CHAPTER 2

LITERATURE REVIEW

Image Compression is achieved by removing the redundancy in the image. Redundancies in the image can be classified into three categories; inter-pixel or spatial redundancy, psycho-visual redundancy and coding redundancy [57-59].

Inter-pixel Redundancy: Natural images have high degree of correlation among its pixels. This correlation is referred as inter-pixel redundancy or spatial redundancy and is removed by either predictive coding or transform coding.

Psycho-visual redundancy: Images are normally meant for consumption of human eyes, which does not respond with equal sensitivity to all visual information. The relative relevancy of various image information components can be exploited to eliminate or reduce any amount of data that is psycho-visually redundant. The process, which removes or reduces Psycho-visual redundancy, is referred as quantization.

Coding redundancy: variable-length codes matching to the statistical model of the image or its processed version exploits the coding redundancy in the image.
Lossy compression: An Image may be lossy compressed by removing information, which are not redundant but irrelevant (psycho visual redundancy). Lossy-compression introduces certain amount of distortion during compression, resulting in more compression efficiency.

2.1 Transform based Image compression

Image coding techniques based on transform use a mathematical transform to map the image pixel values onto a set of de-correlated coefficients, thereby removing inter-pixel redundancy. These coefficients are then quantized (psycho-visual redundancy), and encoded (coding efficiency). The key factor for the success of transform-based coding schemes is their excellent energy compaction property i.e. large fraction of total energy of image is packed in few coefficients. Most of the transform coefficients for natural images have small magnitudes and can be quantized and encoded or discarded without causing significant loss of information.

There are many types of image transform like discrete Fourier transform (DFT), discrete sine transform (DST), discrete cosine transform (DCT), Karhunen-Loeve transform (KLT), Slant transform, Hadamard transform and discrete wavelet transform (DWT). For compression purposes, the higher the capability of energy compaction, the better the transform. Though KLT transform is best in terms of energy compaction (transform coding gain), one drawback of KLT transform is that it is data dependent and overhead of sending the transform may reduce the transform coding gain. Another popular transform is discrete cosine transform (DCT), which offers transform coding gain closer to KLT and higher than DFT. In addition, the computational complexity of DCT is less than DFT. Due to
these reasons, DCT has become the most widely used transform coding technique [60-66].

The N x N transform matrix of DCT is given by equation 2.1

\[
[C_{i,j}] = \begin{cases} 
\frac{1}{\sqrt{N}} \cos \left( \frac{(2j+1)i\pi}{2N} \right) & i = 0, \ j = 0,1, \ldots, N - 1 \\
\frac{2}{\sqrt{N}} \cos \left( \frac{(2j+1)i\pi}{2N} \right) & i = 0,1, \ldots, N - 1, \ j = 0,1, \ldots, N - 1 
\end{cases} 
\]  

2.2 Image Compression Using Wavelet Transform

The Discrete Wavelet Transform (DWT) [67] is the transform adopted by the recent image compression standard JPEG2000 [16-19, 68] and is most popular transform employed in image coding nowadays. It significantly outperforms algorithms based on other transforms, such as the DCT. The success of the DWT lies in ease of computation and its decomposition of an image into spatial sub bands that facilitates the design of efficient quantization algorithms and allows exploitation of the human visual system characteristics. The main advantage of wavelet transforms is that they are capable of representing an image with multiple levels of resolution [69], and yet maintain the useful compaction properties of the DCT, therefore the subdivision of the input image into smaller sub images is no longer necessary as is done in DCT based coding.

An important property of wavelet transform is the conservation of energy (sum of square of pixel values). Wavelet transform results in energy of the image divided between approximation and details images, but the total energy remains constant. In lossy compression, loss of energy occurs because of quantization.
Another property of wavelet transform is energy compaction. The compaction of energy describes how much energy has been compacted into the approximation image during wavelet analysis. Compaction will occur wherever the magnitudes of the detail coefficients are significantly smaller than that of the approximation coefficients. Compaction is important when compressing signals because more the more energy compaction into the approximation image, the higher compression efficiency may be obtained.

