

ECE 533

Digital Image Processing- Fall 2003

Group Project

Embedded Image coding using zero-trees
of Wavelet Transform



Harish Rajagopal

Brett Buehl

12/11/03

Contributions

Tasks	Harish Rajagopal (%)	Brett Buehl (%)
Wavelet Transform codec	100	
EZW codec	50	50
Arithmetic codec		100
Total	50	50

Harish Rajagopal

Brett Buehl

1. Introduction

Embedded Zero-tree Wavelet Transform or EZW transform is specially designed for use with Wavelet transforms. The use of the word 'transform' in describing the codec is ambiguous since the codec is mainly considered with reordering and in some cases discarding lower order coefficients based on some criteria, as explained later, to achieve optimized compression results. The EZW encoder is based on the principle of progressive encoding or embedded encoding, which means that when more bits are added to the bit stream, the decoded image will contain more details. This property is similar to that exhibited by JPEG images. When the EZW scheme is used together with some optimizations we can compress the image very effectively with the property that the compressed data stream can have any bit rate desired. Any bit rate is only possible if there is information loss somewhere so that the compressor is lossy. However, lossless compression is also possible with an EZW encoder, but of course with less spectacular results. In this project we will be mainly concentrating on lossless compression in the sense that we will not be discarding any wavelet coefficients. But due to the inherent nature of wavelet encoding and subsequent round off errors there is always some loss associated with the encoding scheme. An explanation of the Wavelet Transform and the EZW encoding scheme is provided below.

1.1. Wavelet Transform

Wavelets are functions defined over a finite interval (ie., compact support) and having an average value of zero (ie., zero mean). Wavelets provide convenient sets of basis functions for function spaces, like the Fourier basis consisting of sine and cosine functions. The basic idea of the wavelet transform is to represent any arbitrary function $f(t)$ as a superposition of a set of such wavelets or basis functions. These basis functions or baby wavelets are obtained from a single prototype wavelet called the mother wavelet, by dilations or contractions (scaling) and translations (shifts). The Discrete Wavelet Transform of a finite length signal $x(n)$ having N components, for example, is expressed by an $N \times N$ matrix. The wavelets are better suited to represent functions that are localized both in time and frequency (it requires fewer terms than the Fourier analysis) since they themselves are localized in time and frequency. In particular, wavelets enable us to represent functions with sharp spikes or edges with fewer terms. This is important in many applications especially in image and data compression.

A wavelet transform transforms a signal from the time domain to the joint time-scale domain. This means that the wavelet coefficients are two-dimensional. If we want to compress the transformed signal we have to code not only the coefficient values, but also their position in time. When the signal is an image then the position in time is better expressed as the position in space. Figure 1 offers a graphical explanation of the 2-d wavelet transform

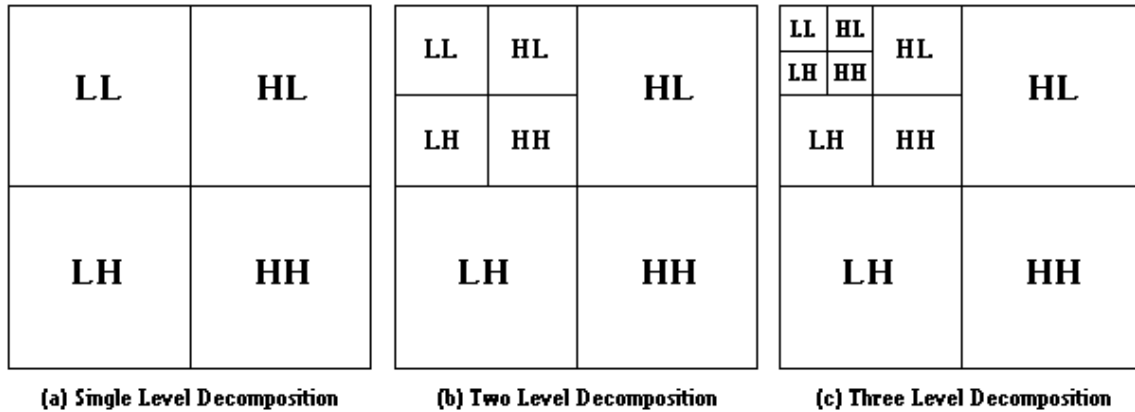


Figure 1: Explanation of 2-d wavelet decomposition

1.2. EZW Encoder

The EZW encoder is based on two important observations:

1. Natural images in general have a low pass spectrum. When an image is wavelet transformed the energy in the sub-bands decreases as the scale decreases (low scale means high resolution), so the wavelet coefficients will, on average, be smaller in the higher sub-bands than in the lower sub-bands. This shows that progressive encoding is a very natural choice for compressing wavelet transformed images, since the higher subbands only add detail;
2. Large wavelet coefficients are more important than small wavelet coefficients.

These two observations are exploited by encoding the wavelet coefficients in decreasing order, in several passes. For every pass a threshold is chosen against which all the wavelet coefficients are measured. If a wavelet coefficient is larger than the threshold it is encoded and removed from the image, if it is smaller it is left for the next pass. When all the wavelet coefficients have been visited the threshold is lowered and the image is scanned again to add more detail to the already encoded image. This process is repeated until all the wavelet coefficients have been encoded completely or maximum bit rate has been satisfied.

Since the wavelet transform is two dimensional we have to use the dependency between the wavelet coefficients across different scales to efficiently encode large parts of the image which are below the current threshold. This is where the concept of zero trees is introduced.

After wavelet transforming an image we can represent it using trees because of the subsampling that is performed in the transform. A coefficient in a low subband can be thought of as having four descendants in the next higher subband (see figure 2). The four descendants each also have four descendants in the next higher subband and we see a *quad-tree* emerge: every root has four leaves.

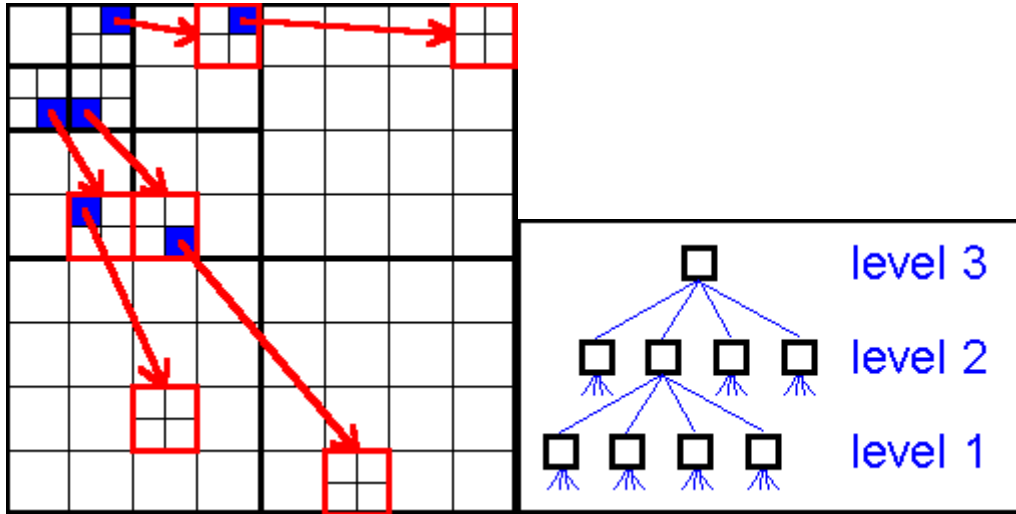


Figure 2:

The relations between wavelet coefficients in different subbands as quad-trees.

A zerotree is defined as a quad-tree of which all nodes are equal to or smaller than the root. The tree is coded with a single symbol and reconstructed by the decoder as a quad-tree filled with zeroes. Also the root has to be smaller than the threshold against which the wavelet coefficients are currently being measured.

