Module 5

EMBEDDED WAVELET CODING

Lesson 14
SPIHT algorithm

Instructional Objectives

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

1. State the limitations of EZW algorithm.
2. Justify the need for magnitude ordering of coefficients in progressive image transmission.
3. Justify the need for efficient encoding of coefficient sorting in progressive image transmission.
4. State the basic objectives of Set Partitioning in Hierarchical Trees (SPIHT) algorithm.
5. Define spatial orientation trees.
6. Define the set partitioning rules.
7. Outline the steps of SPIHT encoding and decoding algorithm.
8. Implement a wavelet coder and decoder based on SPIHT.

14.0 Introduction

In lesson-13, we have studied a fast and efficient approach for encoding wavelet coefficients using Embedded Zerotree Wavelet (EZW) algorithm. The algorithm has two major strengths. First, the bitstream is embedded and the coefficients are ordered in significance and precision, so that it can be truncated according to the bit-rate requirements of the channel. Second, it efficiently utilizes the self-similarity between the subbands of similar orientation and achieves significant data reduction. However, we have pointed out that the EZW algorithm is not exactly optimal and some adjustments of parameters (like, initial threshold) may be necessary to make it optimal with respect to a target bit rate. A single embedded file is unable to give the best performance at all target bit rates. Secondly, EZW has some limitations in efficiently encoding the insignificant coefficients. Although it exploits the self-similarity of coefficients across subbands of same orientation, it does not make any grouping of insignificant coefficients to improve the coding efficiency.

In this lesson, we are going to study a modified form of embedded coding of wavelet coefficients that carries the major strengths of EZW, namely ordered coefficient transmission and self-similarity across subbands of similar orientation. In addition, it partitions the set of coefficients into subsets of insignificant coefficients and identifies each significant coefficient. The approach, proposed by Said and Pearlman is known as Set Partitioning in Hierarchical Trees (SPIHT). At identical target bit rates, experimentations have shown that SPIHT algorithm has improved performance over EZW, because of its ability to exploit the grouping of

insignificant coefficients. This lesson will first provide some of the basics of progressive image transmission and then introduce the concepts of set partitioning. The encoding algorithm will be explained and the students should be able to design and implement a SPIHT coder and decoder.

14,1 Coefficient ordering in progressive image transmission

The hierarchical subband transformation (Wavelet, QMF etc.) may be expressed in a general form as

$$\mathbf{c} = \Omega(\mathbf{s}) \quad \text{.....(14.1)}$$

where, s is the original image array, c is the transformed coefficient array and Ω is the unitary hierarchical subband transformation. Both the original image array and the coefficient array have the same dimensions. The encoder transmits the coefficients and the decoder updates itself according to the received bit-stream. From the approximated coefficient array $\hat{\mathbf{c}}$, it is possible to reconstruct an approximated form of image $\hat{p}$ through the inverse transformation, given by

$$\hat{s} = \Omega^{-1}(\hat{c}) \quad \text{.....(14.2)}$$

The mean-squared reconstruction error of the image at the decoder is given by

$$D_{mse}(s-\hat{s}) = \frac{\|s-\hat{s}\|^2}{N} = \frac{1}{N}\sum_{n_1}\sum_{n_2}\left(s_{n_1,n_2} - \hat{s}_{n_1,n_2}\right)^2 \quad \text{.....(14.3)}$$

where, *N* is the total number of pixels and s_{n_1,n_2} is the pixel intensity at the location (n_1, n_2).

The subband transformation being lossless, the mean square error will be invariant to the transformation, that is,

$$D_{mse}(s-\hat{s}) = D_{mse}(c-\hat{c}) = \frac{1}{N}\sum_{n_1}\sum_{n_2}\left(c_{n_1,n_2} - \hat{c}_{n_1,n_2}\right)^2 \quad \text{.....(14.4)}$$

where, c_{n_1,n_2} is the transform coefficient at the location (n_1, n_2). To start with, the decoder sets $\hat{c}_{n_1,n_2} = 0$ for all coefficients and if the encoder sends the exact value of the coefficient c_{n_1,n_2}, the mean-square error, given by equation (14.4) decreases by $\left(c_{n_1,n_2}\right)^2 / N$. This implies that in an embedded bit-stream, the large-valued coefficients should be transmitted first as they contribute more to the reduction of mean-square error and better reconstruction quality. This justifies why in an embedded bit-stream generation, the transform coefficients should be

ordered according to the magnitude. We had already seen that the EZW algorithm follows this by ordering the significant coefficients in the subordinate pass.

