Analysis of discrete wavelet based image compression technique: A review

N B Chopade¹* and A A Ghatol²
¹Department of Electronics and Telecommunication Engineering, SSGM College of Engineering, Shegaon 444 203, India
²Dr Babasaheb Ambedkar Technological University, Lonere, 402 103, India
Received 08 October 2007; revised 03 August 2009; accepted 07 August 2009

In this paper, wavelet based coding algorithm, Set Partitioning in Hierarchical Trees (SPIHT), is considered for encoding and decoding image data. SPIHT uses recursive set partitioning procedure to sort subsets of wavelet coefficient by maximum magnitude with respect to threshold. SPIHT is simple, less complex and has very fast encoding and decoding, making it very efficient in multimedia communication. Numerical results obtained using MATLAB shows that output image has high value of peak signal to noise ratio with good compression ratio for low bit rate.

Keywords: Bit rate, Discrete wavelet transform (DWT), Peak signal to noise ratio (PSNR), SPIHT

Introduction
Processor speed, digital communication system performance, demand for data storage capacity and data-transmission bandwidth continues to outstrip capabilities of available technologies¹,². Multimedia-based web applications have not only sustained need for more efficient ways to encode signals and images but have made compression of such signals central to storage and communication technology for over 80 billion new digital images produced yearly. Compression and coding reduces storage cost, channel bandwidth and transmission rate³⁻⁷. Fundamental components of compression are reduction of redundancy and irrelevancy. In still image, compression is achieved by removing spatial redundancy and spectral redundancy⁸.

This study presents a wavelet-based coding algorithm, Set Partitioning in Hierarchical Trees (SPIHT), to encode and compress image data.

Wavelet Functions

Continuous Wavelet Transform (CWT)
All wavelet functions used in transformation are derived from mother wavelet through translation or shifting and scaling or compression.

\[ CWT \left( a, b \right) = \int_{-\infty}^{\infty} \psi_{a,b}^{*}(t) f(t) dt \]  \hspace{1cm} \ldots(1)

where \( f(t) \) is signal to be analyzed, \( \psi_{a,b}(t) \) is mother wavelet, \( a \) is scaling factor and \( b \) is shifting factor. Mother wavelet has to satisfy following admissibility condition:

\[ C_{\psi} = \int_{-\infty}^{\infty} \left| \frac{\psi(\omega)}{\omega} \right|^2 d\omega < \infty \]  \hspace{1cm} \ldots(2)

Since, CWT behaves like orthonormal basis decomposition, it can be shown that it is isometric⁸. Function \( f(t) \) can be recovered from its transform as

\[ f(t) = \frac{1}{C_{\psi}} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} CWT(a,b)\psi_{a,b}(t) \frac{dadb}{a^2} \]  \hspace{1cm} \ldots(3)

Discrete Wavelet Transform (DWT)
DWT is based on sub-band coding and yields a fast computation of wavelet transform⁹⁻¹¹. In DWT, \( f(t) \) is passed through filters with different cutoff frequencies at different scales. CWT has drawbacks of redundancy and impracticability with digital computers. Therefore, parameters \( a \) and \( b \) are evaluated on a discrete grid of time-scale plane leading to a discrete set of continuous basis functions. Discretization is performed by setting \( a = a_0^j \) and \( b = k \cdot a_0^j \cdot b_0 \) for \( j, k \in \mathbb{Z} \). where, \( a_0 > 1 \) is a dilated step and \( b \neq 0 \) is a translation step. Family of wavelets then becomes

\[ \psi_{j,k}(t) = a_0^{-j} \psi \left( a_0^{-j} t - k b_0 \right) \]  \hspace{1cm} \ldots(4)
and wavelet decomposition of a function $f(t)$ is

$$f(t) = \sum_{j} \sum_{k} D_f(j,k) \psi_{j,k}(t)$$  \hfill \text{(5)}

where 2-dimensional set of coefficients $D_f(j,k)$ is called DWT of given function $f(t)$.

**Wavelets & Filter Bank**

A 2D DWT can be implemented as a filter bank by combining analysis and synthesis stages together\textsuperscript{11}. A 2D DWT of an image is obtained by using low pass $h_0(m)$ and high pass $h_1(m)$ filters successively (Fig. 1). An image component obtained by low pass filtering of rows and columns is LL image. Low pass filtering of rows and high pass filtering of columns, gives LH image component. High pass filtering of rows and low pass filtering of columns gives HL image component. HH component is result of high pass filtering of rows and columns.

