11-28-2000

Digital signal processing implementation for near real-time wavelet transformation system for binary images

Ovidio Alfonso

*Florida International University*

**DOI:** 10.25148/etd.FI13101575

Follow this and additional works at: [https://digitalcommons.fiu.edu/etd](https://digitalcommons.fiu.edu/etd)

Part of the [Computer Engineering Commons](https://digitalcommons.fiu.edu/etd)

Recommended Citation


[https://digitalcommons.fiu.edu/etd/1065](https://digitalcommons.fiu.edu/etd/1065)

This work is brought to you for free and open access by the University Graduate School at FIU Digital Commons. It has been accepted for inclusion in FIU Electronic Theses and Dissertations by an authorized administrator of FIU Digital Commons. For more information, please contact [dcc@fiu.edu](mailto:dcc@fiu.edu).
FLORIDA INTERNATIONAL UNIVERSITY

Miami, Florida

DIGITAL SIGNAL PROCESSING IMPLEMENTATION FOR A NEAR REAL-TIME WAVELET TRANSFORMATION SYSTEM FOR BINARY IMAGES

A thesis submitted in partial fulfillment of the requirements of the degree of

MASTER OF SCIENCE

in

COMPUTER ENGINEERING

by

Ovidio Alfonso

2000
To: Dean Gordon Hopkins  
College of Engineering

This thesis, written by Ovidio Alfonso, entitled Digital Signal Processing Implementation for a near Real-time Wavelet Transformation System for Binary Images, having been approved in respect to style and intellectual content, is referred to you for judgment.

We have read this thesis and recommend that it be approved

Dr. Malek Adjouadi

Dr. Jean Adrian

Dr. Armando Barreto, Major Professor

Date of Defense: November 28, 2000

The thesis of Ovidio Alfonso is approved

Dean Gordon Hopkins  
College of Engineering

Interim Dean Samuel S. Shapiro  
Division of Graduate Studies

Florida International University, 2000
ACKNOWLEDGMENTS

I would like to extend my sincere gratitude to my advisor Dr. Armando Barreto for his guidance and encouragement throughout my Master’s degree at Florida International University. He has been an excellent mentor and a friend.

I would also like to acknowledge the support and direction provided to me by Dr. Jean Adrian, and Dr. Malek Adjouadi during my Master’s thesis work.

Last but not the least I extend my deepest gratitude to my parents, Ovidio P. Alfonso and Edna E. Alfonso, as well as to my sister Dr. Edna T. Alfonso-Vega. They have been a constant source of encouragement and support through all my endeavors so far.