The idea of coefficient ordering can be extended to bit-planes if the coefficients are ranked according to their binary representations and the most significant bits are transmitted first.

Suppose, we first rank the coefficients in decreasing order of the minimum number of bits required for its magnitude representation, that is, the ordering is done such that

$$\left|\log_2\left|c_\eta(k)\right|\right| \geq \left|\log_2\left|c_\eta(k+1)\right|\right| \quad k=1,2,\cdots\cdots,N \quad \text{.....(14.5)}$$

where, $c_\eta(k)$ represents the coefficients in the ordered space.

As an example, let us consider the following array of coefficients before ordering: -3, -9, 16, 5, -57, 8, 38, 2, -12, 14, -17, -6, 25, -7. An arrangement of ordered coefficients, which obey equation (14.5), is shown in

Table-14.1.

Coefficient magnitude	57	38	25	16	17	14	12	9	8	7	6	5	3	2
Sign-bit	1	0	0	0	1	0	1	1	0	1	1	0	1	0
Bit-5 (msb)	1	1	0	0	0	0	0	0	0	0	0	0	0	0
Bit-4	1	0	1	1	1	0	0	0	0	0	0	0	0	0
Bit-3	1	0	1	0	0	1	1	1	1	0	0	0	0	0
Bit-2	0	1	0	0	0	1	1	0	0	1	1	1	0	0
Bit-1	0	1	0	0	0	1	0	0	0	1	1	0	1	1
Bit-0 (lsb)	1	0	1	0	1	0	0	1	0	1	0	1	1	0

Table-14.1 Ordered coefficients according to the bit-plane requirements.

In Table-14.1, we have shown the signed-magnitude representation of the ordered coefficients along the columns. It may be noted that although the ordered coefficients satisfy equation (14.5), it is not a strict magnitude ordering. For example, the coefficient with magnitude 16 appears before the one with magnitude 17. Both of these require 5-bits for representation. To encode these coefficients, the bit-stream should carry the ordering information of the coefficients (i.e., their coordinates) and also the number μ_n which corresponds to the number of coefficients satisfying the inequality $2^n \leq \left|c_{n_1,n_2}\right| < 2^{n+1}$. In the given example, we have $\mu_5 = 2, \mu_4 = 3, \mu_3 = 4, \mu_2 = 3, \mu_1 = 2$. While transmitting the coefficients, the leading 0s and the first 1s in each column need not be sent, as

these are implied with the μ_n information. The next set of bits, marked in blue are to be transmitted sequentially row-wise, from left to right.

Thus, progressive transmission requires one sorting pass, where the coefficients are ordered according to the inequality $2^n \le |c_{n_1,n_2}| < 2^{n+1}$, followed by the refinement pass of transmitting the bits in order of their significance. Although the method fulfils the objectives of embedded bit-stream generation, the encoder requires large number of bits to be transmitted as ordering information and this call for better and efficient methods for encoding the ordering information.

14.2 Basic Objectives of Set Partitioning

In set partitioning approach, the ordering information is not be explicitly transmitted. Instead, the encoder and the decoder follow the same execution path and if the decoder receives the results of magnitude comparisons from the encoder, it can recover the ordering information from the execution path.
In set partitioning, no explicit sorting of coefficients is done. Instead, for a given value of *n*, the coefficients are examined if they fall within $2^n \le |c_{n_1,n_2}| < 2^{n+1}$. If $|c_{n_1,n_2}| > 2^n$, it is significant. Otherwise, it is insignificant. A subset T_m of coefficients are examined to determine if

$$\max_{n_1,n_2 \in T_m} |c_{n_1,n_2}| \ge 2^n \quad \text{.....(14.6)}$$

If this condition is not satisfied, the subset T_m is insignificant and if the condition is satisfied, then the subset T_m is further partitioned to determine insignificant and significant subsets. The significant subsets are repetitively partitioned till single significant coefficients are identified.

