CHAPTER 4

ADAPTIVE WAVELET PACKET REARRANGEMENT ALGORITHM FOR ZERO TREE IMAGE CODING

4.1 INTRODUCTION

In Wavelet transforms only the scaling coefficients of an image are successively decomposed to generate wavelet coefficients. However, both scaling and wavelet coefficients can be decomposed, which leads to wavelet packet transforms. Wavelet packets pinpoint signal components present locally in the frequency domain. The non dyadic nature of a wavelet packet transform allows finding an orthogonal basis adapted to the content of the given image and the purpose of representation. In this Chapter a rearrangement algorithm for zero tree image coding is presented, which is more efficient and overcomes the consistency problem and computational complexities of other algorithms, The Set Partitioning In Hierarchical Trees and other coding methods have limitations of low performance in terms of PSNR and visual quality at low bit rates. The implemented coder overcomes the above limitations.

4.2 PRINCIPLES OF WAVELET PACKET DECOMPOSITION:

Wavelet packet decomposition (WPD) is a Wavelet transform where the signal is passed through more filters than the DWT as shown in Figure 4.1. In the DWT, each level is calculated by passing only the previous
approximation coefficients through a high and low pass filters. However in the WPD, both the detail and approximation coefficients are decomposed.

The signal $x[n]$ is decomposed into approximation and detailed coefficients this is first level of decomposition. Again both detailed and approximation coefficients are decomposed into separate approximation and detailed coefficients this is second level. Now again all approximation and detailed coefficients are decomposed to get all approximation and detailed coefficients this is third level of decomposition.

For $n$ levels of decomposition the WPD produces $2^n$ different sets of coefficients (or nodes) as opposed to $(3n + 1)$ sets for the DWT. However, due to the down sampling process the overall number of coefficients is still the same and there is no redundancy. From the point of view of compression, the standard wavelet transform may not produce the best result, since it is limited to wavelet bases that increase by a power of two towards the low
frequencies. It could be that another combination of bases produces a more desirable representation for a particular signal. The best basis algorithm by Coifman and Wickerhauser (1992) finds a set of bases that provide the most desirable representation of the data relative to a particular cost function (e.g. entropy). The block diagram is shown in Figure 4.2.

4.3 ENTROPY-BASED BEST BASIS SELECTION

Isotropic wavelet packet tree (WPT) decomposition with orthogonal sub bands is shown in Figure 4.3. Since each signal space is the direct sum of its subspaces, the original signal can be represented by the leaves of any sub tree of the WPT. Fast and efficient implementation of the wavelet packet transform can be done in $O(N \log N)$ time, which is equivalent to that required by the fast Fourier transform. The use of a predetermined sub tree, such as the DWT, may reduce computation time. The search for the “best” non redundant representation of the data by the leaves of a subtree of WPT is called best-basis selection. The algorithm is initialized by evaluating each sub band with desired metric, such as rate or distortion. Then a post-ordered search of the WPT is done in which a best-basis decision is made by comparing the quantitative value of each node to the cumulative effects of the node’s descendant branch (i.e., subspaces exclusive of the space of the given node). The post-ordered search exploits the property that each signal space is the direct sum of its subspaces.

A minimum-entropy basis from the most refined partition upwards. It is started by calculating the entropy of an expansion relative to intervals of length one, then we compare the entropy of each adjacent pair of intervals to the entropy of an expansion on their union. We pick the expansion of lesser entropy and continue up to some maximum interval size. This uncovers the
minimum entropy expansion for that range of interval sizes. This rough idea can be made precise as well as generalized to all libraries with a tree structure.

**Definition:** A library of orthogonal bases is a (binary)tree if it satisfies the following:

1) Subsets of basis vectors can be identified with this interval of N
   \[ I_{nk} = [2^k n, 2^k(n + 1)] \quad k \in \mathbb{Z}, n \in \mathbb{Z} \]
   Where k is the level of binary tree, Z max number of levels and n is the total interval.

2) Each basis in the library corresponds to N based one, composed by disjoint \( I_{nk} \) coverage.

3) If \( V_{nk} \) equals \( I_{nk} \), then
   \[ V_{n,k+1} = V_{2n,k} \otimes V_{2n+1,k} \]

The library of wavelet packet bases is naturally organized as subsets of a binary tree. Each node represents a subspace of the original signal. Each subspace is the orthogonal direct sum of its two children nodes. The leaves of every connected subtree give an orthonormal basis. The library of local trigonometric bases over a compact interval U may be organized as a binary tree by taking partitions localized to a dyadic decomposition of U. Then I, will correspond to the sine basis on U, and \( I_{nk} \) will correspond to the local sine basis over interval n of the \( 2^k \) intervals at level k of the tree.
Figure 4.2 Block diagram representation

Three Level, 1D Tree of Coefficients

Pruned, three level, optimum basis.