Wavelet transform decomposes the signal into various sub bands, each of which has its own spatial orientation feature that can be efficiently used for image coding. Another property of wavelet transform is that quantization error $e_{m,n}$ introduced to coefficient $X_{m,n}$ will appear as a scaled version of wavelet function superimposed on the reconstructed signal. In image coding, this implies that a quantization error from coefficient will not remain confined to the location but will spread through reconstructed image with the shape of corresponding wavelet. Wavelet transform offers various benefits and wavelet coding has been at the core of the many state of art image coders.

2.2.1 Wavelet Transform

Wavelets are functions generated by dilation and translation of a single mother wavelet function $\psi(t)$ as given by equation 2.2.

$$\Psi_{a,b}(t) = \frac{1}{\sqrt{a}} \psi\left(\frac{t-b}{a}\right)$$  \hspace{1cm} (2.2)

where $\psi(t)$ has to satisfy the following conditions
Admissibility condition: \[ \int |\Psi(\omega)|^2 |\omega|^{-1} d\omega < \infty \], where \( \Psi(\omega) \) is Fourier transform of \( \psi(t) \). It implies that \( \Psi(\omega)|_{\omega=0}=0 \) i.e. \( \int \psi(t) dt = 0 \), which means that \( \psi(t) \) is zero mean and must be a wave.

Regularity conditions: That the wavelet function should have some smoothness and concentration in both time and frequency domains.

Discrete wavelet transform function given by equation 2.3 is obtained by putting \( a=2^{-j} \) and \( b=k2^{-j} \) in equation 2.2.

\[
\psi_{j,k}(t) = 2^{j/2} \psi(2^j t - k) \tag{2.3}
\]

Where the \( \psi_{j,k} \) constitute orthonormal basis, i.e., wavelet coefficients of function \( f(t) \)

\[
W_{j,k} = \int f(t) 2^{j/2} \psi(2^j t - k) dt \tag{2.4}
\]

The function \( f(t) \) can be reconstructed from wavelet coefficients as

\[
f(t) = \sum_{j} \sum_{k} W_{j,k} 2^{j/2} \psi(2^j t - k) \tag{2.5}
\]

The above two equations (equation 2.4 and 2.5) forms the synthesis and analysis equation of discrete wavelet transform (DWT). DWT is implemented using filter banks. A filter bank consists of filters, which separates signal into frequency bands of equal band [70]. Consider the filter bank shown in figure 2.1 where a discrete signal \( x(n) \) is applied to a system consisting of a pair of filter banks. Given this signal sequence \( x(n) \) and its corresponding z-transform \( X(z) \), a lower resolution signal can be obtained by low pass filtering with half band low pass filter having impulse response \( l(n) \) with z – transform \( L(z) \). Then half band signal can be made full band signal by down sampling by a factor of two. The “added details” of the signal can be computed in similar manner as high pass filter version of \( x(n) \) (using a filter with impulse response \( h(n) \) and z transform \( H(z) \), followed by down sampling by a factor of two as depicted in figure 2.1. At the receiving
end, the signal \( x'(n) \) is reconstructed using the filter \( l'(n) \) and \( h'(n) \) with z-transform \( L'(z) \) and \( H'(z) \), respectively.