The set partitioning rule is designed to work in the subband hierarchy. The objective of the set partitioning algorithm should be such that the subsets expected to be insignificant contain larger number of elements and the subsets expected to be significant should contain only one element.

Before presenting the set partitioning rule, we define the spatial orientation tree, which includes the hierarchy.

14.3 Spatial Orientation Tree

The spatial orientation tree, illustrated in fig.14.1 defines the spatial relationship between the subbands in the form of a pyramid composed of a recursive four-band split. Each node of the tree corresponds to a pixel and is identified by its pixel coordinate. Other than the leaves, each node of the tree has four offspring corresponding to the pixel at the same position in the next finer level of the

pyramid of same orientation, as shown by arrows in the diagram. The only exceptional case is the LL subband existing at the highest level of pyramid. Pixels in this subband form the root and groups of adjacent 2x2 pixels are composed. Other than one of the pixels (marked as '*') out of these four, all remaining three pixels have their four offspring in the HL, LH and HH subbands of the same scale, as shown. One out of the four is obviously left out since only three subbands exist for determining the descendants.

Fig.14.1 Spatial Orientation Tree

The spatial orientation tree discussed as above has close resemblance with the hierarchical tree structure that was used to examine the zerotrees and the zerotree roots in EZW algorithm. However, there is a major difference too. In the hierarchical tree of EZW, every LL subband pixel at the highest level has three offspring at the HL, LH and HH subbands, whereas the offspring relationship of LL subband pixels in the spatial orientation tree is as what has been already discussed in the last paragraph.

14.4 Set Partitioning Rules

We first define the following sets before presenting the set partition rules:

- $O(n_1, n_2)$: The offspring set. It contains the coordinates of the pixels, which are offspring of the node (n_1, n_2). Click once on fig.14.1. One of the pixels starts blinking. It is this node's offspring which we are going to determine. Click on the figure once more. The four pixels which blink together are the offspring of the node and all of them belong to the offspring set.

- $D(n_1, n_2)$: The descendants' set. It contains all the coordinates of the pixels which are descendants of the node (n_1, n_2). Click once more on fig.14.1. All the pixels which are descendants of the node (n_1, n_2) start blinking. All these belong to the descendant set.
- $L(n_1, n_2)$: It is the difference set of $D(n_1, n_2)$ and $O(n_1, n_2)$. It therefore contains the descendants of the node (n_1, n_2), other than the offspring. Click on fig.14.1 once more. Now only the elements of the set $L(n_1, n_2)$ start blinking.

- *H*: This set includes the coordinates of all spatial orientation tree roots, which belong to the highest level of pyramid, that is, the LL subband.

Based on the above set definitions, the set partitioning rules are presented below:

Rule-1: The initial partition contains $D(n_1, n_2)$ for each $(n_1, n_2) \in H$.

Rule-2: If $D(n_1, n_2)$ is found significant, partition it into $L(n_1, n_2)$ plus four single element sets with $(i, j) \in O(n_1, n_2)$.

Rule-3: If $L(n_1, n_2)$ is found significant, partition it into four sets of $D(i, j)$, where $(i, j) \in O(n_1, n_2)$.

14.5 SPIHT Encoding and Decoding

The SPIHT algorithm applies the set partitioning rules, as defined above on the subband coefficients. The algorithm is identical for both encoder and decoder and no explicit transmission of ordering information, as needed in other progressive transmission algorithms for embedded coding, are necessary. This makes the algorithm more coding efficient as compared to its predecessors. Both the encoder and decoder maintain and continuously update the following three lists, viz.

- List of Insignificant Pixels (LIP)
- List of Significant Pixels (LSP)
- List of Insignificant Sets (LIS)

In all lists, each entry is identified by a coordinate (n_1, n_2). In LIP and LSP, the entry represents individual pixels, whereas in LIS, the entry represents either the set $D(n_1, n_2)$ or the set $L(n_1, n_2)$.