I would also like to acknowledge all of those who have believed in me throughout my scholastic career and I’d to acknowledge the support received for this project from the National Science Foundation grants NSF-CISE-9812636 and NSF-CISE-9906600
Communication has become an essential function in our civilization. With the increasing demand for communication channels, it is now necessary to find ways to optimize the use of their bandwidth. One way to achieve this is by transforming the information before it is transmitted. This transformation can be performed by several techniques. One of the newest of these techniques is the use of wavelets. Wavelet transformation refers to the act of breaking down a signal into components called details and trends by using small waveforms that have a zero average in the time domain. After this transformation the data can be compressed by discarding the details, transmitting the trends. In the receiving end, the trends are used to reconstruct the image. In this work, the wavelet used for the transformation of an image will be selected from a library of available bases. The accuracy of the reconstruction, after the details are discarded, is dependent on the wavelets chosen from the wavelet basis library. The system developed in this thesis takes a 2-D image and decomposes it using a wavelet bank. A digital signal processor is used
to achieve near real-time performance in this transformation task. A contribution of this thesis project is the development of DSP-based test bed for the future development of new real-time wavelet transformation algorithms.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>CHAPTER</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>I. INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>Motivation</td>
<td>1</td>
</tr>
<tr>
<td>Scope of efforts</td>
<td>2</td>
</tr>
<tr>
<td>Assumptions</td>
<td>3</td>
</tr>
<tr>
<td>Organization of Thesis</td>
<td>4</td>
</tr>
<tr>
<td>II. BACKGROUND INFORMATION</td>
<td>5</td>
</tr>
<tr>
<td>General Information</td>
<td>5</td>
</tr>
<tr>
<td>Transformation approach</td>
<td>8</td>
</tr>
<tr>
<td>III. PRELIMINARY WORK</td>
<td>10</td>
</tr>
<tr>
<td>One dimensional cases</td>
<td>10</td>
</tr>
<tr>
<td>Observations from results of the one dimensional case</td>
<td>13</td>
</tr>
<tr>
<td>Extension to the two-dimensional case</td>
<td>13</td>
</tr>
<tr>
<td>IV. SIMULATION IN MATLAB</td>
<td>15</td>
</tr>
<tr>
<td>Reasons for using MATLAB</td>
<td>15</td>
</tr>
<tr>
<td>Structure of system</td>
<td>16</td>
</tr>
<tr>
<td>Transformation and Inverse transformation algorithms</td>
<td>19</td>
</tr>
<tr>
<td>V. INTEGRATION INTO DSP BOARD</td>
<td>25</td>
</tr>
<tr>
<td>Translation of “m” code into C programming language</td>
<td>25</td>
</tr>
<tr>
<td>Performance of the System</td>
<td>27</td>
</tr>
<tr>
<td>Transferring C Code into the DSP Board</td>
<td>33</td>
</tr>
<tr>
<td>Problems Encountered</td>
<td>35</td>
</tr>
<tr>
<td>VI. CONCLUSIONS</td>
<td>39</td>
</tr>
<tr>
<td>General</td>
<td>39</td>
</tr>
<tr>
<td>Thesis Contribution</td>
<td>39</td>
</tr>
<tr>
<td>Future Work</td>
<td>40</td>
</tr>
<tr>
<td>LIST OF REFERENCES</td>
<td>42</td>
</tr>
<tr>
<td>FIGURE</td>
<td>PAGE</td>
</tr>
<tr>
<td>--------</td>
<td>------</td>
</tr>
<tr>
<td>1.1 SIMPLE EXAMPLE OF TRANSLATION OF A WAVELET FUNCTION</td>
<td>8</td>
</tr>
<tr>
<td>1.2 SIMPLE EXAMPLE OF SCALING OF A WAVELET FUNCTION</td>
<td>8</td>
</tr>
<tr>
<td>3.1 SIMPLE 1-DIMENSIONAL EXAMPLES (100 SAMPLES): SYNTHETIC PERIODIC SIGNAL AND VOICE</td>
<td>11</td>
</tr>
<tr>
<td>3.2 MEXICAN HAT AND DAUBECHIES 4 WAVELETS</td>
<td>12</td>
</tr>
<tr>
<td>3.3 ORIGINAL AND RECOMPOSED SIGNAL FOR SIG 1 USING MEXICAN HAT AND DAUBECHIES 4</td>
<td>12</td>
</tr>
<tr>
<td>3.3 ORIGINAL AND RECOMPOSED SIGNAL FOR SIG 2 USING MEXICAN HAT AND DAUBECHIES 4</td>
<td>13</td>
</tr>
<tr>
<td>3.4 STRETCHED MEX HAT WAVELET</td>
<td>13</td>
</tr>
<tr>
<td>3.5 ORIGINAL &amp; RECONSTRUCTED SIGNALS FOR SIG 2</td>
<td>13</td>
</tr>
<tr>
<td>4.1 WAVELET MATRIX COMPOSED BY \textit{matrix_make()} FUNCTION</td>
<td>17</td>
</tr>
<tr>
<td>4.2 FLOW CHART FOR DWITS SYSTEM</td>
<td>20</td>
</tr>
<tr>
<td>4.3 \textit{TRANSFORMATION()} ALGORITHM FLOW DIAGRAM</td>
<td>24</td>
</tr>
<tr>
<td>4.4 \textit{TRANSFORMATION()} ALGORITHM FLOW DIAGRAM (CON’T)</td>
<td>25</td>
</tr>
<tr>
<td>4.5 \textit{INVERSETRANSFORMATION()} ALGORITHM FLOW DIAGRAM</td>
<td>26</td>
</tr>
<tr>
<td>5.1 INPUT SECTION 8 BY 8 OF ICON 2</td>
<td>28</td>
</tr>
<tr>
<td>5.2 &quot;RECONSTRUCTED&quot; SECTION FROM SIMULATION</td>
<td>29</td>
</tr>
<tr>
<td>5.3 &quot;RECONSTRUCTED&quot; SECTION FROM C CODE</td>
<td>29</td>
</tr>
<tr>
<td>5.4 ORIGINAL IMAGE 1</td>
<td>30</td>
</tr>
<tr>
<td>5.5 4:1 FORCED TRANSFORMATION OF IMAGE 1</td>
<td>30</td>
</tr>
<tr>
<td>5.6 2:1 FORCED TRANSFORMATION OF IMAGE 1</td>
<td>31</td>
</tr>
<tr>
<td>5.7 ORIGINAL IMAGE 2</td>
<td>31</td>
</tr>
<tr>
<td>5.8 4:1 FORCED TRANSFORMATION OF IMAGE 2</td>
<td>31</td>
</tr>
<tr>
<td>5.9 2:1 FORCED TRANSFORMATION OF IMAGE 2</td>
<td>32</td>
</tr>
<tr>
<td>5.10 ORIGINAL IMAGE 3</td>
<td>32</td>
</tr>
<tr>
<td>5.11 4:1 FORCED TRANSFORMATION OF IMAGE 3</td>
<td>33</td>
</tr>
<tr>
<td>5.12 2:1 FORCED TRANSFORMATION OF IMAGE 3</td>
<td>33</td>
</tr>
</tbody>
</table>
5.13 RECONSTRUCTED IMAGE 2 WITH MAX ALLOWED MSE OF .125..............................................33
5.14 TRANSFORMATION GRAPH FOR IMAGE 2 MAX ALLOWED MSE .125...........................................34
5.15 RECONSTRUCTED IMAGE 1 WITH MAX ALLOWED MSE OF .125................................................35
5.16 RECONSTRUCTED IMAGE 3 WITH MAX ALLOWED MSE OF .125................................................35
5.17 TRANSFORMATION GRAPH FOR IMAGE 1 MAX ALLOWED MSE .125...........................................35
5.18 TRANSFORMATION GRAPH FOR IMAGE 3 MAX ALLOWED MSE .125...........................................36
5.19 ILLUSTRATION OF THE OPERATION.........................................................................................35
5.20 LARGER TEXT IMAGE WITH A MSE .125.................................................................................39
5.21 MSE OF .0065 TRANSFORMATION RATIO OF 2:1 .................................................................36
5.22 THRESHOLD CHOSEN TOO HIGH.........................................................................................39
5.23 THRESHOLD CHOSEN TOO LOW..........................................................................................40
5.24 3D PLOT OF RECONSTRUCTED IMAGE BEFORE fix_output() IS APPLIED.....................41
5.25 WAVELET 1 FROM LIBRARY.................................................................................................42
5.26 WAVELET 2 FROM LIBRARY.................................................................................................42
Chapter 1

Introduction

This chapter presents an introduction to Wavelets as well as an introduction to image transformations. Also outlined is the objective of the present research performed.

