Module 5

EMBEDDED WAVELET CODING

Lesson 15
EBCOT Algorithm

Instructional Objectives

At the end of this lesson, the students should be able to:

1. Explain the inadequacies of EZW and SPIHT in wavelet packet encoding.
2. Define resolution scalability of embedded bit-stream.
3. Define SNR scalability of embedded bit-stream.
4. State the basic characteristics of EBCOT algorithm.
5. Explain the rate-distortion optimization problem of code blocks.
6. Explain how the truncation points are selected in EBCOT bit-stream.
7. State the highlighting features of block coding algorithm.
8. Define the significance of sub-blocks.
9. Explain the quad-tree structure representation of sub-block significance.
10. Name the four coding passes in block coding algorithm.
11. State the role of each of the coding passes in block coding algorithm.
12. Define the four coding primitives used in block coding algorithm.
13. Explain the formation of the quality layers from the embedded code block bit stream.

15.0 Introduction

In lesson-13 and lesson-14, we have studied two similar approaches to coding of wavelet (in general, subband) coefficients, namely Embedded Zerotree Wavelet (EZW) and Set Partitioning in Hierarchical Trees (SPIHT). Both these approaches use ordering of coefficients by magnitudes for encoding the coefficients in an embedded bit-stream and exploit the self-similarity of coefficients across subbands. The latter approach is more efficient in terms of coding efficiency, as compared to the former, since it doesn't require explicit transmission of ordering information of the coefficients by efficiently partitioning the subsets of coefficients in spatial orientation trees. However, both EZW and SPIHT can only be applied to dyadic partitioning of coefficients, in which only the LL subband at a decomposition level are further analyzed. This is regarded as a limitation, since these two algorithms cannot be applied to wavelet packets in general. Wavelet packets allow more flexible decomposition of subbands and the subbands at higher frequencies can also be decomposed into narrower bands. Moreover, these two approaches only offer SNR scalability by encoding all the subbands at a given precision in an iteration of passes (dominant -subordinate passes in EZW and sorting-refinement passes in SPIHT). These algorithms do not offer any resolution scalability in the sense that we do not complete the

encoding at a given resolution and then do the encoding at the next higher resolution and so on.

In this lesson, we are going to discuss a more recent approach to embedded wavelet coding, namely Embedded Block Coding with Optimized Truncation of bit-stream (EBCOT), which can be applied to wavelet packets and which offers both resolution scalability and SNR scalability. Because of its advantages, the EBCOT algorithm has been accepted incorporated within the most recent still image compression standard JPEG-2000. In this lesson, we shall first define resolution and SNR scalabilities and their combinations. This will be followed by the basic objectives of EBCOT. The algorithm selects the truncation points, that is, where the bit-stream can be terminated based on rate-distortion (R-D) optimization. The encoding primitives and the steps of encoding will be explained later in the lesson.

15.1 Resolution Scalability

Before defining resolution scalability, let us consider an example subband packet decomposition structure, as presented in fig.15.1.