The EZW encoder exploits the zerotree based on the observation that wavelet coefficients decrease with scale. It assumes that there will be a very high probability that all the coefficients in a quad tree will be smaller than a certain threshold if the root is smaller than this threshold. If this is the case then the whole tree can be coded with a single zerotree symbol. Now if the image is scanned in a predefined order, going from high scale to low, implicitly many positions are coded through the use of zero tree symbols. Even though the zero tree rule will be violated often, but as it turns out in practice, the probability is still very high in general. The price to pay is the addition of the zerotree symbols to the code alphabet.

2. Approach

The approach that we adopted for the implementation of the algorithm is described below.

2.1 System

The general block diagram for the scheme that we used is as shown in Figure 3. The system that we implemented includes the Discrete Wavelet Transform block which carries out the 2d DWT on the input image and outputs the 2d wavelet coefficients. This is then passed through an EZW encoder which carries out the embedded zero tree

encoding and give out the results of the dominant pass and the subordinate pass. This is still in the uncompressed form. So now we carry out arithmetic encoding on the EZW reordered data. The resulting bit stream is highly compressed and can be used for transmission to the decoding system. The decoding system consists of the Arithmetic decoder, which decodes the compressed bit stream obtained from the arithmetic encoding block in the transmission end into the dominant pass and the subordinate pass form that can now be passed to the EZW decoder block. The EZW decoder block requires in addition to this the size of the image and the initial threshold level. This is also passed to the EZW decoder using the header file. Now this block gives out the original coefficient levels that we had obtained from the 2d DWT. These coefficients are now passed to the IDWT block that performs the inverse discrete wavelet transform and gives back the reconstructed image.

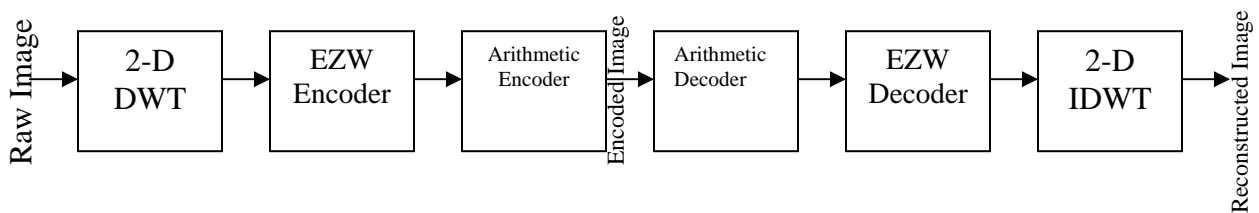


Figure 3: General Block Diagram

2.2 2-D DWT Transform

The Discrete Wavelet Transform (DWT) is a recurrent process of low- and high-pass filtering performed on the input data set. In the first iteration, the whole data vector (array) is low- and high-pass filtered, and the resulting vectors (arrays) stored separately. These two vectors (arrays) will each have the same length as the original vector (array). Since the expansion of the data is undesired, the two vectors (arrays) are down-sampled. This means that every second element in the vectors (arrays) is removed. The new high pass filtered vector (array) is now called a detail coefficient vector (array). The coefficients in this vector (array) are wavelet coefficients. These two vectors (arrays) form the first scale of the DWT. The second and subsequent iterations of the filtering process are performed on the vector (array) generated by the previous iteration's low-pass filter. The vector (array) from the high-pass filter is stored. Every iteration add another scale to the wavelet transformed data.

In order to transform images, a two-dimensional transform is needed. The Square Two-Dimensional DWT is calculated using a series of one dimensional DWTs:

- Step 1: Replace each image row with its 1D DWT
- Step 2: Replace each image column with its 1D DWT
- Step 3: Repeat steps (1) and (2) on the lowest subband to create the next scale
- Step 4: Repeat step (3) until the desired number of scales has been created

An example of a 2-dimensional DWT image is shown in figure 4. The figure 4(a) shows the actual image before 2D-DWT and figure 4(b) shows the wavelet coefficients

after wavelet transform. You can notice the similarity to figure 1 where we showed a graphical representation of the 2D-DWT for three levels.