1.1 Motivation

Entering the 21st century, we find ourselves immersed in a world of technology. It could be argued that one of the primary missions of this technology is the propagation of information throughout the world. Information is constantly being exchanged; from a simple voice message to streams of real-time data across computer networks. This allows the exchange of ideas over great distances with absolute clarity.

All this connectivity however, comes at a cost. The frequency spectrum is finite, and with the constant emergence of new applications requiring frequency bandwidth, maximization of the current bandwidth is of prime importance. Many techniques are used to maximize bandwidth utilization. One of these approaches is the transformation of data prior to transmission. Transformation can be achieved through several methods. One of the relatively new methods is Wavelet transformation.

Though there has been work done in the transformation of data for potential compression, little work has been done towards the application of Wavelets transformations in real-
time. The hope of this thesis is to provide an experimental exploration into the prospective practical use of Wavelets for this application.

1.2 Scope of Efforts

The study of Wavelets is vast and extensive. In order to contribute something meaningful in this area of study it was necessary to narrow the scope of this application. For example, a two-dimensional wavelet can be applied to an image or a one-dimensional wavelet can be applied throughout an image. In order to decide what direction the research was to go, it became necessary to specify a particular area of application to be targeted by this investigation on Wavelet transformation.

First, the type of transmission medium needed to be selected. In today’s fast paced world of mobile communications, wireless seemed a worthy target for this endeavor. Next, the type of wireless transmission needed to be chosen. These choices range from a one dimensional bit stream to real time streaming video. This leaves a wide range of areas to choose from. The intent of this thesis project is to take a first step towards the protocol used for compression using Wavelet transformation in data communications. Although the eventual path of this application is toward streaming video, an application such as that, was determined to be too ambitious for this first exploration. Considering this, the transformation of static images was chosen.

Images, however, exist in a wide variety of types, ranging from simple black and white icons to detailed high-resolution images. Therefore, the scope of the work was further
limited. Since the current move in technology is wireless telephony, this application would be best suited for something along these lines. The transmission of simple black and white icons sent to a cellular phone was selected as a good first goal from which future work could be pursued. This thesis approaches that objective by applying 1-dimensional wavelets, repeatedly, to the input images. It also supplies the infrastructure for more complicated systems. Also, limiting the value of the images to integer values will simplify the implementation. Of course this will affect the overall level of data size modification achieved but it will allow the system itself to be developed.

1.3 Assumptions

Having narrowed the scope of this thesis, certain assumptions can be made. First, because of the assumption that the image is zero or one it causes the reconstructed image's pixels to be cast to zero or one. Overall, this assumption decreases the complexity of the problem and allows the reduction of the size of the wavelet library used in this first exploration.

Because of this assumption, the strategy used to apply the wavelet transformation on to the image is also determined. The maximum difference between one pixel and its nearest neighbor is one; therefore a one-dimensional wavelet applied to each line of the image seems to be a suitable approach. A two-dimensional wavelet would be more suitable for an image that contains shades of gray. This topic is cover in depth in Chapter 3.
Another assumption is that the image will be digitized by another system and the data will be presented to the wavelet transformation system in a 128 by 128 pixel size. In addition, the wavelet library will be available at the receiver side, which in this case would be the cell phone or some other wireless device.

1.4 Organization of the Thesis

The organization of this thesis is as follows:

- **Chapter 1** presents a general overview of the topic. Included in this chapter are the objective and the scope of the research. It also describes the general motivation behind the thesis as well as the assumptions made possible by narrowing the scope of the topic.

- **Chapter 2** introduces the general concept of Wavelets as well as the basic approach used.

- **Chapter 3** presents the preliminary work done in the one-dimensional case and discusses the results obtained. In addition, the extraction of these results to the 2-dimensional case is discussed.

- **Chapter 4** introduces the simulation environment used, the overall system structure, and gives details about the structure of the transformation and inverse transformation algorithms.

- **Chapter 5** outlines gives the steps taken in translating an original MATLAB script implementing into a C program and later porting that program into the DSP board for on-line performance. It will also discuss performance of the system, problems encountered and the subsequent limitations of the system.
• *Chapter 6* presents the conclusions reached and summarizes the thesis with proposals and suggestions for further studies.
Chapter 2

Background Information

This chapter presents a brief introduction to the general theory behind Wavelets as well as the overall approach used.

2.1 General Introduction

Compression through transformation has become a major aspect of today’s modern communication systems. Currently most image compression is done with fixed transforms such as the Discrete Cosine Transform (DCT) or the Haar transform and, in most cases, these transforms perform well [3]. However, there are occasions where these transforms fall short of the desired performance. In these situations a transform that could be chosen for the particular signal or image to compress, based on a calculation of an error margin is desirable [17]. In the proposed project individual wavelet transforms will be contained in a library and applied to an image or part of an image based on a performance competition dealing with each image, specifically [7].