For the error free channel, original signal can be reconstructed if and only if,

\[
L'(z) = 2L(z), \quad H'(z) = 2H(z) \quad \text{and} \quad H(z) = L(-z)
\]

Under these conditions z transform of the output \( x'(n) \)

\[
X'(z) = [L^2(z) - H^2(z)]X(z)
\]

Perfect reconstruction is guaranteed if and only if, \( L^2(z) - H^2(z) = 1 \)

If \( l(n) \) is a linear phase finite impulse response (FIR) filter with even number of coefficients then above condition holds if.

\[
|L(\omega)|^2 + |L(\omega-\pi)|^2 = 1
\]

A filter that meets above constraints are said to possess perfect reconstruction properties and are often called Quadrature Mirror Filters (QMF’s) or conjugate Mirror Filters (CMF’s) and is used for multi resolution sub band decomposition using wavelets [69].
2.2.2 Wavelet Representation of Image

Figure 2.2 shows one stage of sub band decomposition of an image. Wavelet based decomposition of an image can be interpreted as an image filtering process. For an image A of size $2^n \times 2^n$, and can be performed as follows: The wavelet filter $l(n)$ is a low pass filter with frequency response $L(\omega)$. Filtering the image A with $L(\omega)$ gives low frequency information or the background of the image, whereas $H(\omega)$ gives the high frequency information corresponding to edges in the image. Subsequent down sampling by a factor of two gives two sub bands, $L_{1r}A$ and $H_{1r}A$ of image A (subscript r denotes that filtering is applied on rows of the image). Since down sampling factor of two is applied in vertical direction of each sub band, the size of these two down sampled sub band is $2^n \times 2^{n-1}$. The filter $L(\omega)$ and $H(\omega)$ are then applied on columns of the images $L_{1r}A$ and $H_{1r}A$ followed by down sampling by two resulting in four sub bands each of size $2^{n-1} \times 2^{n-1}$; $L_{1c}L_{1r}A$ (low resolution sub band), $H_{1c}L_{1r}A$ (horizontal orientation sub band), $L_{1c}H_{1r}A$ (vertical orientation sub band), and $H_{1c}H_{1r}A$ (diagonal orientation sub band).

![Figure 2.2: Filter Bank for One Level Sub-band Decomposition Using DWT](image-url)
The low-resolution sub band contains the smooth information (low frequency) and background intensity of the image, while other sub bands contain the detail (high frequency) information of the image. Figure 2.3(a) and (b) shows one stage and two-stage image decomposition structure. Reversing the same process can perform Reconstruction of the original image. The one stage and two-stage wavelet decomposition of image “LENA” is shown in figure 2.3(c) and (d).

Figure 2.3 Illustration of
(a) One level sub band decomposition (b) two level dyadic sub band decomposition
(c) One level decomposition of ‘LENA’ (d) two level dyadic decomposition of ‘LENA’
2.3 Wavelet based Image Coding Algorithms

Although use of wavelet transform in image coding was reported in literature [27,71], but EZW [22] has laid foundation for wavelet based efficient embedded image coders. SPIHT [23] is the more efficient and advanced version of EZW and has been very popular. SPECK [30] is a block based image coder, which has less complexity and memory requirement than SPIHT. Since then, many wavelet coders were proposed in literature. The recent image-coding standard JPEG 2000 is based on DWT, due to its compression efficiency among various other features. Both EZW and SPIHT are tree-based coders while SPECK is a block-based coder. A low-complexity implementation of SPECK, called SBHP [33] has been included in the framework of JPEG 2000.

2.3.1 JPEG 2000

JPEG 2000 [16-19, 68], is the recent image compression standard based on wavelet transform given by the Joint Photographic Experts Group committee in 2000 which replaces their original DCT-based JPEG standard. The JPEG 2000 is an image compression algorithm which after image tiling (dividing into rectangular region) these tiles are transformed using CDF 9/7 wavelet transform, for lossy compression and CDF 5/3 wavelet transform for lossless compression. This process results in collection of sub bands on various resolution levels. The quantized sub-bands are split into code blocks. These code blocks are encoded starting from most significant bit plane to the lower bit planes employing EBCOT [20] algorithm. Encoding of each bit plane consists of three passes; Significance Propagation, Magnitude Refinement, and Cleanup pass. Significance propagation pass encodes bits and signs of insignificant
coefficients with significant coefficient as neighbors. Then magnitude refinement encodes refinement bits of coefficients found significant in earlier bit planes. Lastly, in cleanup pass coefficients without any coefficients found significant in earlier bit planes are encoded.

