International Journal of Research and Development in Applied Science and Engineering (IJRDASE)
ISSN: 2454-6844

Improved Image Compression by Set Partitioning Block Coding by Modifying SPIHT

Kirti Saraswat
Electronics & Communication
U.P.T.U., Lucknow (U.P.)
ksaraswat1503@gmail.com

Imran ullah khan
Associate Prof., Electronics & Communication Dept
Integral University, Lucknow (U.P.)
imranuk79@gmail.com

Abstract-- The proposed image compression method present a different implementation, based on set partitioning in hierarchical trees (SPIHT), which provides even better performance than already reported extension of the wavelet based compressions that surpassed the performance of the conventional methods like DCT and run length like codlings . The image compression files results are calculated by original images for compression ratio and with reconstructed by the decompression methods for PSNR. The results are found either comparable to or surpass previous results obtained by other computationally complex methods. In addition of this the developed compression and decompression methods are very fast, and the performance can be made faster, with only small loss in performance, by reducing the bitrates values.

Keywords--Image Compression, SPIHT, Wavelet, DSP, Quantization

1. Introduction:

Compression is the technique of reducing the number of bits needed to store or transmit information. Compression technique involve encoding information using fewer bits than original information content. Compression can be either lossy or with loss. The Lossless compression technique reduces bits by identifying and removing redundancy. In the lossless compression technique no information content is lost. However lossy compression technique reduces bits by identifying unnecessary information and eliminating it. This process of reducing the size of a data file is popularly known as data compression or coding, but its popular name is source coding. Coding or compression is very useful because it helps to reduce usage of resource, viz. data storage space or transmission capability [1-3]. Almost all data compression algorithms consist of at least one model and a code associated with it. In which model estimates the probability distribution and coder assigns shorter codes to the more probable symbols. Compression or coding is very useful in most situations because the compressed data will save time and the space if it is compared to the un encoded information in it. There are various types of coding techniques available for compression of information. Out of them here using arithmetic coding technique because it will create code word for the entire data altogether, rather than creating code word for each data word.

Arithmetic coding technique is a well-known method for lossless data compression or source coding. The Arithmetic coding is mostly popular in image and video compression and its applications. That is if we have a message composed of symbols over finite alphabets, we can create the exact number of bits that corresponds to that symbol. The arithmetic coding bypasses the idea of replacing an input symbol with its specific code. Also it replaces a stream of input symbols with a single floating point number. This arithmetic coding offers the opportunity to create a code that exactly represents the frequency of any character . This coding technique is the most efficient method to code symbols according to their occurrence probability. Unlike a binary Huffman code tree the arithmetic coding offers a better compression rate clearly, as it generates a single symbol rather than several different code words. Arithmetic coding involves a message that is encoded as real number in an interval from zero to one [4]. Arithmetic encoder generates a unique identifier for the sequence of length 'm' to be coded and will assign a unique binary code to this sequence. Since main data structure of arithmetic coding is an interval which represents the string constructed so far as its initial value is either 0 or 1. At each and every stage, the current interval [min, max] is subdivided into sub-intervals corresponding to the probability model for the next given character. Such interval chosen will be the one representing the next character. The more frequently occuring the character, the larger the interval assigned. The output of the coder is a number in the last interval. Here we try to design an arithmetic coder which is to be used in Set Partitioning of Hierarchical Trees (SPIHT) encoder for further compression of the discrete wavelet based decomposed images as per requirement. After this SPIHT transformation some regularities will present in the file. These regularities can allow us for further file compression. Having this in mind we investigated the addition of arithmetic compression to an image which is SPIHT encoded [5].

The SPIHT algorithm is one of the powerful algorithm present for the compression. One can use Wavelet transform SPIHT algorithm to encode the coefficients of wavelet. This SPIHT sorting involves comparison of two elements at a time which results in yes/no condition. According to this sorting pass coefficients are categorizes into 3 lists: LIS (List of Insignificant Sets) are the set of coefficients having their magnitude smaller than the threshold taken .And LIP (List of Insignificant Pixels) are the pixels having magnitude smaller than the threshold considered. The other one is LSP (List of Significant Pixels) which are the pixels whose magnitude is greater than that of considered threshold. This coding method can obtain optimal performance for its ability to generate codes with fractional bits and it is widely used in various image compression techniques. More

importantly the set partitioning in hierarchical trees uses an Arithmetic coding method to improve its peak signal to noise ratio value. Here we try to design an arithmetic coder for SPIHT to increase the performance and further result in good compression of the image [9].