To better understand the concept of wavelets, it will be useful to examine the Fourier transform relationships: [2]

\[ X(f) = \int_{-\infty}^{\infty} x(t) e^{-j2\pi ft} \quad \text{eq 1.1} \]
\[ x(t) = \int_{-\infty}^{\infty} X(f) e^{-j2\pi ft} \]

\text{eq 1.2

Considering this transformation from a practical standpoint, it can be seen that equation (1.1) cannot be implemented for an arbitrary waveform, since the integral is defined from \(-\infty\) to \(+\infty\)\[14\]. Thus, these types of equation are only accurate when describing a periodic signal that has been in existence and shall continue to be in existence indefinitely. This, however, is not typically the case for signals processed by practical communications systems\[13\]. Any variation from these criteria will lead to a broadening of the spectrum and an uncertainty about the actual frequency composition of the signal. This leads to the creation of a function \(\Psi(x)\), where \(x\) is the independent time or space coordinate, which can be scaled and translated to allow for analysis at various locations and levels of resolution as shown in figures 1.1 and 1.2 \[3\].

\[ \Psi_s(x) = \alpha \Psi(s,x) \]

\text{eq 1.3

\(\Psi_s(x)\) is the scaled version of \(\Psi(x)\) and \(\alpha\) is a term added to allow for energy invariance over the range of scales used. The wavelet transform of a signal \(f(x)\) is then the inner product

\[ Wf(s,u) = \langle f(x), \Psi_s(x-u) \rangle \],

\text{eq 1.4

where \(\Psi_s(x-u)\) is the translated version of \(\Psi_s(x)\).}
What this function $\psi_s(x)$ allows, is the representation of a transient signal in terms of a particular wavelet [20]. With this wavelet representation, certain signals can be easily compressed. The compression and re-composition can be represented by the following equations:[6]

\[ C = \psi_s \ast R \]  \hspace{1cm} \text{Eq 1.4}

\[ Rr = \psi_s' \ast C \]  \hspace{1cm} \text{Eq 1.5}

where, $R$ is original data vector, $Rr$ is re-composition of $R$ vector, $\psi_s$ is the Wavelet, $C$ is the compression coefficients derived from the convolution of $\psi_s$ and $R$, $\ast$ symbolizes the convolution operator, $'$ symbolizes the transposition of a vector.
Once the wavelet coefficients (Eq 1.4), which represent the signal, are calculated, they can then be encoded using many different techniques for their proper transmission over the communication channel. Although such techniques are beyond the scope of this thesis, the need for the encoding step before transmission should be acknowledged [4].

A body of work already exists in the area of wavelets. In fact, Wavelets have been used in a variety of fields from seismology to the monitoring of oscillations in engine blocks [18]. Despite the growth in these areas of research, not much work has been done in applying a wavelet transform to a given image in near “real-time”. Such task is the main objective of this thesis. In this work, an image is loaded into the DSP and processed, using a cost function to determine (from a library of wavelets) which wavelet can best describe the image [7]. This wavelet will then be applied to the image to yield the coefficients, which describe the image according to the chosen wavelet. The focus of this thesis will be on the implementation of this fundamental transformation system in a DSP processor [9]. The performance can be improved in the future by the addition or modification of the basic algorithm to be developed in this thesis.

2.2 Transformation approach

The DSP Wavelet Image Transformation System or DWITS will be composed of a DSP real-time processing board, which will run the DSP implementation of the algorithm. The software will take a preprocessed section of a gray-scale image and load it into the DSP board. The DSP code will then apply the first wavelet in the library to the image
section being processed. Once this has been completed the transpose of the original wavelet is applied to the calculated coefficients [5]. At this point the image segment will be reconstructed from the compressed information, and the mean squared error will be calculated for the given transform, using the following equation:

\[
M = \frac{\sum_{x=1}^{m} \sum_{y=1}^{n} (X(x,y) - Y(x,y))^2}{m \cdot n}
\]

where,
- \( M \) is the mean squared error,
- \( X(x,y) \) is the original image,
- \( Y(x,y) \) is the reconstructed image,
- \( m \) size of section in x direction (pixels),
- \( n \) size of section in y direction (pixels)

\( M \) is then compared with the current “lowest value” of mean squared error recorded for that particular image section [11]. This continues until all wavelets in the library have been tried. Once the end of the library has been reached, the set of coefficients that corresponds to the lowest mean squared error are transmitted along with a set of bits that indicate the corresponding wavelet used. The next section of the image is then loaded into the DSP and the process continues.

It is apparent that the quality of the reconstructed image obtained from the DWITS depends on the complexity of the section chosen, which is in turn affected by the way in which the image is partitioned [12]. Although the DWITS would benefit from an adaptive sectioning algorithm, it has been decided in the interest of time, to limit the scope of this thesis to the implementation of the wavelet transformation using a library of wavelets, without pre-processing [19].
Chapter 3

Preliminary Work

This chapter presents the preliminary work done in the one-dimensional case and the initial extension of the approach to the 2-Dimensional case.

3.1 One dimensional cases

All preliminary analysis and results shown below were simulated in MATLAB® [8]. A one-dimensional transformation system was simulated in order to establish that the concept was realizable. A series of signals were generated to serve as input signals in order to test the feasibility of the system, for a one-dimensional prototype [10].