Figure 4: Example of a Two dimensional DWT. (a) shows the original image and (b) shows the scaled DWT coefficients

2.3 EZW Codec

The EZW encoder consists mainly of two algorithms, viz., the dominant pass and the subordinate pass. The dominant pass and subordinate pass procedures are the algorithms for the EZW encoder that helps in selecting the significant values and discarding the sub threshold values. The EZW algorithm calculates the initial threshold, using the following formula:

$$T_{\text{initial}} = 2^{(\log_2(k))}$$

Where, T_{initial} is the initial threshold, and k is the maximum absolute coefficient value. That is, the initial threshold is the largest power of two that is less than the maximum absolute value of the coefficients. Using this initial threshold, a 'dominant-pass' procedure carried out on all the coefficients will identify all those that are significant; that is, all those that are larger in absolute value than the initial threshold. After the dominant-pass procedure is completed, the current threshold is halved, and a subordinate-pass procedure is performed. It progressively encodes detailed information about the significant coefficients. This process (starting with a dominant-pass at the current threshold) is reiterated until some desired state is achieved. The 'desired state' is either when the threshold reaches a minimum value (usually one but it is possible to continue with threshold values between zero and one) or when a predetermined function is satisfied. The most common function is one counting the number of bits written to the compressed data stream and ending when it reaches a specified value. Since the input bit count is known and the number of output bits controlled, this function can guarantee exact compression ratios. We adopt this procedure in this implementation. The flow chart of the EZW encoder is also shown in figure 5.

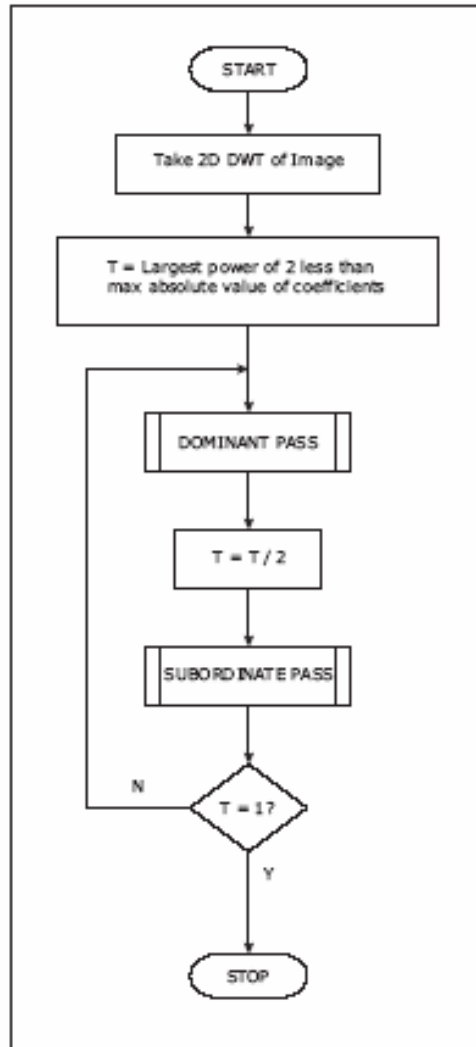


Figure 5: Flow chart for EZW encoding

2.3.1 Dominant Pass

The flow chart for the dominant pass procedure, shown in figure 6 demonstrates how the dominant pass selects which symbol to output. The first step is to check whether the current coefficient was previously found to be significant in which case the coefficient is skipped and nothing is outputted. If the coefficient has become significant at the current threshold, the absolute value of the coefficient minus the current threshold value is appended to the subordinate list and either the positive significant (P) or negative significant (N) symbol is outputted. The subordinate list records all the significant coefficients.

