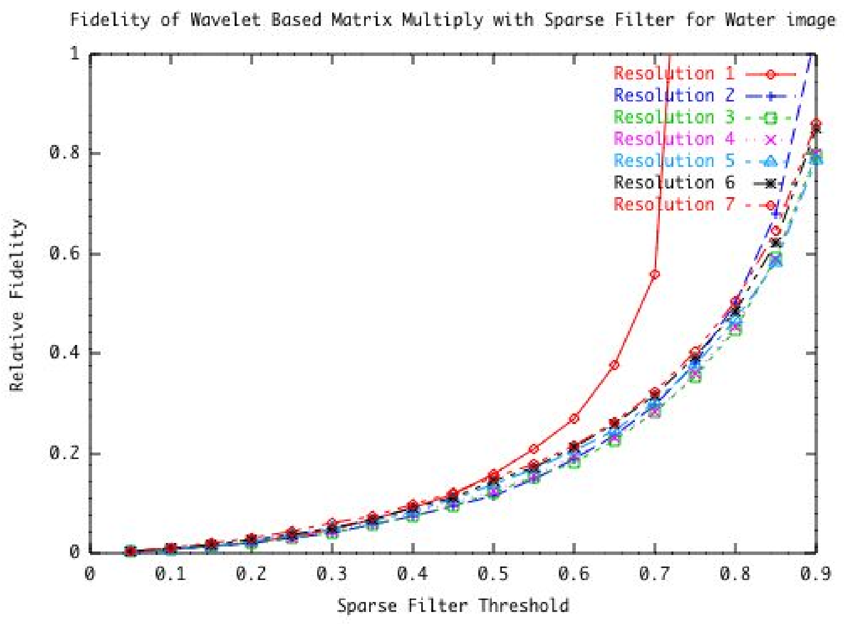

Figure 5.15: This fidelity level graph shows that Psi Wavelet Transform (full decomposition) approximation error in matrix multiplication. This graph in particular was obtained by multiplying the Water image by itself with various resolution levels of Wavelet Transforms applied[16].

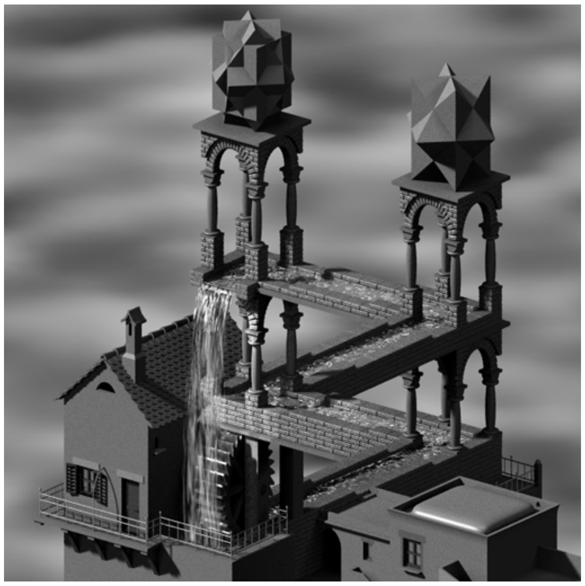

Figure 5.16: Raytraced representation of M.C.Escher's (1898-1972) famous lithography 'Waterfall' (1961), based upon an illusion drawn by Roger Penrose [13].

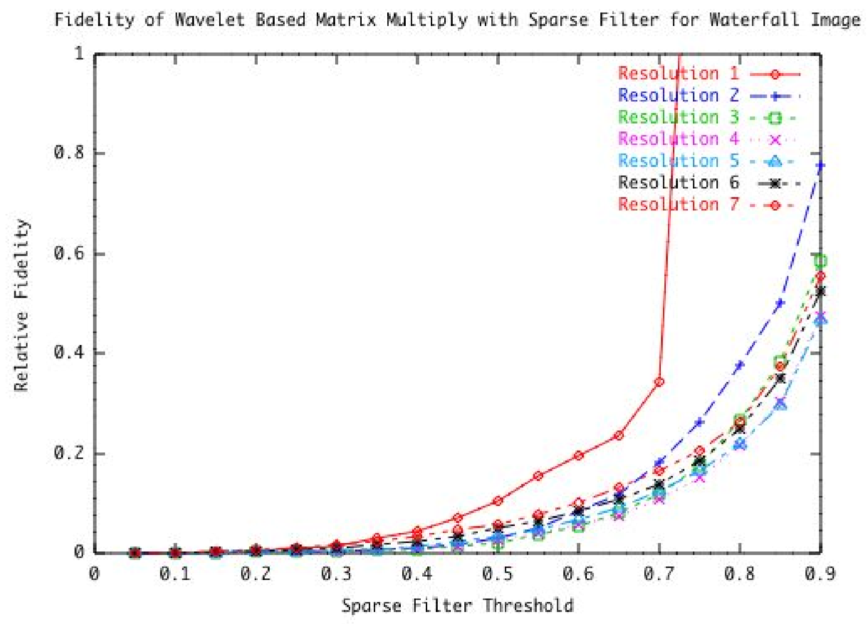

Figure 5.17: This fidelity level graph shows that Psi Wavelet Transform (full decomposition) approximation error in matrix multiplication. This graph in particular was obtained by multiplying the Waterfall image by itself with various resolution levels of Wavelet Transforms applied. [13]