The first signal (Signal 1) is composed of an addition of the periodic functions \( \sin \left( \frac{2\pi n}{8} \right) \) and \( \cos \left( \frac{2\pi n}{4} \right) \). The second (Signal 2), is a 100-point sample of a voice recording collected at a sampling rate \( f_s = 27.5 \) kHz. The wavelets chosen for the

![Fig 3.1 Simple 1-dimensional examples (100 samples): Synthetic periodic signal and voice recording](image)

The first signal (Signal 1) is composed of an addition of the periodic functions \( \sin \left( \frac{2\pi n}{8} \right) \) and \( \cos \left( \frac{2\pi n}{4} \right) \). The second (Signal 2), is a 100-point sample of a voice recording collected at a sampling rate \( f_s = 27.5 \) kHz. The wavelets chosen for the
preliminary testing on these signals were the Daubechies 4 [15] and the “Mexican Hat” [16] (Fig 3.2).

![Fig 3.2 Mexican Hat and Daubechies 4 wavelets](image)

After transformation and inverse transformation, the original and reconstructed signals were plotted together and the mean square error was calculated. (Fig 3.3)

![Fig 3.3 Original and recomposed signal for Sig 1 using Mexican Hat and Daubechies 4](image)

Using the Mexican hat and the Daubechies 4 wavelet on Signal 1, the mean squared error was calculated to be 0.1594 and 0.0228, respectively. These results, as well as the similarity of the original and reconstructed signals in Fig 3.3 show the viability of the process for this signal. These same wavelets were also applied to Signal 2.
In this case, the mean squared error was calculated to be 0.0503 for the Mexican hat and 0.0507 for the Daubechies. From observing figure 3.3 it is apparent that these two wavelets are not capable of representing the most significant frequencies of the voice sample. However, when the Mexican hat wavelet, is “slowed down” (stretched in time) and scaled (fig 3.4), the results are as shown in figure 3.5:
This modification of the wavelet yields a significantly lower mean squared error of 0.0323.

3.2 Observations from Results of the One-Dimension case

The previous discussion yields important observations. The stretching and scaling a mother wavelet can be used to obtain a better approximation of the data. After reviewing the assumptions made in Section 1.3, several other observations can be made. The quality of the overall transformation is dependent on the wavelets included in the library. The library in the proposed system includes wavelets that are stretched as well as scaled. This factor increases the overall complexity of the DWITS, but it is also likely to enhance the accuracy of the reconstructed images obtained from the system.

3.3 Extension to the Two-Dimension case

From the discussion in section 3.1, and keeping in mind the assumptions made, it is possible to propose an extension of the method to the 2-dimensional case. As previously stated, since the images being compressed are destined for cell phone LCD displays, the possibility of gradual transitions from black to white or vice versa are eliminated. This eliminates the need for slow frequency wavelets such as the Mexican-Hat. The transition from one pixel to the next will be immediate; therefore a Wavelet such as the Haar should be most effective. This will be confirmed in the 2-dimensional simulation by simply providing a variety of wavelets in the library and letting the system chose the most appropriate one for each image segment processed. In addition, a 2-dimensional wavelet
can be expected to be ineffective because of these same sharp (high frequency) transitions. A line-by-line sequential application of a 1-D wavelet executed throughout the image in both the x and y directions are likely to be more effective. Unavoidably, the problem of lost information arises after transformation is done in both the x and y-axis. When all the rows have been compressed, through application of the 1-D wavelet to each of them, a new image is generated. In essence, half the data in one direction has been removed, so when the column transformation commences it is actually performing a transformation of this new data, not the original image column. This can lead to the loss of data. With some images this may become a significant loss, so a threshold must be set. This threshold will be used to determine whether transformation of the section will take place in the x and y direction, or just the y direction, or just the x direction. This will however, cause the reduction of the compression of the number of element in the resulting matrix. Thus, a balance between matrix compression and recovered error must be obtained. This will become evident only after the simulation of the entire 2-dimensional system, introduced in the next chapter.
Chapter 4

Simulation in MATLAB

This chapter presents the reason for simulating in MATLAB as well as the structure of the system and the results obtained.

4.1 Reasons for Using MATLAB

When deciding how to simulate the DWITS system it was import to keep in mind the eventual goal of the thesis, the implementation of this system in a DSP board. Knowing that the final form of the system is assembly code, it was important to decide at what level programming would be most effective. Programming in assembly would be the best way to efficiently program the transformation and inverse transformation algorithms but this would require a large amount of time and increase the complexity of the thesis unnecessarily. C programming language could be translated by the help of software into assembly. This simplified the programming of the wavelet algorithms. Although C is easier than assembly code, it is far too cumbersome for trail and error experimentation. MATLAB easily lends its self to this type of experimentation. MATLAB allows the programmer to call functions without worrying what types are being passed or returned [8]. In addition, linear algebra in MATLAB is much easier than in C programming language. C cannot simply multiply two matrices in a single instruction; it must multiply the two element by element [8]. This translation from MATLAB to C has been done, but using it for the initial simulation would have proved to be too time consuming.
4.2 Structure of System

The DWITS system structure is composed of two primary functions and three support functions. The primary functions are the *transformation()* algorithm, which transforms an 8X8 section of the original image, and the *inverse_transformation()* algorithm, which recovers the original section from the coefficients. The support functions include the *matrix_make*, *fix_input*, and the *fix_output*. The *fix_input* function takes the raw image and formats it to discrete values. This function also inverts the values of the pixels. This was done to ensure the input would be compatible with the DWITS system. The *fix_input* function also filters the input image of any errors that may have been introduced when the image is brought from a third party application. The *fix_output* function is similar to the *fix_input* in that it formats the output into discrete values; the difference is that it has an adjustable threshold to allow for fine-tuning. This value could have only been found through trial and error. Its primary function is to increase the accuracy of the system. The *matrix_make* function is a function that takes a mother wavelet and convolves it through a 4X8 matrix. This matrix is the tool with which the compression is done. The matrix is as follows, where A1 thru A4 are the coefficients of the wavelet.

```
A1  A2  A3  A4  0   0   0   0
0   0   A1  A2  A3  A4  0   0
0   0   0   0   A1  A2  A3  A4
A3  A4  0   0   0   0   A1  A2
```

Fig 4.1 Wavelet matrix composed by *matrix_make()* function
The coefficients of the wavelet may range from 4, as shown above, to as many as 8. The coefficients are stored in the library and are passed to the matrix_make function on an individual basis.