**Peak Signal to Noise Ratio (PSNR)**

PSNR, a function of mean squared error (MSE), is industry accepted metric for objective quantification of image compression algorithm. If $x_i$, $\tilde{x}_i$ are input and reconstructed pixel values in image respectively, $M$ is maximum peak-to-peak value in image (typically 256 for 8 bit image). A good PSNR performance is a prerequisite for any modern compression algorithm\textsuperscript{5} as

$$MSE = \sum_{i=0}^{N-1} \left( x_i - \tilde{x}_i \right)^2$$  \hfill \text{(6)}

$$PSNR = 10 \log_{10} \left( \frac{M^2}{MSE} \right)$$  \hfill \text{(7)}

**Set Partitioning in Hierarchical Trees (SPIHT)**

**Spatial Orientation Trees**

Most of image energy is concentrated in low frequency components. Consequently, variance decreases from highest to lowest levels of subband pyramid. There is a spatial self-similarity between subbands, and coefficients are expected to be better magnitude-ordered while moving downward in pyramid following same spatial orientation. For instance, large low-activity areas are expected to be identified in highest levels of pyramid, and they are replicated in lower levels at same spatial locations. A tree structure defines spatial relationship on hierarchical pyramid (Fig. 2). Each node of tree corresponds to a pixel and is identified by pixel coordinate. Its direct descendants (offspring) correspond to pixels of same spatial orientation in next finer level of pyramid. Tree is defined in such a way that each node has either no offspring (leaves) or four offspring, which always form a group of 2 x 2 adjacent pixels. Arrows are oriented from parent node to its four offspring (Fig. 3). Pixels in highest level of pyramid are tree tools and are also grouped in 2 x 2 adjacent pixels. Their offspring branching rule is different, and in each group, one of them
(indicated by LL3 in Fig. 2) has no descendants\(^9\).

In a practical implementation, significance information is stored in three ordered lists [list of insignificant sets (LIS), list of insignificant pixels (LIP), and list of significant pixels (LSP)]. Sorting algorithm uses following sets of coordinates: i) \(O(i, j)\) — Set of coordinates of four offspring of node \((i, j)\), if node \((i, j)\) is a leaf of a spatial orientation tree, then \(O(i, j)\) is empty; ii) \(D(i, j)\) — Set of coordinates of the descendants of node \((i, j)\); iii) \(H(i, j)\) — Set of coordinates of roots of all spatial orientation trees; and iv) \(\{i, j\}\) — Difference set \(D(i, j) - O(i, j)\), which contains all descendants of tree node \((i, j)\) expect its four offspring\(^1\).

**Coding Sequence**

Encoder and decoder of SPIHT have same sorting algorithm, so that decoder can duplicate encoders execution path and receives result of magnitude comparisons. Pixels are grouped together in sets, which comprise of regions in transformed image. A set of \(\tau\) of pixels is significant with respect to \(n\) if Eq. (8) is true otherwise it is insignificant.

\[
\max_{(i, j) \in \tau} \left\{ |c_{i,j}| \right\} \geq 2^n \tag{8}
\]

where \(c_{i,j}\) are transformed coefficients obtained after applying a wavelet transform, \(2^n\) is a threshold value.

Sorting algorithm divides set of pixels into partitioning subsets \(T_m\) performs magnitude test. If decoder receives a ‘no’ to answer, subset is insignificant and if ‘yes’ subset is significant. A certain rule shared by encoder and decoder is used to partition \(T_m\) into new subset \((T_m, t)\) and significance test is then applied to new subset. This set division continues until magnitude test is done to all single coordinate significant subset in order to identify each significant coefficient\(^4\)\(^5\).
Encoding Algorithm

1) Initialization

Set \( n = \log_2 (\max_{i,j} |c_{ij}|) \) and transmit \( n \). Set LSP as an empty. Set LIP to coordinates of all roots \((i, j) \in H\). Set LIS to coordinates of all roots \((i, j) \in H\) that have descendants.