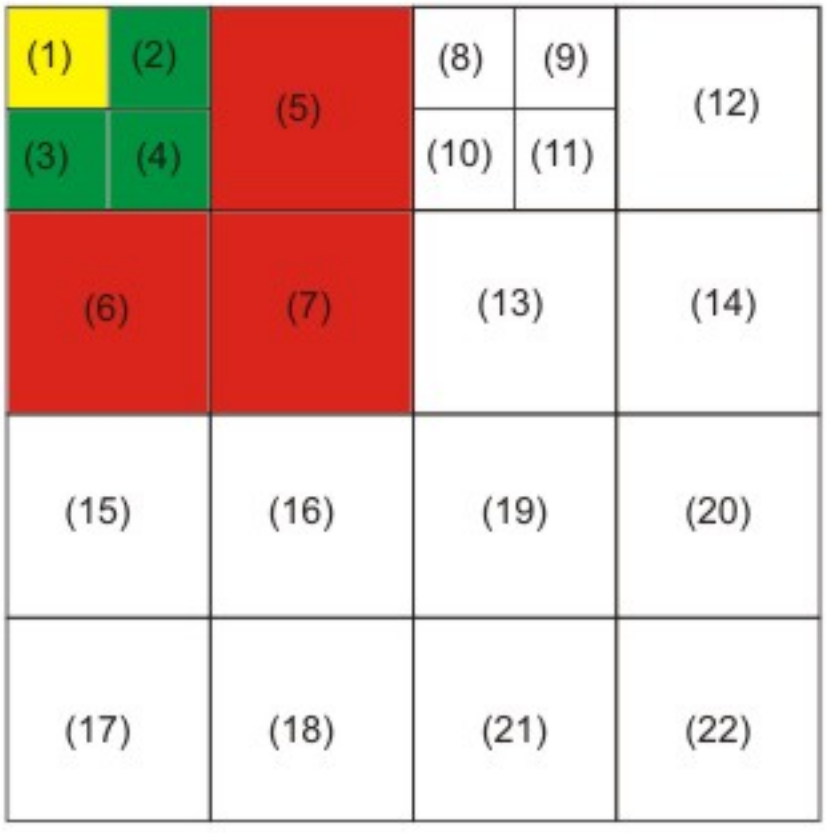

Fig 15.1: *Example subband decomposition structure*

These subbands are grouped into different resolution levels $L_0, L_1, L_2, \cdots\cdots L_V$ where L_0 corresponds to the lowest resolution level and contains only the single LL subband (1), indicated in yellow colour. The next resolution level is L_1, which is green in colour and includes subbands 2, 3 and 4. The other resolution levels are $L_2 = \{5,6,7\}$ (red), $L_3 = \{8,9,10,\cdots\cdots,21,22\}$ (white).

Fig 15.2: *Resolution scalable bit stream*

If an embedded bit-stream contains distinct subsets representing each resolution level, the bit-stream is called resolution scalable. In the illustrative diagram of fig.15.2, B_l represents the bit-stream contributed by resolution level l. The embedded bit-stream starts with the lowest resolution level corresponding to coarsest representation and progressively the higher resolution bit-stream follows.

15.2 SNR Scalability

An embedded bit-stream is said to be SNR scalable if it contains distinct subsets B_q such that $\bigcup_{k=0}^{q} B_k$ together represents the samples from all the subbands at some quality (SNR) level-q. The best examples that can be cited are the EZW and the SPIHT algorithms. There, what we generate at the end of each dominant-subordinate or sorting-refinement pass is a SNR scalable bit-stream, since all encoded significant coefficients correspond to a bit plane.

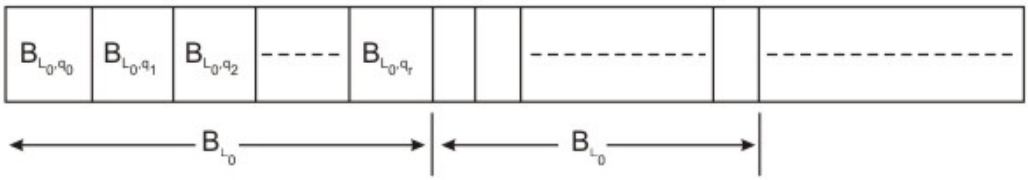

Fig 15.3: *Both resolution and SNR scalable bit-stream.*

A bit-stream is said to be both resolution and SNR scalable, if it contains distinct subsets $B_{l,q}$ which hold the quality refinements in resolution level L_l. Fig.15.3 illustrates a bit-stream that is both resolution and SNR scalable. The EBCOT algorithm incorporates both these forms of scalability.

15.3 Basic Characteristics of EBCOT algorithm

Before we describe EBCOT algorithm is detail, we list some of its basic characteristics:

- It offers both resolution and SNR scalability.
- It has a *random access* attribute. Given any region of interest and a wavelet transform with finite support kernel, it is possible to identify the regions within each subband.
- Each subband is partitioned into small non-overlapping block of samples, known as *code blocks*. EBCOT generates an embedded bit-stream for each code block. The bit-stream associated with each code block may be truncated to any of a collection of rate-distortion optimized truncation points.
- A code block is encoded by a two-tier coding structure, as shown in fig.15.4. In the first stage, a small amount of summary information is collected during the generation of each code block's embedded bit-stream. Based on this summary information, the truncation points for each code block are determined, using which the quality layers are formed in the second stage.