Finally the bits generated in sorting pass are then encoded by a context based (context formed from its neighbors) binary arithmetic coder. The resulted bit stream is then split into packets, consisting of bits of group of code blocks. Packets containing less significant bits may be discarded to achieve bit budget. Packets from all sub-bands are then collected in ‘layers’, in such a way that the image quality increases with decoding of each layer, thereby supporting progressive transmission.

The JPEG-2000 encoding algorithm is very efficient and has many desirable features. However, it is a complex and computationally intense and has limited embeddedness [72-73], therefore it is not suitable for resource-constrained environment.

2.3.2 SPIHT

SPIHT [23] was proposed by A. said and Pearlman in 1996. The SPIHT coding is modified and improved version of the EZW algorithm that achieves higher compression and better performance than EZW. The algorithm is based on a simple ordering of the coefficients by magnitude and bit plane, with transmission order determined by a set partitioning algorithm (SPIHT). SPIHT achieve compression by aggregating a large number of insignificant coefficients in spatial orientation trees (SOT) also called as zero trees (figure 2.4). A zero tree is based on the property that if a
coefficient is insignificant; it is very likely that its descendants in higher frequency sub bands are also insignificant. SPIHT exploit inter-band correlations among transform coefficients and achieves excellent rate–distortion performance and provides an embedded bit-stream that allows precise bit rate scalability i.e. Progressiveness.

Significance information in SPIHT algorithm is stored in three data dependent ordered lists: a list of insignificant pixels (LIP), a list of insignificant sets (LIS), and a list of significant pixels (LSP). The algorithm initializes by adding the coefficients in the lowest frequency sub band (LL-band) in LIP, and those with descendents are added to LIS as type ‘A’ entries. The LSP starts as an empty list.

The SPIHT coding of wavelet coefficients involves a sequence of sorting and refinement passes applied with decreasing thresholds. In the sorting pass, significance decision of a set or coefficient with respect to the current
threshold is coded. For a significant coefficient found for the first time, sign bit is immediately transmitted. After the sorting pass, the refinements pass adds resolution to the coefficients that have been found significant earlier with a current threshold.

The algorithm starts with the most significant bit plane and proceeds toward the finest resolution. At every bit plane, the sorting pass goes through the LIP followed by LIS and then refinement pass goes to LSP. For each coefficient in LIP, one bit is used to describe its significance with respect to current threshold. If the coefficient is not significant, it remains in LIP; otherwise, the sign bit is transmitted and the coefficient is moved to LSP. After LIP, sets of LIS are scanned. Each set (Spatial orientation tree) of LIS is tested against threshold and significance is encoded. A significant type ‘A’ set will be partitioned into four offspring coefficients and a type ‘B’ set (grand descendants), if the grand descendant set exists; the type ‘B’ set is added to the end of LIS, while four coefficients are tested for their significance against current threshold and added to LIP and LSP accordingly. Significant types ‘B’ set is partitioned into four type ‘A’ sets (with offspring coefficients as corresponding nodes) and are added to the end of LIS. The partitioning of type ‘A’ set and type ‘B’ set of SPIHT is shown in figure 2.5. Finally, in refinement pass, each coefficient of LSP, except those added in current bit plane, is refined with one bit. Then the threshold is divided by two, and the algorithm then repeats the above procedure. Effectiveness of the algorithm can be further enhanced by entropy coding its output, but at the cost of a larger encoding/decoding time.
A major problem of the SPIHT coder is that the algorithm uses data dependent lists to keep track of sets and coefficients to be tested for significance against threshold. The use of lists in SPIHT causes a dynamic memory requirement, and added computational complexity for appending/sorting of memory. The data dependant memory requirement caused by the linked list structures in the SPIHT algorithm can be eliminated by fixed size memory (static memory) that keeps track of the state information of sets and coefficient, thus, it enables a fast scanning of the bit planes. Various listless implementation of SPIHT has been reported in research papers that use state memory to keep track of sets/coefficients for significance testing. NLS [48] proposed by wheeler and Pearlman uses 4 bits per coefficients state memory, while implementation in [51] requires 3 bits per coefficients state memory to facilitate coding.