2) Sorting Pass

2.1) For each entry \((i, j)\) in LIP do:

2.1.1 output \( S_n(i, j) \);

2.1.2 if \( S_n(i, j) = 1 \), move \((i, j)\) to the LSP and output sign of \( c_{ij} \).

2.2) For each entry \((i, j)\) in LIS do:

2.2.1 if entry is of type A then

• output \( S_n(D(i, j)) \);

• if \( S_n(D(i, j)) = 1 \), move \((i, j)\) to end of the LIS as an entry of type B entry and go to Step 2.2); else , remove entry \((i, j)\) from LIS;

2.2.2) if entry is of type B then

• output \( S_n(\xi(i, j)) \);

• if \( S_n(\xi(i, j)) = 1 \), append each \((k, l) \in O(i, j)\) to LIS as a type A entry;

• if \( S_n(\xi(i, j)) = 0 \), append \((k, l) \in O(i, j)\) to end of LIP as an entry of type B entry; else , remove \((i, j)\) from LIS.

Table 1 — Result of SPIHT algorithm on different images

<table>
<thead>
<tr>
<th>Image</th>
<th>MRI</th>
<th>Lena</th>
<th>Cameramen</th>
<th>Rice</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit rate</td>
<td>PSNR, dB</td>
<td>CR*</td>
<td>PSNR, dB</td>
<td>CR</td>
</tr>
<tr>
<td>0.2</td>
<td>33.81</td>
<td>83.28</td>
<td>32.52</td>
<td>80.22</td>
</tr>
<tr>
<td>0.4</td>
<td>37.23</td>
<td>81.16</td>
<td>35.00</td>
<td>79.56</td>
</tr>
<tr>
<td>0.6</td>
<td>40.00</td>
<td>79.34</td>
<td>36.74</td>
<td>78.34</td>
</tr>
<tr>
<td>0.8</td>
<td>42.62</td>
<td>78.88</td>
<td>37.32</td>
<td>74.80</td>
</tr>
<tr>
<td>1.0</td>
<td>44.94</td>
<td>64.65</td>
<td>39.22</td>
<td>68.94</td>
</tr>
</tbody>
</table>

*CR, compression ratio

Table 2 — PSNR comparison for different algorithms

<table>
<thead>
<tr>
<th>Bit rate</th>
<th>EZW PSNR, dB</th>
<th>SPIHT PSNR, dB</th>
<th>SPECK PSNR, dB</th>
<th>EBCOT PSNR, dB</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.25</td>
<td>26.84</td>
<td>28.64</td>
<td>28.85</td>
<td>25.93</td>
</tr>
<tr>
<td>0.50</td>
<td>27.98</td>
<td>30.23</td>
<td>30.75</td>
<td>26.74</td>
</tr>
<tr>
<td>0.75</td>
<td>29.26</td>
<td>31.68</td>
<td>32.81</td>
<td>28.17</td>
</tr>
<tr>
<td>1.00</td>
<td>31.53</td>
<td>32.46</td>
<td>32.81</td>
<td>30.18</td>
</tr>
</tbody>
</table>

* if \( \xi(i, j) \neq 0 \), move \((i, j)\) to end of the LIS as an entry of type B entry and go to Step 2.2); else, remove entry \((i, j)\) from LIS.
3) **Refinement Pass**

For each entry (i, j) in LSP, except those included in last sorting pass (one with same n), output \(n^{th}\) most significant bit of \(|c_{i,j}|\).

4) **Loop**

Decrement n by 1 and go to Step 2 if needed.

**Numerical Results**

SPIHT coding/decoding algorithm has been implemented in MATLAB and tested on images of different size and type. Biorthogonal 9/7 wavelet filters are used. DWT level is kept equal to three. Numerical results are presented for images of different types and size (Table 1) in terms of PSNR values and compression ratio (CR) for various bit rates. Fig. 4 gives rate distortion curves for different image data. Comparative performance of different coding algorithm is given in Fig. 5. PSNR performance of this algorithm is compared with state of art coding and compression algorithm (EZW, SPIHT, EBCOT) and numerical results are obtained (Table 2). Original images and decoded images at different data rates are quite distinct (Fig. 6).

**Conclusions**

Wavelet based SPIHT image coding algorithm has been presented. Increasing bit rate, PSNR increases and CR decreases. SPIHT coding algorithm with its embedded code and fast execution are so impressive that it is used for standardization in many image compression systems.

**References**