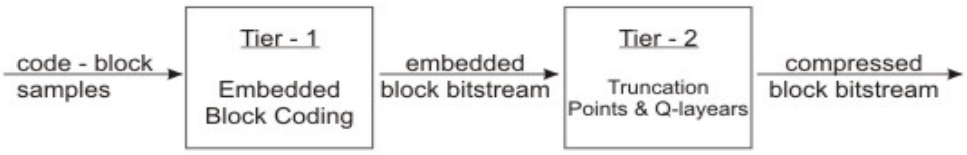

Fig 15.4: *Two-tier coding structure of EBCOT.*

- The EBCOT algorithm works on a *layered bit-stream* concept, where each layer corresponds to a quality and a collection of these layers form a rate-distortion optimized image.
- A single bit-stream offers a range of scalability – single layer possesses resolution scalability, few layers target at some specific bit-rate of interest and a large number of layers offer SNR and resolution scalability with random access attributes.

15.4 Rate-Distortion Optimization

Suppose that a truncation point n_i is selected for a code-block B_i. The rate and the distortion for the code-block B_i associated with this truncation point are given

by $R_i^{n_i}$ and $D_i^{n_i}$ respectively. The distortion metric used is usually the mean square error and is additive over all the code blocks. Thus, the overall distortion in the image is given by

$$D = \sum_i D_i^{n_i} \quad \text{..... (15.1)}$$

which is to be minimized subject to the constraint on the maximum bit-rate $R^{\max}$, given by

$$R^{\max} \geq R = \sum_i R_i^{n_i} \quad \text{..... (15.2)}$$

The Rate-Distortion (R-D) optimization problem requires minimization of $D + \lambda R$, where λ is a Lagrange multiplier. If we can find a value of λ such that the truncation points $\{n_i^{\lambda}\}$ selected from different code blocks achieve $R = R_{\max}$, we can say that $\{n_i^{\lambda}\}$ is the optimal set of truncation points. Since the truncation points are discrete, it is not possible to exactly make $R = R_{\max}$. In practice, the code blocks are small in size and there are many truncation points. It is sufficient to determine the smallest value of λ such that $R \leq R_{\max}$.

We have separate minimization problem for each code block B_i. To determine the truncation point n_i^{λ}, which minimizes ($D_i^{n_i^{\lambda}} + \lambda R_i^{n_i^{\lambda}}$) for all values of λ, we select a set N_i of feasible truncation points, enumerated by $j1 < j2 < \cdots\cdots$ whose rate-distortion slopes $S_i^{jk} = \Delta D_i^{jk} / \Delta R_i^{jk}$ (where $\Delta D_i^{jk} = D_i^{jk-1} - D_i^{jk}$ and $\Delta R_i^{jk} = R_i^{jk} - R_i^{jk-1}$) are strictly decreasing. Then, the optimal selection of truncation points is given by $n_i^{\lambda} = \max\{jk \in N_i \mid S_i^{jk} > \lambda\}$. This unique set may be determined through a process of convex hull analysis.

15.5 Block Coding Algorithm

The block coding algorithm generates separate embedded bit-stream for every code block. Some highlighting features of the block coding algorithm are:

- Uses the concept of “fractional bit plane”, in which every bit-plane is encoded in multiple numbers of passes.
- Uses context-sensitive arithmetic coding.
- Code blocks are further subdivided into sub-blocks whose significance are efficiently encoded prior to sample by sample encoding.

Before presenting the algorithm, we introduce you to the symbols used in the encoding process:

$s_i(k_1,k_2)$: 2-D sequence of subband samples belonging to the code block B_i. For the LL, LH and HH subbands, k_1 and k_2 represent the horizontal and the vertical positions respectively. The HL subband is transposed, so that k_1 and k_2 represent the vertical and the horizontal positions respectively.

$\chi_i(k_1,k_2)$: The sign of the subband sample $s_i(k_1,k_2)$. Hence, $\chi_i(k_1,k_2) \in \{1,-1\}$

$\upsilon_i(k_1,k_2)$: Quantized magnitude of the subband sample, given by

$$\upsilon_i(k_1,k_2) = \left\lfloor \frac{s_i(k_1,k_2)}{\delta_{\beta_i}} \right\rfloor$$

where, δ_{β_i} is the quantization step-size for the subband to which the code-block B_i belongs.

$\upsilon_i^p(k_1,k_2)$: The *p* th bit-plane of $\upsilon_i(k_1,k_2)$. $p=0$ corresponds to the least significant bit and $p_i^{\max}$ is the most significant bit.

$\sigma_i(k_1,k_2)$: A binary state-variable, indicating the significance map. Its entry is initialized to zero, but is set to one, when the relevant sample's first non-zero bit-plane $\upsilon_i^p(k_1,k_2)=1$ is encountered.