Figure 2.5 : (a) partition of Type A SOT of SPIHT (b) partition of Type B SOT of SPIHT
2.3.3 SPECK

SPECK [30] proposed by Pearlman et al., is bit plane coding algorithm, and encodes significance map of bit planes in decreasing order of their importance. SPECK coder uses two types of set: $S$ and $I$ set as shown in figure 2.6. In the process of coding, $S$ sets are partitioned by quad partitioning scheme while $I$ sets are partitioned by octave band partitioning scheme. Each pass of SPECK comprises of sorting, and refinement. It uses two lists: list of insignificant sets (LIS) and list of significant pixels (LSP) to store state of sets and coefficients. The coding algorithm proceeds as follows.

The algorithm initializes with defining initial threshold, which depends on the bit plane. The LIS is initialized with LL-sub band. In the sorting pass, sets of LIS are tested against threshold and their significance is encoded. A significant set is partitioned into four equal sets and parent set is removed from the LIS. Each of the newly formed set is tested for their
significance and insignificant is added to LIS while significant set is recursively partitioned until a coefficient is reached. For a significant coefficient, sign bit of the coefficient is also coded and coefficient is send to LSP. After all sets of LIS are tested for the significance, set $I$ is tested for significance. A significant I set is partitioned the octave band partitioning resulting in three sets and a reduced $I$ set. The newly formed sets are processed in regular image scanning order.

After all sets of LIS are processed, the refinement pass is initiated which refines the quantization of the coefficients in the LSP found significant in earlier passes. The threshold is then reduced by a factor of two and the sequence of sorting and refinement passes is repeated. Sets of LIS are processed in increasing order of their size. The whole process is repeated until the desired bit rate is achieved.

State Memory: SPECK use linked lists to store the significant information of coefficients, and blocks there by requiring a data dependent memory. The required memory size for SPECK is proportional to the number of entries in the corresponding lists (LIS and LSP). In SPECK, each entry in LIS has the address of a square block of arbitrary size including that of a single coefficient. In actual implementation, a separate list is maintained for each block size, therefore size of block set is not stored. However, LSP contains the address of significant coefficient. Let $N_{\text{LIS}}$ and $N_{\text{LSP}}$ be the number of entries in LIS and LSP respectively and ‘s’ be number of bits required to store addressing information, then the total required memory due to lists in SPECK is given by equation 2.6.

$$M_{\text{SPECK}} = 8[N_{\text{LIS}} + N_{\text{LSP}}] \text{ bits} \tag{2.6}$$
In worst case, all the coefficients may be in either LIS or in LSP in SPECK coder. For an image size of 512x512, worst-case state memory required for SPECK is 576 KB.

Though SPECK is an efficient algorithm, large dynamic state memory requirement and computational complexity limits application of SPECK coder in memory constrained environment such as handheld multimedia devices, VSN and WMSN’s.

### 2.2.4 WBTC Coding

WBTC [35] proposed by A. A Moinuddin and E. khan, is a wavelet based image coding algorithm, which exploits redundancies among sub band along with partial exploitation of redundancies within sub band. It is a bit plane coding algorithm where magnitude and bit plane ordering of the coefficients firstly transmits the coefficients that yield the largest decrease in mean squared error distortion.