CHAPTER 6

CONCLUSION

The production of this thesis yielded three categories of results. One category was prototypes for the three forms of multi-resolution for wavelet transforms. Second, a proof was generate that satisfactorily shows that matrix multiplication is sound for the Haar Wavelet Transform with ψ^n expansion. Third, empirical results were obtained showing the fidelity of these multiplications.

For each of these categories, three segments were the sub sections for each one. Those sub-sections were the three different forms of multi-resolution for the wavelet transform. For these categories, a strong foundation was shown in the overview, clearly showing what the wavelet transform is and how it can be used.

In conclusion, it is clear that there is a trade off between full wavelet transform expansion and sparse threshold for maintaining an acceptably precise result from matrix multiplication. This thesis may further be used to explore other wavelet basis functions in matrix multiplication. Also, it may be used to explore other numerical methods. Furthermore, it may be interesting to explore wavelets a preconditioning agent for comparing Strassen versus sparse matrix multiplication techniques. However, it is sufficient for this publication to present these methods of wavelet transform, the proof of the Haar Wavelet Transform as a preconditioning agent for matrix multiplication, and empirical results this agent when used in sparse matrix multiplication.

BIBLIOGRAPHY

[1] Opera house of lyon. fabien a. p. petitcolas. open domain.

[2] Peppers. Signal and Image Processing Institute at the University of Southern California. The copyright status of this image is unclear.

[3] R. E. Bank and C. C. Douglas. Sparse matrix multiplication package. 2001.

[4] G. Beylkin. On wavelet-based algorithms for solving differential equations. In *Wavelets: Mathematics and Applications*. CRC Press, 1993.

[5] G. Beylkin. Wavelets and fast numerical algorithms. In *AMS-93 Proceedings of Symposia in Applied Mathematics*, volume 47, pages 89–117, 1993.

[6] G. Beylkin, R. Coifman, and V. Rokhlin. Communications on pure and applied mathematics. Master's thesis, University of Colorado, 1991.

[7] C. Chui. *An Introduction to Wavelets.* Academic Press San Diego, 1992.

[8] G. Fernández, S. Periaswamy, and W. Sweldens. LIFTPACK: A software package for wavelet transforms using lifting. In M. Unser, A. Aldroubi, and A. F. Laine, editors, *Wavelet Applications in Signal and Image Processing IV*, pages 396–408. Proc. SPIE 2825, 1996.

[9] R. C. Gonzolez and R. E. Woods. *Digital Image Processing.* Prentice Hall, Inc. Upper Saddle River, New Jersey 07458, 2002.

[10] A. Graps. Introduction to wavelets. http://www.amara.com/current/wavelet.html. Retrieved March 20, 2004.

[11] S. Huss-Lederman, E. M. Jacobson, J. Johnson, A. Tsao, and T. Turnbull. Strassen's algorithm for matrix multiplication: Modeling, analysis, and implementation. In *Proceedings of the 1996 ACM/IEEE conference on Supercomputing (CDROM)*, page 32. ACM Press, 1996.

[12] B. Jawerth and W. Sweldens. An overview of wavelet based multiresolution analyses. *SIAM Reviews*, 36(3):377–412, 1994.

[13] S. Ledinsky. Waterfall. Internet Raytracing Competition, October 1998. http://oz.irtc.org/ftp/pub/stills/1998-10-31/waterfa1.txt.

[14] S. Lipschutz. *Schaum's Outlines Theory and Problems of Linear Algebra, 2nd Edition.* McGraw Hill New York, 1991.

[15] K. Odhner. Pocket watch on a gold chain. Internet Raytracing Competition, 1998.

[16] J. V. Piqueres. Always running, never the same... Internet Raytracing Competition, October 1998. http://oz.irtc.org/ftp/pub/stills/1998-10-31/running.txt.

[17] G. Tabor. Wavelets - the idiots guide. http://monet.me.ic.ac.uk/people/gavin/java/waveletDemos.html. Retrieved March 20, 2004.

[18] J. Walker. *A Primer on Wavelets and Their Scientific Applications*. Chapman and Hall/CRC, 1999.

[19] E. W. Weisstein. Translation. From MathWorld–A Wolfram Web Resource, 2004. http://mathworld.wolfram.com/Translation.html.

[20] M. V. Wickerhauser. *Adapted Wavelet Analysis from Theory to Software*. Society for Industrial and Applied Mathematics Philadelphia, 2001.