15.5.1 Sub-blocks and their significance:

Every code-block is further subdivided into sub-blocks, each of which is typically of the size 16 x 16. For each bit-plane $p_i^{\max} \geq p \geq 0$, the significance of the sub-blocks are represented in a quad-tree structure, where the sub-blocks belong to the leaf nodes of the quad-tree $B_i^0(j_1,j_2)$ (j_1, j_2 are the coordinates of the sub-block and 0 is the level of the leaf node). The nodes at the level-*t* in the tree are formed from the nodes at the level- (*t*-1) as

$$B_i^t(j_1,j_2) = \bigcup_{z_1,z_2 \in \{0,1\}} B_i^{t-1}(2j_1+z_1, 2j_2+z_2) \quad \text{where}, 0 \leq t \leq T$$

where, *T* is the level corresponding to the root of the tree, given by $B_i^T(0,0)$, which indicates the entire code-block. The quad-tree data structure for the sub-blocks is illustrated in fig.15.5.

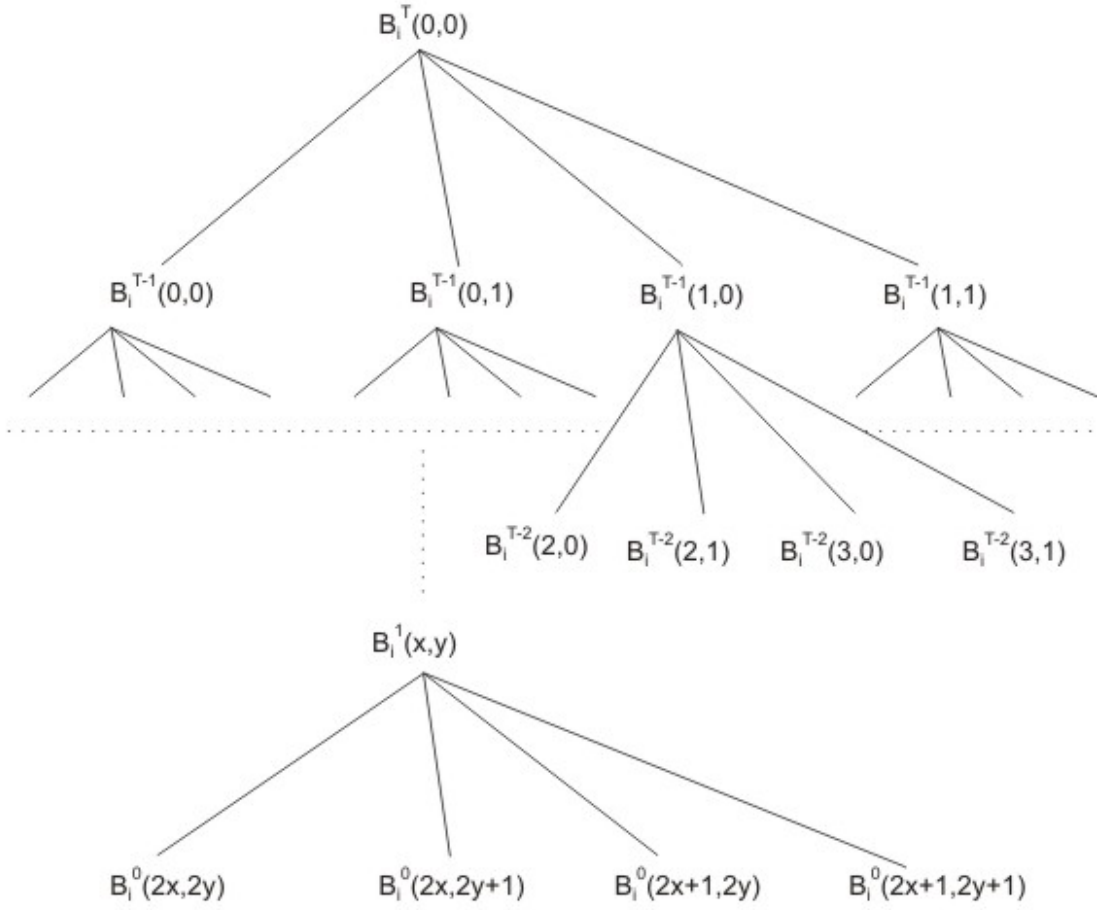

Fig. 15.5 Quadtree data structure for subblocks

The significance of a sub-block $B_i(j_1, j_2)$ for a bit-plane *p* is defined as follows. If any of the quantized magnitudes $\upsilon_i(k_1, k_2) \geq 2^p$ for $(k_1, k_2) \in B_i(j_1, j_2)$, the sub-block $B_i(j_1, j_2)$ is defined as significant for the bit-plane *p*. The significance of a sub-block is propagated to the nodes at the higher levels in the quad-tree, which means that the node $B_i^t(j_1, j_2)$ is significant, only if at least one of its descendant sub-blocks is significant. The significance of a quad-tree node at level-*t* for bit-plane *p* is indicated by $\sigma^p\left(B_i^t(j_1, j_2)\right)$. The significance of the quad-tree data-structure is encoded by arithmetic coding. If a quad-tree node is insignificant, the significance of its descendants need not be encoded. Also, if a quad-tree node is significant in the previous bit-plane that is, (*p*+1), it will remain significant in all the bit-planes from *p* onwards.

The encoding of the sub-block significance, sub-blocks containing one or more significant samples are identified and this leads to an efficient coding, since all

other sub-blocks which are insignificant, are by-passed in the remaining coding phases for the bit-plane.

15.5.2 Coding Passes:

The embedded bit-stream for each bit-plane belonging to a code block is generated in four different passes in the following order, namely (a) Forward significance propagation pass $\left(P_1^p\right)$, (b) Reverse significance propagation pass $\left(P_2^p\right)$, (c) Magnitude refinement pass $\left(P_3^p\right)$and (d) Normalization pass $\left(P_4^p\right)$. The quad-tree significance code S^p for the p th bit-plane is sent before the normalization pass P_4^p, so that the coefficients that become significant in bit plane-p are ignored until P_4^p. The generation of the embedded bit stream starts with the most significant bit-plane $p_i^{\max}$ and proceeds up to the least significant bit-plane p_i^0.