Wavelet basis

Figure 4.3 Three level 1 D Tree of coefficients
4.4 REARRANGEMENT ALGORITHM

As Zero tree coding has been developed so that it may be adequate for dyadic multiresolution wavelet decomposition, it is very difficult to perform zero trees coding in the WP domain, which does not have a dyadic multiresolution structure. But the rearrangement algorithm presented in this section makes it possible to perform zero tree coding even in the WP domain. Let us consider a wavelet packet (WP) decomposed image, which is obtained by means of performing another one-level WT on a certain sub band of a typical dyadic multiresolution WT image. If we rearrange the coefficients of the 4 sub bands in the manner represented and regard the rearranged domain as one sub band (although it consists of 4 sub bands), we can virtually obtain the dyadic multiresolution structure. Because the rearranged coefficients corresponding to the same spatial location are located so that they belong to the same coefficient tree, it becomes possible for zero trees coding to be performed in the rearranged WP domain like in the WT domain.

In this Chapter a new rearrangement method for WP is proposed. The basic unit of this rearrangement method is 4 coefficients. The coefficients which belong to the same band also belong to the same tree. Surely, this rearrangement method partially deteriorates the spatial meaning of coefficients in the wavelet (packet) decomposition domain. But, as mentioned before, the fact that the optimization algorithm of constructing WP basis declares that it is more profitable for a certain band to be split, means the energy is concentrated on one band among deriving four bands. Therefore, for efficient zero tree coding, it is more desirable to rearrange coefficients so that coefficients corresponding to the same frequency band may gather as children of a parent coefficient by emphasizing their frequency meaning than to their spatial meaning. By performing the rearrangement process recursively, we can get a dyadic multiresolution structured WP. As shown in Figure 4.4, the
rearranged WPT domain can be regarded as if it is a dyadic wavelet decomposed domain. Therefore, zero tree coding can be performed in the WPT domain through the rearrangement process explained.

4.5 ZERO TREE CODING

The Zero tree wavelet encoder is based on progressive encoding to compress an image into a bit stream with increasing accuracy. This means that when more bits are added to the stream, the decoded image will contain more detail, a property similar to JPEG encoded images. It is also similar to the representation of a number like $\pi$. Every digit we add increases the accuracy of the number, but we can stop at any accuracy we like. Progressive encoding is also known as embedded encoding.

Coding an image using the Zero tree wavelet scheme together with some optimizations, results in a remarkably effective image compressor with the property that the compressed data stream can have any desired bit rate. 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 a Zero tree wavelet encoder, but with less spectacular results.
The Zero tree wavelet 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 sub bands 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 another criterion has been satisfied (maximum bit rate for instance). The dependency between the wavelet coefficients across different scales to efficiently encode large parts of the image which are below the current threshold.

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. After wavelet transforming an image we can represent it using trees because of the sub sampling that is performed in the transform. A coefficient in a low sub band can be thought of as having four descendants in the next higher sub band. The four descendants each also have four descendants in the next higher sub band and we see a quad-tree emerge: every root has four leafs.

A zero tree is 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. To clutter this definition we have to add that the root has to be smaller than the threshold against which the wavelet coefficients are currently being measured.

The Zero tree wavelet encoder exploits the zero tree 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 zero tree 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. Of course the zero tree rules 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 zero tree symbols to our code alphabet.

A very direct approach is to simply transmit the values of the coefficients in decreasing order, but this is not very efficient. This way a lot of bits are spend on the coefficient values and we do not use the fact that we know that the coefficients are in decreasing order. A better approach is to use
a threshold and only signal to the decoder if the values are larger or smaller than the threshold. If we also transmit the threshold to the decoder, it can reconstruct already quite a lot. To arrive at a perfect reconstruction we repeat the process after lowering the threshold, until the threshold has become smaller than the smallest coefficient we wanted to transmit. We can make this process much more efficient by subtracting the threshold from the values that were larger than the threshold. This results in a bit stream with increasing accuracy and which can be perfectly reconstructed by the decoder. If we use a predetermined sequence of thresholds then we do not have to transmit them to the decoder and thus save us some bandwidth. If the predetermined sequence is a sequence of powers of two it is called bitplane coding since the thresholds in this case correspond to the bits in the binary representation of the coefficients.