On the other hand, if the coefficient is not significant, the next step is to check if the insignificant coefficient is a child of an already discovered zerotree, in which case no symbol is outputted. If the coefficient is not part of an already discovered zerotree, it

must either be a root of a new zerotree or an isolated zero and the appropriate symbol (Z or T) is outputted.

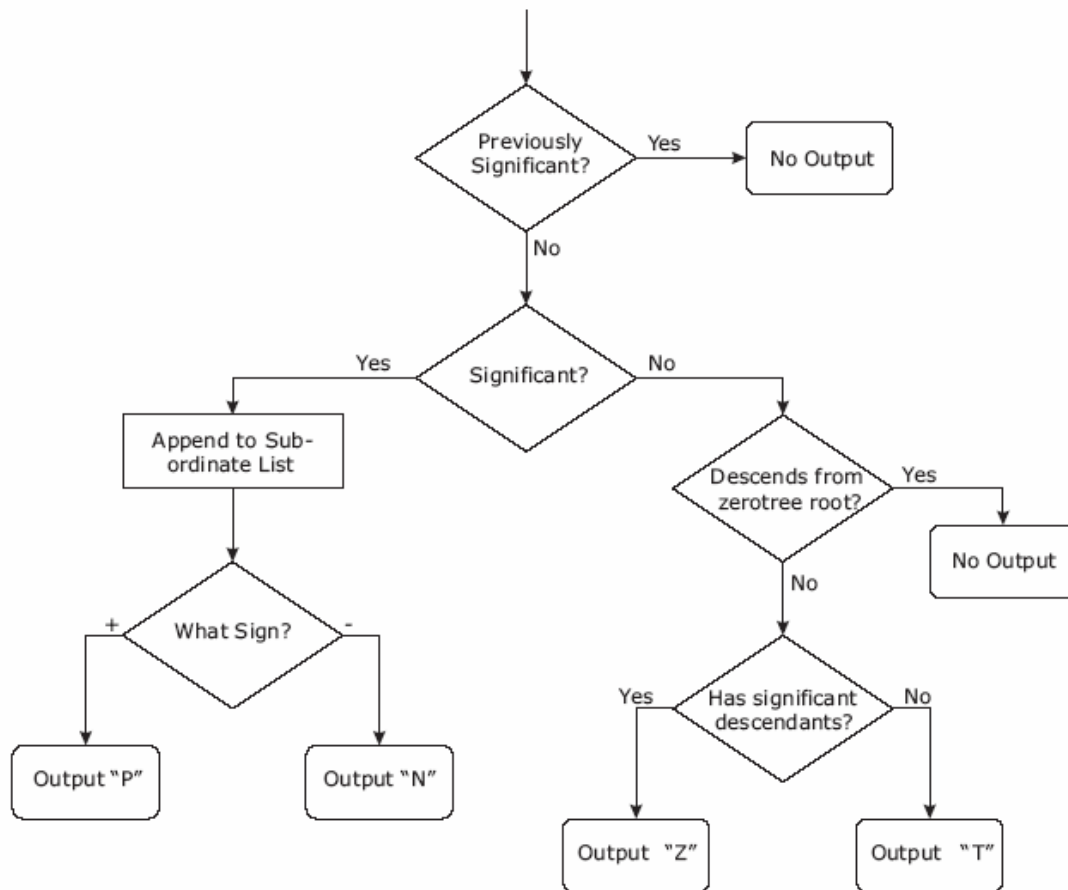


Figure 6: Flow chart for Dominant Pass

2.3.2 Subordinate Pass

The flow chart for the subordinate pass procedure is as shown in figure 7. The Subordinate Pass, sometimes called the Refinement Pass, ‘refines’ the value of each significant coefficient. For each coefficient in the subordinate list (coefficients are added by the dominate pass when they are found to be significant), the subordinate pass checks if their current value is larger or smaller than the current threshold value. If it is larger, a ‘1’ is sent to the entropy encoder and the current threshold is subtracted from the coefficient value in the subordinate list. If the coefficient is smaller than the threshold, a ‘0’ is sent to the entropy encoder.