2. Spatial Orientation Trees:

Most of an image's energy is concentrated in its low frequency components. Consequently, the variation decreases as we move from the highest to the lowest levels of the sub band pyramidal structure. Also, it has been observed that there is a spatial self-similarity between existing sub bands, and their coefficients are expected to be better magnitude if we move downward in the pyramid following the same orientation in space. For an instance, large low-activity areas are supposed to be identified in the highest levels of the pyramidal structure, and then they are replicated in the lower levels at the same spatial locations [6].

Tree structure, called spatial orientation tree, naturally defines the spatial relationship existing on the hierarchical pyramid. Fig. 1 shows clearly how our spatial orientation tree is defined in a pyramid constructed with recursive splitting of four-subband. Every node of the tree corresponds to a pixel, and it is identified by the pixel coordinate. Its just direct descendants (offspring) correspond to the pixels of the same spatial orientation in the next finer level of the considered pyramid. The tree that defined in such a way that each node has either no offspring (the leaves) or have four offsprings, which always form a group of 2X2 neighbouring pixels. In the given Fig. 1 the arrows are oriented from the parent node to its four obtained offspring. The pixels that are in the highest level of the pyramid are the tree roots and are also grouped in 2X2 adjacent pixels as shown. However, their relative offspring branching rule is different, and hence in each group one of them (indicated by the star in Fig. 1) has no descendants. The sets of coordinates are used to present the new coding method are as follows:

- O(i;j): set of coordinates of all the offspring node (i; j);
- D(i; j): set of coordinates of all the descendants of nodes (i; j);
- H: set of coordinates of all the spatial orientation tree roots (nodes in the highest pyramid level);
- L(i,j) = D(i, j) - O(i, j)

For instance, except at the highest and the lowest pyramid levels, we have

O(i,j) = {(2i.2j) , (2i . 2j+10 , (2i + 1 , 2j) , (2i +1 , 2j +1)}

We use some parts of the spatial orientation trees as the partitioning subsets in the sorting algo. The set partitioning rules are simply as given below:

1. The initial partition is performed with the sets {(i; j)} and D(i; j), for all (i; j) Є H;
2. if D(i; j) is significant then it is again partitioned into L(i; j) plus the four single-element sets with (k; l) 2 O(i; j).
3. if L(i; j) is significant then it is again checked and partitioned into the four sets D(k; l), with (k; l) Є O(i; j).

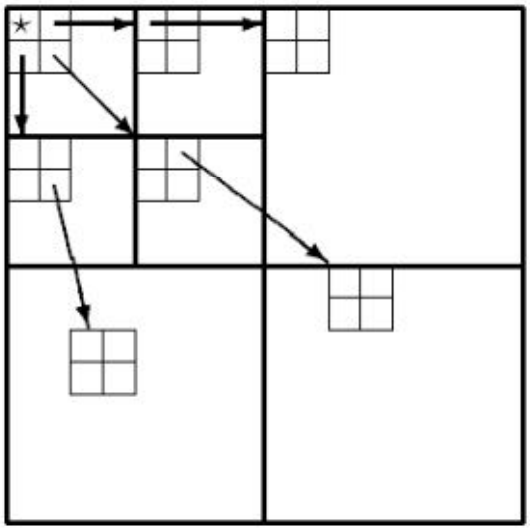

Fig 1: Examples of parent-offspring dependencies in the spatial orientation tree.

3 Coding Algorithm:

The order in which the subsets are checked for significance is very important, in every practical implementation the significance information is stored in three ordered lists, generally known as the list of insignificant sets (LIS), list of insignificant pixels (LIP), and the last is list of significant pixels (LSP). In all these lists each entry is recognised by its coordinate (i; j), which is in the LIP and LSP individual pixels, and for the LIS it is represented either as the set D(i; j) or as L(i; j). To differentiate between them we say that a LIS entry is of type A if it is represented as D(i; j), and is of type B if it can be represented as L(i; j).