The coefficients of L-level dyadic wavelet transformed image are grouped together in blocks of size m x n and then block trees are formed with roots in the topmost (LL-) sub band. A block tree is a set of all descendent blocks of a node block. In the LL- band, out of each group of 2x2 blocks, one (top left) block has no descendents, and each of the other three blocks has four offspring blocks in the corresponding orientation high resolution sub bands, A block tree is referred by a node block and each node block has four offspring blocks. Set of all descendent blocks are referred as type ‘A’ block tree, while set of grand descendent blocks are referred as type ‘B’ block tree as shown in Fig 2.7.
WBTC [35] is a bit-plane coding algorithm and comprises of two main stages within each bit-plane; sorting and refinement pass. WBTC uses three ordered lists; list of insignificant blocks (LIB), list of insignificant block sets (LIBS) and a list of significant pixels (LSP) to store state of blocks/trees and coefficients.

The coding process of WBTC starts with the most significant bit plane and proceeds from coarsest resolution towards the finest resolution. The algorithm is initialized with LSP as an empty list, LIB with blocks of LL band and LIBS with blocks in LL band having descendents as type ‘A’ entries. Sorting pass begins with significance testing of blocks in LIB. Significance of block is encoded using one bit. An insignificant block (zero blocks) is encoded with ‘0’ and blocks remain in the LIB. A significant block is encoded with ‘1’ and is partitioned into four adjacent block using quad partitioning recursively till partitioning is needed or block of size (2x2) is reached. For a significant 2x2 block significance of all the four coefficients are encoded and if a coefficients is found significant its sign bit is coded and the coefficient is moved to LSP. All insignificant blocks

![Spatial orientation Block tree of WBTC and a SOT of SPIHT](image-url)
resulting from quad partitioning are added in LIB while parent block is removed from LIP. A coefficient is treated as 1x1 block. The encoder then traverses the LIBS and performs significance test on each block tree in LIBS. Insignificant block trees remain in LIBS while a significant type ‘A’ tree is partitioned into a type ‘B’ set and four offspring blocks. The type ‘B’ set is added to the end of LIBS while the four offspring blocks (each of the same size as the root block) are immediately tested for significance in the same manner as if they were in the LIB. A significant type ‘B’ set will be partitioned into four type ‘A’ sets with offspring blocks as node block and all of them are added to the end of LIBS. After each sorting pass, the coefficients in LSP, except those just added in current bit plane, are refined with one bit. The threshold is then divided by a factor of two and then above procedure is repeated until the desired bit rate is achieved or all the bit planes are encoded.

It should be noted that here for a block size of 1x1 (single coefficient), WBTC [35] leads to the SPIHT [23] algorithm. WBTC combines many clustered zero trees of the SPIHT, which may occur in the early passes and intra- sub band correlations are also partially exploited by zero blocks with reduced memory requirement and encoding time in comparison to SPIHT.

Like all zero tree-based coders, WBTC use linked lists to store the significant information of coefficients, blocks and block trees there by requiring a data dependent memory that necessitates need of memory management. The required memory size for WBTC is proportional to the number of entries in the corresponding lists (LIB, LIBS and LSP). In WBTC, each entry in LIB is the address of a square block of arbitrary size including that of a single coefficient. In actual implementation, a separate
list is maintained for each block size. Each entry in LIBS is the address of a block of size $\beta = n \times n$ and its type (‘A’ or ‘B’). However, LSP contains the address of significant coefficient. Let

- $N_{LIB}$ = Total number of nodes in LIB
- $N_{LIBS}$ = Number of nodes in LIBS
- $N_{LSP}$ = Number of nodes in LSP
- $s$ = Number of bits to store addressing information of a block node

Then, the total required list memory size in WBTC is given by equation 2.7

$$M_{WBTC} = s(N_{LIB} + N_{LIBS} + N_{LSP}) + N_{LIBS}$$

(2.7)

Though WBTC offers superior performance than SPIHT due to partial exploitation of zero blocks. However, WBTC requires slightly lower run time memory (dynamic memory) than SPIHT, but its data dependent memory is severe constraint for low memory applications. In addition, multiple memory access and memory management increases the complexity. The large run time memory requirements and increased complexity limits its application in constrained environments such as handheld multimedia devices, VSN and WMSN’s.