The subordinate pass may be left out of the algorithm. This is because its only purpose is to refine the already encoded coefficients. Although this will reduce the code size, execution time and memory requirements, it will adversely affect the quality of the reconstructed image. In this implementation we use the full subordinate pass algorithm and depend on the arithmetic encoding for higher encoding.

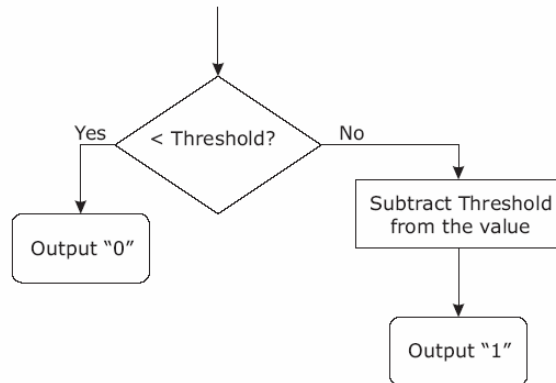


Figure 7: Flow chart for Subordinate Pass

The EZW encoder thus mainly rearranges the actual raw image in 8-bit format in such a way that it offers a higher compression ratio compared to the raw image for arithmetic encoding or entropy encoding. Here we use arithmetic encoding for our purposes.

2.4. Arithmetic Encoder

Arithmetic image coding is a coding scheme that utilizes the probabilities of the individual pixel values in a given image. These probabilities enable the algorithm to translate a bit stream into real numbers of determined precision. As more symbols are added to the encoding stream, the precision and the number of bits required to represent the real number increases. Since each symbol represents a unique interval less than 1, adding a symbol to stream decreases the code word range in a unique way. This allows for unambiguous decoding which results in a lossless coding scheme. The one problem that arises in machine computation is data precision. Larger numbers of distinct pixel values in a given image decreases the probabilities for individual pixels, thus decreasing the subinterval for each pixel. This causes smaller real numbers (between 0 and 1) which are difficult to represent and can encounter round-off error. The problem of precision was remedied, however, by scaling the probabilities into integer representation. Varying numbers of pixels, determined by the number of bits allowed for representation, are coded into individual code words to ensure that no information is lost. The diagram below (Figure 8) demonstrates how the symbol probabilities are used to encode a given stream.

The arithmetic encoder/decoder is included in this package to further increase the compression ratio of the EZW coded image. The encoder accepts the EZW encoded data output as its input in the form of a row vector. This vector consists of the symbols 'P', 'T', 'N', 'Z', '0', '1', and 'E'. The vector is encoded using the scheme described previously and is returned in the "uint8" Matlab precision format. Although all operations are carried out in double precision format, the 8-bit integer format requires less information for storage, maximizing the compression ratio.

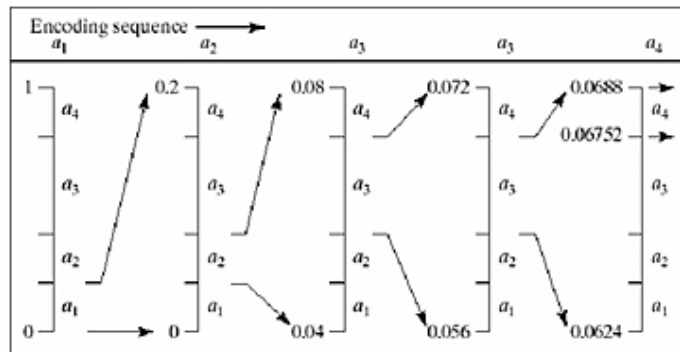


Figure 8: Shows process of encoding symbols using symbol probabilities.

3. Work Performed

In this project we coded all the blocks shown in the general block diagram of figure 3. The DWT and the IDWT block were coded so that it does forward and inverse wavelet transforms for a square image whose size is a power of 2. This was one of the simplifying assumptions that we made in order to speed up the transform speed. In addition we introduced a scale factor before the EZW encoder so that we could control the number of significant digits that we needed to consider before we used the EZW encoder. A higher scale factor results in the EZW encoder using more number of significant digits of the wavelet coefficient before the encoding this generally improves the clarity and characteristics of the image. The results that we obtained using different scale factors are also shown in the results section of the report.