As an initialization step, the number (n) of magnitude refinement passes that will be necessary is determined from the maximum magnitude of the coefficients. Initially, all pixels are treated as insignificant. The initialization is followed by three major passes – the sorting pass, the magnitude refinement pass and the quantization step update pass which are iteratively repeated in this order till the least significant refinement bits are transmitted. During the sorting pass, the pixels in the LIP, which were insignificant till the previous pass, are tested and those that become significant are moved to the LSP. Similarly, the sets in LIS are examined in order for significance and those which are found to be significant are removed from the list and partitioned. The new subsets with more than one element are added to the LIS and the single pixels are added to LIP or the LSP, depending upon their significance. During the magnitude refinement pass, the pixels in the LSP are encoded for n^{th} most significant bit. The encoding algorithm can be summarized as follows:

Step-1: Initialization:

Output $n = \left\lfloor \log_2 \left(\max_{(n_1, n_2)} \left\{ \left| c_{n_1, n_2} \right| \right\} \right) \right\rfloor$

Set the LSP = $\{\varnothing\}$

Set the LIP = $\{(n_1, n_2) \in H\}$ and LIS= $\{D(n_1, n_2), \quad (n_1, n_2) \in H\}$

Step-2: Sorting pass:

Step-2.1: For each entry in the LIP, output the significance ("1" if significant, "0" if not significant). If found significant, remove it from the LIP and add to the LSP.

Step-2.2: For each entry in the LIS, output the significance. If found significant, output its sign. Perform the set partitioning using the rule-2 or rule-3, depending upon whether it is the $D(n_1, n_2)$ set or the $L(n_1, n_2)$ set. According to the significance, update the LIS, LIP and LSP.

Step-3: Refinement pass:

For each entry in the LSP, except those which are added during the sorting pass with the same *n*, output the n^{th} most significant bit.

Step-4: Quantization-step update pass:

In this pass, *n* is decremented by 1 and the steps-2, 3 and 4 are repeated until *n* = 0.

The decoder steps are exactly identical. Only the output from the encoder will be replaced by the input to the decoder. As with any other coding method, the efficiency of the algorithm can be improved by entropy coding its output, but at the expense of very high coding / decoding time.

We can explain this algorithm, by considering an example 8x8 array, which is converted into two levels of subband decomposition and the coefficient array is shown in fig.14.2.

First, we perform the initialization on this array. The maximum magnitude of the coefficient is 63 and thus, $n = \lfloor \log_2 63 \rfloor = 5$. This value will be send as an output from the encoder. The initial LIP will contain the entire LL subband coefficients' coordinates and are given by $LIP = \{(0,0), (0,1), (1,0), (1,1)\}$

The initial LIS will contain the descendant subsets of LL subband coefficients, i.e., $LIS = \{D(0,1), D(1,0), D(1,1)\}$

Note that this list does not include $D(0,0)$ as the node $(0,0)$ does not have any descendants. Also, the initial LSP list is empty, that is, $LSP = \{\varnothing\}$.

Now, let us follow the first sorting pass. The coefficients will be examined from the LIP and the first element to consider is (0, 0). This is significant, since,

$2^5 \le |c(0,0)| < 2^6$. The significance (a bit value of "1") and its corresponding sign (in this case, "0", since it is positive) will be sent as output. Now, we move the coordinate entry (0,0) from the LIP to the LSP. The other LIP coefficients will be examined in the order. The students may please verify that the remaining significance bit outputs will be: $\{1,0,0\}$. Also, the sign of the other significant coefficient will be output as "1" (since $c(0,1) = -34$, i.e. negative). At the end of this LIP sorting, the status of the LIP and the LIS will be

$LIP = \{(1,0),(1,1)\}$
$LSP = \{(0,0),(0,1)\}$

Next, we follow the LIS sorting. First $D(0,1)$ is examined. This is significant and hence this significance status (=1) will be first sent as output. The set $D(0,1)$ needs partitioning. So, we examine the offspring of $D(0,1)$, which are (0,2), (0,3), (1,2) and (1,3). We output their significance and sign (if significant) as before. The coefficients (0,2) is significant. So, it is added to the LSP. All remaining ones are insignificant and added to the LIP. Further, the set $L(0,1)$ is added at the end of the LIS and $D(0,1)$ is removed.

The student may now carry out these steps for the other entries in the LIS. The refinement pass only considers the entries in the LSP and their encoding for the current value of *n* is very straightforward.