### 2.4 Linear Indexing

Let an image $X$ of size $(R, C)$ with $R=C=2^p$ being transformed using $L$ level of dyadic lifting wavelet transform. The transformed image $\{C_{i,j}\}$ is of size $(R, C)$ where a coefficient is addressed by zero based row and column index $r$ and $c$ respectively.

The linear index is a single number to address a coefficient. The two dimensional array $\{C_{i,j}\}$ of size $R \times C$ can be converted into one
dimensional array of zero based index $i$, varying in the range of 0 to $RC-1$ by simply bits interleaving of binary representation of row and column index $r$ and $c$. The dyadic wavelet transformed image $\{C_{i,j}\}$ is read into linear array $\{C_i\}$ with zero based index $i$. Linear indexing allows indexing of transformed coefficients in Z scan order. Fig. 2.8(a) shows an example of this indexing scheme in Z scan order for an $8\times8$ image. While figure 2.8(b) shows sub bands at different resolution levels are shown in Z-scan order.

An important property of the linear indexing is that it efficiently supports the operations on coefficient positions with one computation instead of two required in 2-D wavelet transform array, needed for block tree/quad partitioning of tree-based/block-based algorithms. Figure 2.9 demonstrates quad partition of blocks, while figure 2.10 describes block tree partitioning in 2-D and in linear indexing. The set structures uses in SPECK and WBTC in 2-D wavelet transform and in linear indexing is shown in table 2.1.
Figure 2.9: Quad partitioning of a block in
(a) 2 D indexing (b) linear indexing

Figure 2.10: Partition of a Block tree in
(a) 2-D indexing (b) linear indexing
Table 2.1: Set structures in 2D indexing and linear indexing

| L-level dyadic wavelet transformed 2D array \(|C_{ij}|\) of size \((R \times C)\); \(R=C=2^p, p \) is an integer | L-level dyadic wavelet transformed linear array \(|C_{i}|\) of length \((N_{pix}=RC=4^p, p \) is an integer) |
|---|---|
| L-band size | \(2^{(p-L)} \times 2^{(p-L)}\) |
| S Block Size | \(m \times m\) |
| \(\beta = m^2\) |
| S Block \(B_{k,m}^m = \{c_{ij}: k \leq i < k + m, l \leq j < l + m\}\) | \(B_k^p = \{c_i: k \leq i < k + \beta\}\) |
| Quad Partition of S block | \(Q_{k,m}^{m} = \left\{ \begin{array}{c} B_{2k,2l}^{m} \quad \quad \quad \quad \quad \quad \quad B_{2k+2m,2l}^{l} \\ B_{2k+m,2l}^{m} \quad \quad \quad \quad \quad \quad B_{2k,m+2l}^{l} \end{array} \right\} \) |
| \(Q_k^p = \left\{ \begin{array}{c} B_{k+2\beta,4}^{2} \quad B_{k+2\beta,4}^{2} \\ B_{k+2\beta,4}^{2} \quad B_{k+2\beta,4}^{2} \\ B_{k+2\beta,4}^{2} \quad B_{k+2\beta,4}^{2} \end{array} \right\} \) |
| I-set | \(I_q = \{C_{i,j}\} - 2^{(p-L-q)} ; \ 0 \leq q < L\) |
| \(I_q = \{C_{i,j}\} - S_0^{(p-L-q)} ; \ 0 \leq q < L\) |
| Type ‘A’ tree \(D_{k,n}^{m} = \{O_{k,n}^{m}, L_{k,n}^{m}\}\) | \(D_k^p = \{O_k^p, L_k^p\}\) |
| A Set of Offspring blocks | \(O_{k,l}^{m} = \{B_{2k,2l}^{m}, B_{2k+m,2l}^{m}, B_{2k,2m+l}^{m}, B_{2k+m,2m+l}^{m}\}\) |
| vertical SOBT | \(O_k^p = \{B_{4k+2\beta,4}^{2}, B_{4k+2\beta,4}^{2}, B_{4k+2\beta,4}^{2}, B_{4k+2\beta,4}^{2}\}\) |
| diagonal SOBT | \(O_{k,l}^{m} = \{B_{2k,2l}^{m}, B_{2k+m,2l}^{m}, B_{2k,2m+l}^{m}, B_{2k+m,2m+l}^{m}\}\) |
| Type ‘B’ tree | \(I_{k,j}^{m} = \{B_{2k,2l}^{m}: 2^j + 2^j m \leq 2^p, j > 1\}\) |
| horizontal SOBT | \(L_k^p = \{B_{4k+4\beta,4}^{2}: 4^j k + 4^j \beta \leq 2^p\}\) |
| \(I_{k,j}^{m} = \{B_{2k,2l}^{m}: 2^j k + 2^j m \leq 2^p, j > 1\}\) |
| vertical SOBT | \(I_{k,j}^{m} = \{B_{2k,2l}^{m}: 2^j k + 2^j m \leq 2^p, j > 1\}\) |
| diagonal SOBT | \(I_{k,j}^{m} = \{B_{2k,2l}^{m}: 2^j k + 2^j m \leq 2^p, j > 1\}\) |