To measure the efficiency of the image transform we saved the EZW encoded image in a file along with a header file containing the data needed for reconstruction of the image. The size of this file was compared against the size of the image obtained by JPEG compression. The comparisons are also shown in the results section. In addition to all these we used different basis function for the 2-dimensional discrete wavelet transform to study the differences between them. We mainly used ‘Haar’ and ‘Daubechies’ wavelets for this comparative study. Unless otherwise specified all the rest of the comparisons were carried out using the Daubechies wavelet of order 1, which was found to be less memory and execution time intensive while at the same time offering better performance.

In conclusion, we tested the efficiency of an image compression system using DWT and EZW encoder along with an arithmetic encoder. The testing was carried out for a number of images and the compression ratios of each were compared to that of JPEG transforms. The results obtained are presented in the next section.

4. Results

4.1 Compression Ratio Comparison:

Here we have compared the compressed sizes of the JPEG image and the EZW encoded image using our compression system. The results are presented below for three

images. We obtained comparable sizes for the two systems. For some images our system actually outperformed the JPEG system.

Figure 9 shows a comparison between the JPEG compressed image and EZW compressed image for three images. We can see that the EZW encoding actually outperformed the JPEG encoding in two images while the JPEG compression provides better compression for one image. This is the general trend that we observed with most images having comparable sizes.

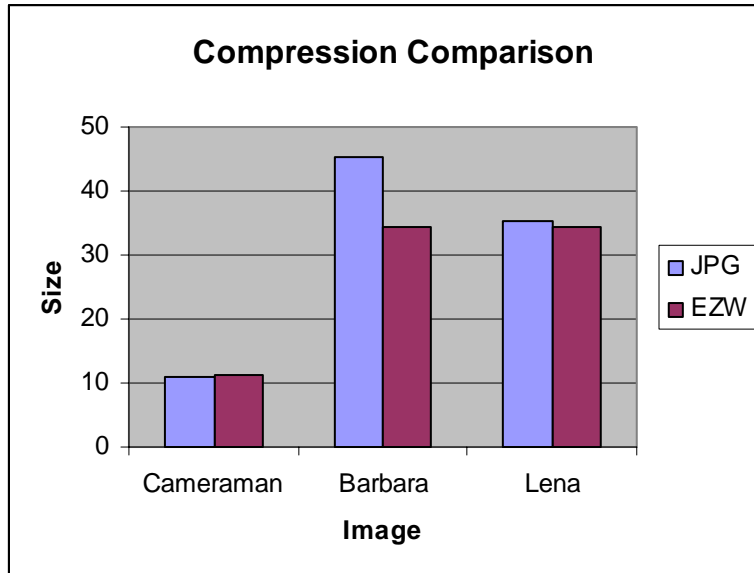


Figure 9: Comparison of the compressed images

The next figure (figure 10) shows the comparison of the sizes of EZW images with the BMP formatted raw images.