One important thing is however still missing; the transmission of the coefficient positions. Indeed, without this information the decoder will not be able to reconstruct the encoded signal (although it can perfectly reconstruct the transmitted bit stream). It is in the encoding of the positions where the efficient encoders are separated from the inefficient ones.

Zero tree wavelet encoding uses a predefined scan order to encode the position of the wavelet coefficients (Figure 4.5). Through the use of zero trees many positions are encoded implicitly. Several scan orders are possible (Figure 4.6 and Figure 4.7) as long as the lower subbands are completely scanned before going on to the higher subbands. In a raster scan order is used, while in some other scan orders are mentioned. The scan order seems to be of some influence of the final compression result.
The SPIHT algorithm proposed by Said and Pearlman (1996) is unique in that it does not directly transmit the contents of the sets, the pixel
values, or the pixel coordinates. What it transmits is the decisions made in each step of the progression of the trees that define the structure of the image. Because only decisions are being transmitted, the pixel value is defined by what points the decisions are made and their outcomes, while the coordinates of the pixels are defined by which tree and what part of that tree the decision is being made on. The advantage to this is that the decoder can have an identical algorithm to be able to identify with each of the decisions and create identical sets along with the encoder. The wavelet decomposition tree structure is shown in Figure 4.8.

![Figure 4.8 wavelet decomposed tree structure](image)

**Figure 4.8 wavelet decomposed tree structure**

A general procedure for the code is as follows:

The following are the lists that will be used to keep track of important pixels.

- **LIS** : List of Insignificant sets
- **LIP** : List of insignificant pixels
- **LSP** : List of significant pixels

1. Initialization: output n, n can be chosen by user or pre defined for maximum efficiency,
   LSP is empty; add starting root coordinates to LIP and LIS.
2. Sorting pass: (new n value)
   a. for entries in LIP: (stop if the rest are all going to be insignificant)
      - decide if it is significant and output the decision result
      - If it is significant, move the coordinate to LSP and output sign of the coordinate
   b. for entries in LIS: (stop if the rest are all going to be insignificant)

   IF THE ENTRY IN LIS REPRESENTS \( \mathcal{D}(i,j) \)
   (every thing below node on tree)
      - decide if there will be any more significant pixels further down the tree and output the decision result- if it is significant, decide if all of its four children (\( \mathcal{O}(i,j) \)) are significant and output decision results
         * if significant, add it to LSP, and output sign
         * if insignificant, add it to LIP

   IF THE ENTRY IN LIS REPRESENTS \( \mathcal{L}(i,j) \)
   (not children but all others)
      - decide if there will be any more significant pixels in \( \mathcal{L}(i,j) \) further down the tree and output the decision result
      - if there will be one, add each child to LIS of type \( \mathcal{D}(i,j) \) and remove it from LIS

3. Refinement Pass: (all values in LSP are now \( 2^n \leq |c_{ij}| \))
   a. For all pixels in LSP, output the nth most significant bit

4. Quantization-step Update: decrement n by 1 and do another pass at step 2.
   This procedure is highly efficient for compression at high bit rate.
The limitations we have in these coders are that they have low performance in terms of PSNR (Peak signal to noise ratio).

**4.7 MODIFIED SPIHT (MSPIHT)**

SPIHT algorithm is well known for its simplicity and efficiency. SPIHT's high memory requirement is a major drawback for hardware implementation. In this study, a modification of SPIHT named Modified-SPIHT (MSPIHT) is presented. The MSPIHT coding algorithm is modified using one list to store the co-ordinates of wavelet coefficients instead of three lists of SPIHT, defines two terms number of error bits and absolute zero tree, and merges the sorting pass and the refinement pass together as one scan pass. Comparison of MSPIHT with SPIHT on different test images shows that for coding a 512x512, grey-level image, MSPIHT reduce execution time for coding at most 7 times and for decoding at most 11 times at low bit rate, saves at least 0.5625 MBytes of memory.

**4.7.1 Proposed Algorithm**

**Step 1:** The input image is subjected to wavelet packet decomposition.(full wavelet packet tree)

**Step 2:** Optimal tree is generated by first calculating entropy of each node and getting a matrix of entropy values and later apply Best basis selection rules to get the optimal tree.

**Step 3:** Zero tree Encoding is Implement by performing dominant pass and subordinate pass. The output is the compressed data streams.