2.6 Listless SPECK Image Coding Algorithm

Reduced memory listless SPECK image compression (Listless SPECK) LSK [53] is listless version of SPECK [30] algorithm but does not use I partitioning of SPECK. State information of coefficient /block is kept in a fixed size array, with two bits per pixel of image to enable fast scanning of the bit planes. In LSK, efficient skipping of blocks of insignificant coefficients is accomplished using linear indexing [56]. The following
markers are placed in the 2 bit per coefficient state table mark, to keep track of the set partitions.

MIP: The coefficient is insignificant or untested for this bit-plane.
MNP: The coefficient is newly significant & will not be refined for this bit-plane.
MSP: The coefficient is significant and will be refined in this bit-plane.
MS²: The block of size 2×2, i.e., 4 elements to be skipped.

MS² markers are successively used to skip 4×4, 8×8 block, and so on.

LSK coder uses fixed size array of two bits per coefficient to store markers facilitating coding. For a 512×512 image decomposed to 5 levels, the LSK requires 64 KB memory to store markers while for image size of 1024×1024, it is 256 KB, which is significantly high for memory, constrained handheld devices. In addition, LSK does not use I sets of SPECK thereby generating more bits in earlier passes.

LSK uses a static memory of size equal that of coefficients array to store markers. As LSK uses two bit markers then for a image size (N x N) memory required for marker is given by equation 2.8.

\[ M_{LSK} = 2N^2 \] bits

(2.8)

2.7 Image Compression for Wireless Sensor networks

Wireless sensor networks with imaging capabilities are highly constrained in terms of memory, processing capability, and battery power and they need to transmit images on wireless networks, which suffer by high probability and burstyness of noise. It is also important that computational cost of compression is less than the related cost of
transmission to enhance energy efficiency and elongate life of network. Therefore, a fast and efficient image compression algorithm with low memory requirement, and bit rate scalability (embeddedness) is desirable for implementation on these networks [39].

The characteristics of the various image compression algorithms are presented in [9,10,74]. Although JPEG 2000 provides highest coding efficiency, JPEG2000 compression is very time and energy consuming [74] and is therefore not suitable for implementation in recourse-constrained environment. In [75], it is shown that compression and transmission, using JPEG is more energy inefficient than transmitting the image without compression with high quality level. In [40], It has been concluded that SPIHT gives highest coding efficiency while achieving the best compromise between quality of reconstructed image and energy consumption. In addition, SPIHT is preferred over EBCOT coding for hardware implementation [76]. SPECK coder because of its simplicity, coding efficiency and low memory, is more suitable candidate among the wavelet based embedded coders for implementation in WMSN and VSN [10,39]. Low memory version of SPECK image-coding algorithm is desired for resource constrained handheld multimedia devices, and VSN/ WMSN.