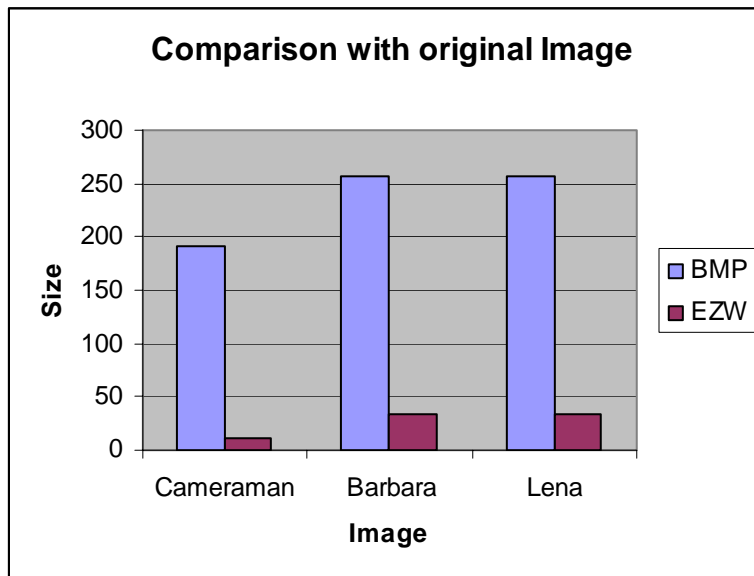


Figure 10: Comparison of compressed and uncompressed images

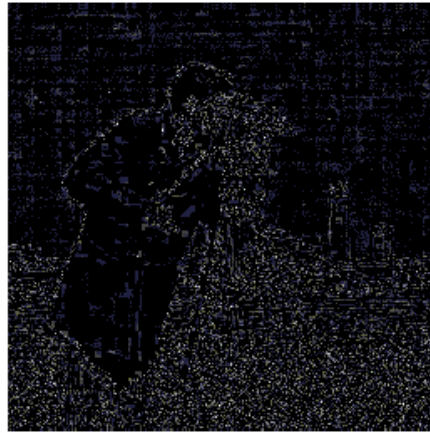
Figure 11 shows the initial image and the reconstructed image for a sample image and the scaled difference or error. The error is mainly due to the scaling factor and is a huge factor in the speed as well as the accuracy of the reconstruction.



(a)



(b)



(c)

Figure 11: (a) Initial Image (b) Reconstructed Image (c) Scaled Difference (Error)

4.2 Scaling Factor Comparison

The figure below (figure 12) shows the images obtained for different scaling factors. We can see that there is a noticeable difference in quality of the picture from scaling factor 1 to scaling factor 100. This shows the importance of the scaling factor to the picture quality. Also the size of the EZW encoded file depends on the scaling factor. Hence the scaling factor can be used as a size performance trade off parameter.



(a)



(b)



(c)

Figure 12: (a) Scaling Factor 1 (b) Scaling Factor 10 (c) Scaling Factor 100

4.3: Wavelet Comparison

For wavelet comparison we chose ‘Haar’ wavelets and ‘Daubechies’ wavelets of the first order. The images were comparable in quality but the size that we obtained for ‘Haar’ was almost twice what we obtained from the ‘Daubechies’ for all the images that we had tried with the same scaling factor.

5. Discussion

The performance that we obtained from the image compression system was well worth future development. The existence of a scaling factor which could control the image quality that we wanted was an added advantage for the EZW compression system. Also the independence of the system from the actual basis function ensures upward compatibility of the system to any future development and provides flexibility in the implementation of the system from the actual basis function used.

A detrimental factor for the system was the higher amount of time that was required for the codec to work. But this can be due to the inefficiency of the code in

Matlab. The algorithm is well suited for hardware implementation and acceleration and has a good scope in that direction.

In conclusion the system that we developed provides comparable results to JPEG encoding and also allows some amount of flexibility in the implementation as shown by the results.

6. References

J. Shapiro, Embedded image coding using zerotrees of wavelet coefficients, IEEE Trans. SP, vol. 41, pp. 3445-3462, 1993.

J. Shapiro, Smart compression using the embedded zerotree wavelet (EZW) algorithm Signals, Systems and Computers, 1993. The Twenty-Seventh Asilomar Conference on , 1-3 Nov. 1993 Page(s): 486 -490 vol.1

C. D Creusere, A new method of robust image compression based on the embedded zerotree wavelet Algorithm. IEEE Transactions on Image Processing, Vol. 6, No. 10 (1997), p. 1436-1442.

Subhasis Saha, Image Compression - from DCT to Wavelets : A Review. ACM Crossroads Student Magazine. (Website: <http://www.acm.org/crossroads/xrds6-3/sahaimgcoding.html>)