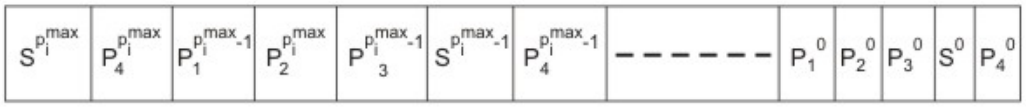

Fig 15.6: *Apperance of coding passes in EBCOT bit-stream.*

Fig.15.6 illustrates the bit-stream generation order. The coding passes are described below:

(a) Forward significance propagation pass $\left(P_1^p\right)$: In this pass, the coefficients in a code block are visited in scan line. Those which are insignificant till the previous bit-plane, that is $\upsilon_i^p(k_1,k_2) \le 2^{p+1}$and have a *preferred neighborhood* are coded and the rest are skipped. For LL, LH and HL subbands (note that the HL subband coefficients are already transposed, as described earlier.), the sample $s_i(k_1,k_2)$ is said to have a preferred neighborhood, if at least one of its horizontal neighbors is significant, that is $\sigma_i(k_1 \pm 1,k_2)$=1. The HH subband's sample $s_i(k_1,k_2)$ is said to have a preferred neighborhood, if at least one of its four diagonal neighbors are significant, that is, $\sigma_i(k_1 \pm 1,k_2 \pm 1)$=1. To each such sample, one of the two coding primitives, namely Zero Code (ZC) or Run Length Code (RLC) (to be described shortly) is applied to indicate if the sample first becomes significant in the current bit-plane *p*. If so, another coding primitive, Sign Code (SC) is invoked to encode the sign of the coefficient. This pass is referred to as significance propagation, since, the coefficients

already found to be significant serve as seeds and propagate their significance in the direction of their scan.

(b) Reverse significance propagation pass $\left(P_2^p\right)$**:** This coding pass is identical to P_1^p except for the order of scanning the coefficients which is reverse. Also, the concept of preferred neighborhood is extended to eight-connected neighbors of the sample.

(c) Magnitude refinement pass $\left(P_3^p\right)$**:** In this pass, the coefficients which are already found to be significant are encoded using the Magnitude Refinement (MR) primitive (to be described shortly).

(d) Normalization pass $\left(P_4^p\right)$**:** All coefficients which were skipped in the earlier three passes are encoded in this pass. This includes coefficients which are insignificant till the previous bit-plane and do not have any preferred neighborhood. To encode such coefficients, we make use of RLC and SC coding primitives.