The DWITS system begins by taking a raw 128x128 pixel image and passing it to the fix_input function. This function, as previously described, will format the image as to aid in the transformation of the image. Next, an 8x8 pixel section is scanned from the image. This scanning process is termed "tiling" and is done to limit the section to which the wavelet is applied. This new "fixed" section is then passed to the transformation algorithm. Here the wavelet is applied to each line in the x and y of the section. The details of the operation as well as the details of the inverse transformation algorithm will be described in section 4.3. Once returning from the transformation algorithm the data should be in ether, a 4x4 transformed matrix, an 8x4 partially transformed matrix, or a 4x8 partially transformed matrix. This data, as well as the wavelet used and transformation degree will be passed to the encoding unit. At this point in the simulation, the data is merely transformed not encoded. The option of encoding the data has been left for future implementations that may actually transmit the images through a communications channel and will not be considered here.

Once the data has been received by the inverse_transformation() algorithm, the wavelet indicated will be applied according to the degree of compression specified. This recovered section is then stored into a "holding" matrix. The entire process repeats until the complete image has been received and retransformed into its original state.
fix_output function is then applied to the recovered image. This function applies a threshold to the image, riding it of non-discrete values.
Fig 4.2 Flow chart for DWITS system
4.3 Transformation and Inverse transformation Algorithms

The transformation algorithm begins by calling the Make_matrix() function. This creates the wavelet matrix that is used for the transformation of the section. The matrix is then multiplied by the first line of the X-axis (i.e. the first column of the image segment). This yields a 4x1 vector. This is the wavelet representation of the input vector in terms of the wavelet applied. In other words, this is the approximation of the input vector. The details are discarded when the multiplication takes place. This is repeated for all the other columns, until all the columns have been multiplied and reduced to four coefficients. The 4x8 matrix generated is stored and is termed hold_matrix_X. This matrix is used in deciding whether the algorithm will transform the input section in the x and y directions or just the y or x directions.

A second 8x4 matrix termed hold_matrix_Y is also generated for transformation in the y direction. Now the convolution wavelet matrix is applied to the 4x8 hold_matrix_X in the Y-direction. This yields the hold_matrix_XY with only 4 rows and 4 columns, which yields a 4:1 compression of the elements of the input matrix. Once all three transformations have been obtained, the algorithm begins to retransform the 4x4-transformed matrix in the Y-axis. This is done by multiplying the transpose of the wavelet matrix to each 4x1-transformed vector. This will yield a “reconstructed” 8x1 vector. This operation is then repeated for all columns and then rows. The Y-axis must be done first because the image was compressed in the Y-axis last. The inverse transformation must “undo” what the transformation algorithm has done. Thus, the reverse steps must be taken in order to retrieve the original section.
The same process is then repeated in the X-direction once the rows have been transformed back to their original state. This retransformed matrix is termed \textit{Recovered}_2D, and will be needed later. The reverse transform is also applied to both the \textit{hold} \textit{matrix} \textit{X} and \textit{hold} \textit{matrix} \textit{Y}. Since these sections were only transformed in the one direction, it is not necessary to retransform the sections in both the X and Y directions. These operations will also yield two other reconstructions of the original section, \textit{Recovered} \textit{1D} \textit{Y} and \textit{Recovered} \textit{1D} \textit{X}.

To determine which transformation will effectively be used, a way to compare the quality of the three is necessary. The mean squared error or MSE between \textit{Recovered} \textit{1D} \textit{Y}, \textit{Recovered} \textit{1D} \textit{X}, and \textit{Recovered} \textit{2D} and the original section is calculated. If the mean squared error calculated for the 2D case is less than the \textit{Current}_MSE \textit{2D} then the \textit{Current}_MSE \textit{2D} is assign the new value. Now the MSE for both 1-D cases are calculated. If the algorithm has used all the wavelets in the library and has calculated the MSE for all cases, the algorithm will move on. Otherwise, the algorithm repeats until this has been completed. The final lowest 2D MSE is compared to the threshold, which was passed into the transformation function. If the MSE is lower than the threshold then the 4x4-compressed matrix is returned along with the corresponding wavelet identifier and transformation direction. If this value is larger then the threshold, the 4:1 element compression ratio is decided to be too lossy and either the Y 2:1 element compression ratio or the X 2:1 is returned depending on which has the lowest MSE. The way in which the wavelets are chosen for the 1D case and 2D case are
independent. The algorithm may choose on wavelet for the 2D case and another for the 1D case. The algorithm is allowed to choose whichever has the lowest mean squared error.