**Step 4:** The reconstruction procedure is applied by doing zero tree decoding and inverse wavelet packet decomposition.
**Step 5:** The reconstructed image is compared with the original image to calculate CR and PSNR to visualize the efficiency of the implemented method.

The above Algorithm is done for SPIHT and MSPIHT encoding.

List of significant values: 63 -34 49…..The first Dominant pass on 8 bit matrix is shown in Figure 4.9.

**Figure 4.9 First dominant pass on 8 bit matrix.**

### 4.8 SIMULATION RESULTS AND DISCUSSION

Tables 4.1 and 4.2 give the PSNR and CR values for the proposed method and is compared with the existing methods. Also Figure 4.10 shows the reconstructed images for images such as Lena, rice, cameraman and peppers. The performance of the proposed Algorithm is analyzed in terms of subjective quality, PSNR values of the reconstructed images for different bit rate. A number of such standard test images are available in the website ftp:/ipl.rpi.edu/pub/image/still/usc. These test images
are chosen in such a way that these contain low, high as well as medium frequency components. The Graphs for the proposed method for Rice Image and Camera man image is shown in Figure 4.11 and 4.12.

Table 4.1 Performance comparison of proposed method in terms of PSNR in db for Rice Image

<table>
<thead>
<tr>
<th>BITRATE</th>
<th>ZERO TREE</th>
<th>SPIHT</th>
<th>MSPIHT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>27.93</td>
<td>31.05</td>
<td>32.04</td>
</tr>
<tr>
<td>0.2</td>
<td>30.33</td>
<td>32.44</td>
<td>34.64</td>
</tr>
<tr>
<td>0.3</td>
<td>32.21</td>
<td>33.63</td>
<td>35.42</td>
</tr>
<tr>
<td>0.4</td>
<td>33.94</td>
<td>34.71</td>
<td>36.63</td>
</tr>
<tr>
<td>0.5</td>
<td>34.63</td>
<td>35.06</td>
<td>37.82</td>
</tr>
<tr>
<td>0.6</td>
<td>35.02</td>
<td>35.86</td>
<td>38.03</td>
</tr>
<tr>
<td>0.7</td>
<td>36.01</td>
<td>36.62</td>
<td>38.83</td>
</tr>
<tr>
<td>0.8</td>
<td>36.92</td>
<td>37.41</td>
<td>39.92</td>
</tr>
</tbody>
</table>

Table 4.2 Performance comparison of proposed method in terms of PSNR in db for Camera man Image

<table>
<thead>
<tr>
<th>BITRATE</th>
<th>ZERO TREE</th>
<th>SPIHT</th>
<th>MSPIHT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>25.32</td>
<td>30.34</td>
<td>31.24</td>
</tr>
<tr>
<td>0.2</td>
<td>27.43</td>
<td>31.54</td>
<td>32.41</td>
</tr>
<tr>
<td>0.3</td>
<td>30.22</td>
<td>32.33</td>
<td>33.43</td>
</tr>
<tr>
<td>0.4</td>
<td>32.47</td>
<td>33.35</td>
<td>34.76</td>
</tr>
<tr>
<td>0.5</td>
<td>33.64</td>
<td>34.36</td>
<td>36.82</td>
</tr>
<tr>
<td>0.6</td>
<td>34.20</td>
<td>34.94</td>
<td>37.13</td>
</tr>
<tr>
<td>0.7</td>
<td>35.11</td>
<td>35.56</td>
<td>38.23</td>
</tr>
<tr>
<td>0.8</td>
<td>35.72</td>
<td>36.23</td>
<td>38.97</td>
</tr>
</tbody>
</table>
Figure 4.10 Reconstructed images for MSPIHT, SPIHT, Zero tree encoding

(a) Reconstructed images for MSPIHT
(b) Reconstructed images for SPIHT
(c) Reconstructed images for Zero tree
A general tree structure for wavelet packet transform with zero tree, SPIHT and MSPIHT is presented. Experimental results show that performance of MSPIHT is better in terms of PSNR and visual quality than its
counter part SPIHT and zero tree. A possible explanation for this is the relative simplicity of this coder and its inability to exploit intra–subband redundancies. However, this coder maintains better visual quality for complex textured images such as camera man, rice, lena etc. over the wavelet based methods. Therefore it is seen that simulation results for MSPIHT are good when compared to SPIHT and Zerotree methods. Therefore the PSNR values are high for MSPIHT than SPIHT and Zerotree Encoding.