Although the original EBCOT algorithm proposed by Taubman uses the four passes listed above, in the JPEG-2000 image coding standard, the four passes are simplified to three by using only one significance pass, instead of the forward and the reverse and using eight neighbors for preferred direction.

15.5.3 Coding Primitives:

To encode the code block coefficients, one of the following four coding primitives are used:

- **Zero Coding (ZC):** This primitive is used in the significance propagation passes to encode those coefficients which are insignificant till the last bit-plane, have a preferred neighborhood and do not form a run of insignificant samples. The coefficient under consideration i.e. $s_i\left(k_1,k_2\right)$ is encoded using the context of its neighbors in terms of significance. The significance of the neighbors of $s_i\left(k_1,k_2\right)$ are grouped into three categories:
  - **Horizontal:** Given by $h_i\left(k_1,k_2\right)=\sum_{z\in\{-1,1\}}\sigma_1\left(k_1+z,k_2\right)$, so that $0\le h_i\left(k_1,k_2\right)\le 2$.
  - **Vertical:** Given by $v_i\left(k_1,k_2\right)=\sum_{z\in\{-1,1\}}\sigma_1\left(k_1,k_2+z\right)$, so that $0\le v_i\left(k_1,k_2\right)\le 2$.

- **Diagonal:** Given by $d_i(k_1,k_2)=\sum_{z_1,z_2\in\{-1,1\}}\sigma_1(k_1+z_1,k_2+z_2)$, so that $0\le d_i(k_1,k_2)\le 4$

The label associated with the ZC primitive depends on the values of $h_i(k_1,k_2)$, $v_i(k_1,k_2)$ and $d_i(k_1,k_2)$. These are quantized to nine distinct coding contexts.

- **Run-length Coding (RLC):** This primitive is used in place of the ZC primitive, when each of the following conditions hold good:
  - Four consecutive samples are all insignificant, i.e. $\sigma_i(k_1+z,k_2)=0 \quad \text{for } 0\le z\le 3$
  - All these samples have insignificant neighbors, i.e. $h_i(k_1+z,k_2)=v_i(k_1+z,k_2)=d_i(k_1+z,k_2) \quad \text{for } 0\le z\le 3$
  - The samples must reside within the same code block.
  - Horizontal index of the first of these four samples, k_1 must be even.

  When a group of four samples satisfy the above conditions, a single symbol is used to identify whether any of the four samples become significant in the current bit plane.

- **Sign Coding (SC):** This primitive is used only once for each sample, when a previously insignificant sample is found to be significant during a ZC or RLC operation. It is observed that the sign bits $\chi_i(k_1,k_2)$ of adjacent samples tend to be correlated and that is why, the label assigned to the SC primitive takes into consideration the contexts of $h_i(k_1,k_2)$, $v_i(k_1,k_2)$ and $\chi_i(k_1,k_2)$.

- **Magnitude Refinement (MR):** This primitive is used to encode the subband samples in a bit-plane, which are already significant from the previous pass. A new state variable $\tilde{\sigma}_i(k_1,k_2)$ is introduced which makes a transition from 0 to 1 after the MR primitive is first applied to $s_i(k_1,k_2)$. The bit $\upsilon_i^p(k_1,k_2)$ is coded with contexts that depend upon $h_i(k_1,k_2)$ and $v_i(k_1,k_2)$.

15.6 Formation of Quality Layers

We now focus our attention to the second stage of the two-tier coding structure shown in fig.15.4, which accepts the embedded bit-stream and other summary information from each of the code blocks to form the quality layers. As discussed

in section-15.4, each code block B_i has a collection of truncation points $\{n_i^q, \quad q = 1,2,\cdots\cdots\}$ corresponding to the different quality layers. The embedded bit stream for each code block, generated from the first stage of the coding engine is composed of a collection of quality layers Q_q, where $q = 1,2,\cdots\cdots$ are the indices of the quality layers in the increasing order of quality. The first $R_i^{n_i^q}$ bytes in the embedded bit-stream of B_i is composed of the quality layers Q_1 to Q_q, in which the layer *q* makes an incremental contribution $L_i^q = R_i^{n_i^q} - R_i^{n_i^{q-1}} \geq 0$ from the code block B_i. The code blocks may make empty contributions to some of the quality layers. For each code block, following quantities are sent as summary information: (a) the index of the quality layer Q_q to which the code block makes the first nonempty contribution, (b) the incremental contribution L_i^q to each quality layer and (c) the value of $p_i^{\max}$. These quantities are available for all the code blocks from the first stage. Two quantities show significant amount of inter code block redundancies. These are $p_i^{\max}$ and the index q_i of the quality layer to which the code block B_i makes its first contribution. These inter code block redundancies are exploited by using a separate embedded quad-tree code within each subband.