The inverse transformation algorithm works very similarly to the second half of the transformation algorithm. However, since the wavelet used, the transformation direction, and the transformed data are passed into the function, the function has to only compute the results. Once it begins, the algorithm calls the Make_matrix() function, passing the wavelet used, to obtain the wavelet matrix. The matrix is then transposed to obtain the reverse wavelet matrix. A decision is made as to the value of the element compression ratio. If it is 4:1, the transposed wavelet matrix is applied to all lines in the Y and X direction, respectively. If the value is 2:1 then only one direction needs to be multiplied to the transposed wavelet matrix line by line, depending on the transformation_direction parameter. The “reconstructed” matrix is returned to the main function. Since the inverse_transformation() function does not have to scroll through a library or calculate the MSE for each wavelet for both the 1D and 2D cases, it runs much faster than the transformation() function.
Set $W'$ applied to $XY$ in $Y$ direction. Results in $R1$.

$M_{aike} = \text{entire section been applied through inverse transformation}$

$W'$ applied to $RX$ in $X$ direction. Results in $RY$.

$\text{Note - All bold letters are matrices.}$

$\text{Fig 4.3 transformation() algorithm flow diagram}$
Fig 4.4 transformation() algorithm flow diagram (con't)
Fig 4.5 inverse_transformation() algorithm flow diagram
Chapter 5

Integration into DSP Board

The chapter will describe the process of translating the m code of MATLAB into C code and then transferring that code into the DSP board. It will also mention the problems encountered and solutions to those problems.

5.1 Translation of M code to C Programming Language

Now that the simulations are complete and that appropriate parameter values have been obtained, the next step can begin: translating the MATLAB m-code into C programming language. The translator, which is used for translating C into assembly for the DSP board, accepts basic C. When translating the m-code into C it is important to realize certain operations, such as assigning the value of a matrix to another, are not as simply done in C. For example, the simple line “MATRIX_B = MATRIX_A * 2;” must be done in an element-by-element manner in C.

```
for (int x=0;x<8;x++)
{
    for(int y=0;y<8;y++)
    {
        MATRIX_B[y][x]=MATRIX_A[y][x];
    }
}
```

That simple line is expanded to 5 in C programming language. In addition, simple MATLAB functions such as transposing a matrix must be done step by step in C. For this reason, function such as Matrix_Transpose(), Multiply1() (which multiplies a 8x1
vector by a 4X8 matrix), and Multiply2() (which multiplies a 8x1 vector by a 8x4 matrix) must be written.

Once the functions have been coded, they need to be tested. In order to do this, a 16x16 section is taken from icon 1 of the simulation. This data is written to a text file where it is read in by a C++ "wrapper" program. This program will call the transformation and inverse transformation algorithms and then write the output data back to a text file. Figure 5.1 shows the input section taken from icon 2. Figure 5.2 shows the section after it has been through the transformation and inverse transformation sequence. Figure 5.3 is the output from the C code. The two outputs are identical to each other. Therefore, it can be assumed that the C functions are coded correctly and the next step can be undertaken.
In order to get an idea of the quality of transformation achievable it is necessary to force the algorithm to compress the elements entire of an input matrix by a 4:1 ratio. This will allow the calculation of the absolute worst possible mean squared error for the system. Likewise, forcing the simulation to compress the elements of an input matrix by a 2:1 will obtain the absolute minimum mean squared error the simulation can achieve with the wavelet library at its disposal. Figures 5.5, 5.8, and 5.11 are the 4:1 total transformation
results for icons 1, 2, and 3 respectively. Figures 5.6, 5.9, and 5.12 are the 2:1 total transformation results for icons 1, 2, and 3.

Fig 5.4 Original Image 1

Fig 5.5 4:1 forced transformation of Image 1
Fig 5.6 2:1 forced transformation of Image 1

Fig 5.7 Original Image 2

Fig 5.8 4:1 forced transformation of Image 2
Fig 5.9 2:1 forced transformation of Image 2

Fig 5.10 Original Image 3
The simulation was run using several different maximum squared error levels for three different test images. Each image was loaded into a 128x128 matrix. From there an 8x8 section is passed to the `transformation()` algorithm along with the maximum squared error. The images were chosen to represent simple icons that may appear on a LCD type of display. Entering figure 5.7 into the simulation with a maximum allowed mean squared error of .125 yields the image shown in figure 5.13.
The overall mean squared error for the reconstructed image is .0074. It can be seen that the image is reasonably close to its original Fig 5.12. The transformation direction plot is a graphical representation of transformation level was used as well as whether transformation was done solely in the X or Y. When looking at the transformation direction used plot (Fig 5.14) generated, it can be seen that the image is almost entirely composed of the black, which represent a 4:1 transformation (i.e. transformation in both the X and Y axis). The white and gray sections represent the location where a 2:1 transformation was used, white being in the X-direction and gray in the Y-direction.

![Transformation direction graph for Image 2 Max allowed MSE .125](image)

The same can be observed for icon 2 (figure 5.15) and icon 3 (figure 5.16) and their transformation direction tables (figure 5.17 and figure 5.18). Their MSE are .0189 and .0255 respectively.
Fig 5.15 Reconstructed Image 1 with Max allowed MSE of .125

Fig 5.16 Reconstructed Image 3 with Max allowed MSE of .125

Fig 5.17 Transformation direction graph for Image 1 Max allowed MSE .125
Figure 5.16, the reconstruction of Image 3 at a max MSE of .125, shows a larger amount of error when compared to the previous result of figure 5.12. This difference is due to the value of the threshold applied. The higher value of .125 allows the algorithm to utilize the 4:1 matrix element compression ratio.