While the sorting pass (see Algorithm I) the pixels present in the LIP which were insignificant in the previous pass are tested accordingly, and those pixels that found significant are moved to the LSP set [7].

Similarly, those sets that are sequentially evaluated following the LIS order, and whenever se found a significant set it is removed from the list and is again partitioned . All the new subsets thus formed with more than one element are added back to LIS, whereas the single-coordinate sets are added to the end of the LIP or at the LSP, depending on whether they are found insignificant or are significant, respectively. So formed LSP contains the coordinates of the pixels that are visited in each refinement pass [8].

Here we propose the new encoding algorithm. Which is essentially equal to Algorithm I, However it uses the set-partitioning approach in each sorting pass.

ALGORITHM :

1. Initialization: The output n = | log2 ($\max_{(i;j)}\{|c_{i;j}|\}$; then set the LSP as an empty list, and add the coordinates (i; j) Є H to LIP; only those with descendants are also added to the LIS, as a type A entries.

2. Shorting Pass:

2.1. for each entry (i; j) in the LIP do the following:

2.1.1.for output Sn(i; j);
2.1.2. if Sn (i; j) = 1 then move (i; j) to LSP and the output sign of ci;j ;
2.2. for each entry (i; j) in the LIS do the following:
2.2.1. if the entry is of type A then
For output Sn(D(i; j));
if Sn (D(i; j)) = 1
for each (k; l) Є O(i; j) do:
The output Sn(k; l);
if Sn(k; l) = 1 then (k; l) is added to the LSP and output the sign of ck;l;
if Sn(k; l) = 0 then (k; l) is added to the end of the LIP;
if L(i; j) 6= ; then (i; j) is moved to the end of the LIS, as an entry of type B, and go to Step 2.2.2; otherwise, remove entry (i; j) from the LIS; In
2.2.2. if the entry is found to be of type B then
output Sn(L(i; j));
And if Sn (L(i; j)) = 1 then
each (k; l) 2 O(i; j) is added to the end of the LIS as an entry of type A;
and (i; j) is removed from the LIS.
3. **Refinement pass:** for every entry (i; j) in LSP, except those that are included in the last sorting pass (i.e., with same n), the n-th most significant bit of $|c_{i;j}|$ is presented as output;
4. **Quantization-step update:** n is decremented by 1 and then go to Step 2.

One of the most important characteristic of the algorithm is that the entries added to the end of the LIS in Step 2.2 are evaluated before that same sorting pass finished. So, whenever we say that for each entry in the LIS" we also mean that those are being added to its end. With II Algorithm the rate can be precisely controlled because the transmitted information consists of single bits. The encoder can also use the property in equation number (4) to estimate the reduction in progressive distortion and stop at some desired distortion level.

Note that in the Algorithm II all branching conditions based on the significance data S_n which can only be calculated with the knowledge of ci;j are given at the output by encoder. Hence, to obtain the desired algorithm at decoder, which only duplicates the execution path of the encoder as it sorts the significant coefficients only, we just simply have to replace the words output by input in II Algorithm .By comparing the algorithm above to the Algorithm I, we can also see that the ordering of information η(k) is recovered when the coordinates of the significant coefficients are added to the end of LSP, i.e, the coefficients pointed by the coordinates in the LSP are sorted according to (5). However note that whenever the decoder inputs the data, the three control lists of decoder (LIS, LIP, and LSP) are identical to the ones used by the encoder at the moment it gives that data at its output, it means that the decoder indeed recovers the ordering from execution path. Now it becomes easy to see that with this scheme coding and decoding both have the same computational complexity.