The second-stage of the coding engine, utilize the summary information from the code blocks rather than each sample to compose the quality layers. The quality layer formation is illustrated in fig.15.7. The bit-stream from each code block, truncated into specific truncation points corresponding to the quality layers are arranged in increasing order of *q*. To compose the bit-stream corresponding to each quality layer, we have to proceed along the code blocks in their scanning order and add their contributions to the quality layer.

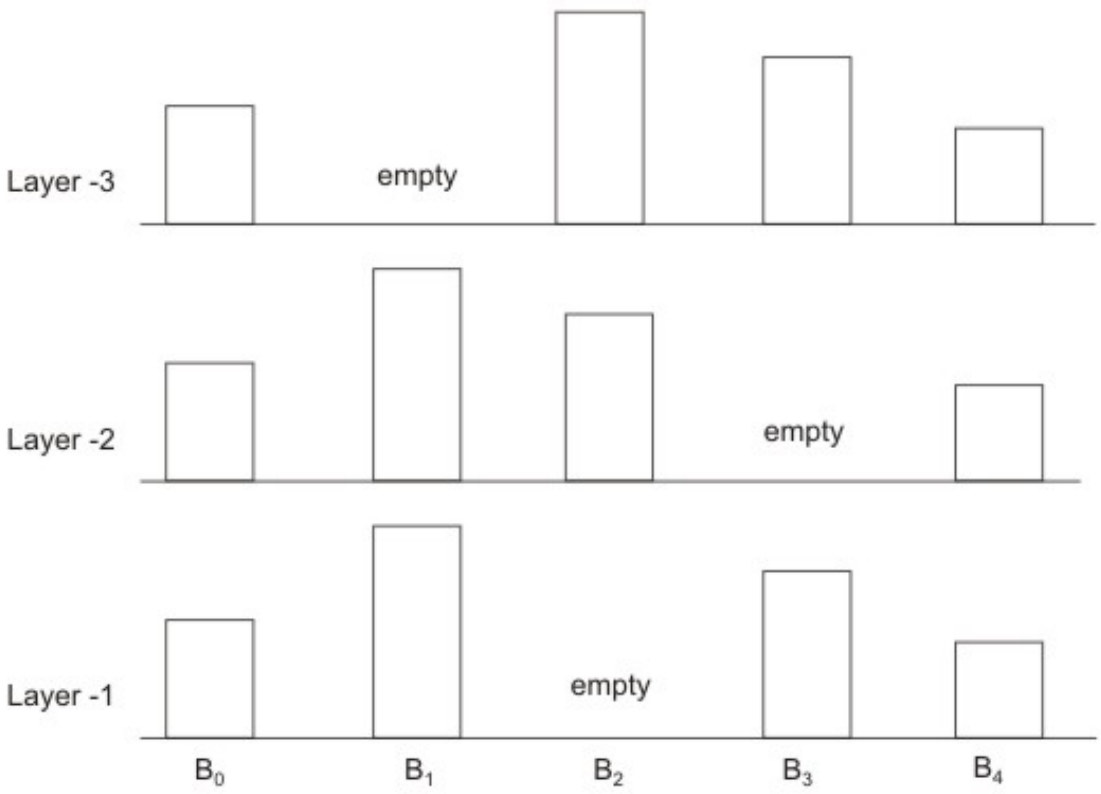

Fig . 15.7 Quality layer formation from the code blocks

15.7 Summary and Conclusions

In this lesson, we have presented EBCOT algorithm, which offers resolution scalability, and SNR scalability within finely embedded bit-streams for relatively small blocks of subband samples. The EBCOT algorithm forms the basis for the latest still image compression standard, JPEG-2000. We shall discuss about the deviations of the JPEG-2000 algorithm from the original EBCOT in lesson-17.

Source: http://nptel.ac.in/courses/Webcourse-contents/IIT%20Kharagpur/Multimedia%20Processing/pdf/ssg_m5l15.pdf