5.3 Transferring C Code into the DSP board

Now that the system has been simulated in MATLAB and the transformation and inverse transformation algorithms have been coded, it is now time to bring it to a real-time implementation. The hardware chosen to implement this system was the ADC64 PCI Bus DSP Data Acquisition Card incorporating a Texas Instruments TMS320C32 floating point DSP processor integrated with 16 bit analog input/output capability, 16 bits of digital I/O, and up to 512K bytes of onboard sRAM. This card is manufactured by the Innovative Integration Corporation.
To aid in the programming of the DSP code into the DSP card a Development Package was obtained with the ADC64. This package consists of C compiler, terminal emulator, and a coefficient down loader. Also included are header files, which contain functions, designed by the board manufacturer to be utilized by the user to control the target DSP via a C program running outside the DSP board. The C compiler is an ANSI C compatible compiler, which produces optimized assembly code for the TMS320C3x family processors. Included with the compiler are sample code and card I/O functions that are used to manipulate the card. Such functions include `read_mailbox()`, `talker_download()`, and `start_app()`.

After the C code has been written and compiled into optimized assembly for the DSP processor, it must be download into the DSP board. The command to download the assembly, now called the “coeff” file, is sent by the host computer. This host computer runs a program designed to feed in sections of data to the DSP algorithms. Once the “coeff” file has been downloaded and started in the DSP, data is transferred into the DSP from the host computer. The transfer of data is done through “mailboxes” which are memory locations specified for the transfer of between the target system (DSP board) and the host. A 16-bit integer value is sent to the mailbox by the host and at that point the host waits until the target reads the mailbox. This continues until the entire section is completely loaded. The algorithm then begins to transforms the section. In a real life implementation the transformed data would then be passed to another module, but in order to prove that the system works the transformed data is returned and then sent back to the host using the same mailbox method. This process is repeated until the entire
image has been transformed and inverse transformed. Figure 5.19 show an Illustration of

![Mailboxes](image)

**Mailboxes**

Figure 5.19 Illustration of the operation

### 5.4 Problems encountered

It was found that the size of the details in the image affected the performance of the system. Text that is drawn with lines that are only 1 pixel in width caused the system to be too lossy. The MSE calculation reflects the overall difference between the original image and the reconstructed image. This value may be small, even if the reconstructed image is not readable enough. To verify this, an image with a large text size was ran though the system at the same max MSE of .125 (Fig 5.20).
This reconstruction yielded an MSE of .0219. Although this is not much lower than the previous value the lettering in this image is clearly readable.

Another problem encountered through the experimentation process was the threshold of the fix_output() function. If the value was chosen too high, important details were cut. If the threshold was too low, the extraneous information was not eliminated. Figure 5.22 and figure 5.23 are examples of the fix_output() threshold being chosen too low and too high, respectively.
The appropriate value for the threshold was found to be dependent on the image being compressed. So an intermediate value was chosen based on 3-dimensional plot of the reconstructed image before the `fix_output()` function is applied (Fig 5.24).
Although adjusting the `fix_output()` threshold improved the performance, the selection of wavelet bases used in the library proved to be the key issue. It was found that some images were better decomposed with certain wavelets. These wavelets were found through trial and error. Figures 5.25 and 5.26 are examples of such wavelets:
These wavelets are capable of describing the details of the images as well as the instances where no changes are present.
Chapter 6

Conclusion

This chapter summarizes the research efforts presented in the thesis and indicates scope for further studies chapter.

6.1 General

Today's world is overwhelmed with information. Information, however, may not be always useful. It must be routed to the correct person at the correct time. With this necessity comes the need for real-time transmissions. Since transmission airways are limited the information must be compressed. Wavelet transformation is a method to achieve this compression.

Though the system implemented here does not achieve the compression and accuracy of a JPEG system or some other compression schemes, it provided a first exploration into this approach to compression using wavelet transformation and also supplies a useful template for future work. Since the design of the implemented system was in a modular fashion, improvements and refinements can easily be done.

6.2 Thesis Contribution

This body of work has provided the necessary frame and development strategy for future "near real-time" applications. It has shown some of the potential of a compression
system using wavelet transformation and has shown what that would benefit the most from further development. The hope is that this work can be used as a starting point for future, more specific development. Such future work can simply "add" or "modify" the DWITS system and not redesign a completely new methodology.

6.3 Future Work

The present thesis is groundwork for future implementation of wavelet compression in real-time. These future works can include but are not limited to:

- Improvement in compression vs. mean squared error by implementing an adaptive "tiling" algorithm for the input images.

- Reduction of process time by implementing a spectrum analyzer that will eliminate certain wavelets in the wavelet library based on their frequency spectrum.

- Implementing a 2-D wavelet transform for images with gradual transitions from black to white. (i.e. Shades of gray)

- Expanding the system to images composed of float numbers and thus expanding for the possibility of gray transition [1].
With all the prospectives for future work, wavelet compression presents extensive possibilities for new compression standards. Companies like Motorola, Erickson, and Nokia are investing millions of dollars to develop compression standards that utilize wavelet transformations. Some early returns of this investment are already found in the JPEG2000 standard, which includes a wavelet transformation option. This current implementation offers a background for the development of future work on this topic.
LIST OF REFERENCES


5. Christopher Albin Edmonds, Adaptive Segmenting of Non-Stationary Signals (Florida International University, 1998) 24-31