Another additional task done by decoder is to update the reconstructed image. For every value of n when a coordinate is moved to LSP, and it is known that $2^n \leq |ci;j| < 2^{n+1}$. So,when the decoder uses that information, and the sign bit that is input just after the insertion in the LSP,for $c_{i;j} = \pm 1{:}5X2^n$. Similarly, during the refinement pass the decoder adds or subtracts 2^{n-1} from $c_{i;j}$ when it inputs the bits of the binary representation of value $|c_{i;j}|$. In this way the distortion gradually decreases during both of the sorting and the refinement passes.
As in any other coding method, efficiency of Algorithm can be improved by entropy-coding of its output, but at the cost of a larger coding or decoding time. Practical experiments shows that normally there is little to be gained by entropy-coding the coefficient signs or the bits that are put out during refinement pass. On the other side, the significanct values are not having equal probablity, and hence there exists a statistical dependence between $S_n(i; j)$ and $S_n(D(i; j))$ value, and this also present between the significance of adjacent pixels.
We have exploited this dependence using the adaptive arithmetic coding algorithm Witten et al. [13]. In order to increase the coding efficiency,the groups of 2X2 coordinates were kept together in the mentioned lists, and their significance values were coded as a single symbol by the arithmetic coding algorithm as discussed. As the decoder only needs to know the transition from insignificant to significant (the inverse is impossible)values, the amount of information that is needed to be coded, changes according to a number m of insignificant pixels present in that group, and for each case it can be conveyed by an entropy-coding alphabet with max of 2m symbols. With this arithmetic coding it is straightforward to use several adaptive models, each with 2^m symbols, where m Є {1; 2; 3; 4}, in order to code the information in a group of four pixels.
By coding the significance information together the average bit rate corresponds to a m-th order entropy. At the same time, by using different models for the different number of insignificant pixels, each adaptive model contains probabilities conditioned to the fact that a certain number of adjacent pixels are either significant or are insignificant. This is the way dependence between magnitudes of adjacent pixels is completely exploited. The above scheme was also used to code the significance of trees rooted in groups of pixels os size 2X2.
With the arithmetic entropy-coding it is still possible to produce a coded file with almost the exact code rate, and probably a few unused bits to pad the file to desired size.

4. Result and Discussion:
In this paper we have shown the results that are obtained by our ASPIHT compression method over the coloured images. We have taken five different images and tested our algorithm for different bit rates of our image compression algorithm. The images which we have tested is Figure 4.1(a) “heart.jpg” and the bit rates that we have considered are:
1).0625
2)0.125
3)0.25
4)0.5
5)1
For every time our algorithm generated the reconstructed image along eith the PSNR and CR values. These results are

tabulated and shown in the next sections of this chapter one by one.

4.1 Compression of image "heart.jpg"

The heart.jpg image is resized as image of size 512X512 using the MATLAB command thereafter we have applied the image compression algorithm ASPIHT and reconstructed image is shown at different bit rates.The original image is shown in fig. 2(a) for reference and comparison purpose.The reconstructed image at different bit rates are shown in fig 2(b) to fig.2(f).

Orihnal Image

Fig. 2(a) Original image 'heart.jpg'

reconstructed image at bit rate= 0.0625

Fig. 2(b) Reconstructed Image at bitrate =0.0625.

reconstructed image at bit rate= 0.125

Fig. 2(c) Reconstructed Image at bitrate =0.125.

reconstructed image at bit rate= 0.25

Fig. 2(d) Reconstructed Image at bitrate =0.25.

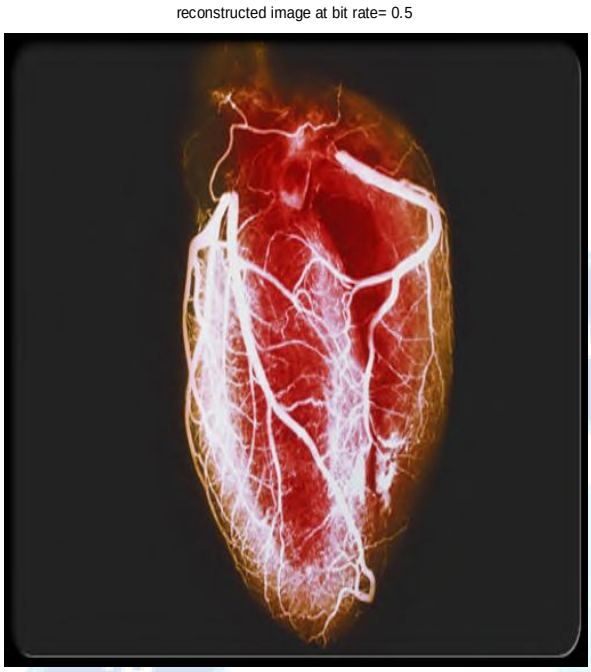

Fig. 2(e) Reconstructed Image at bitrate =0.5.

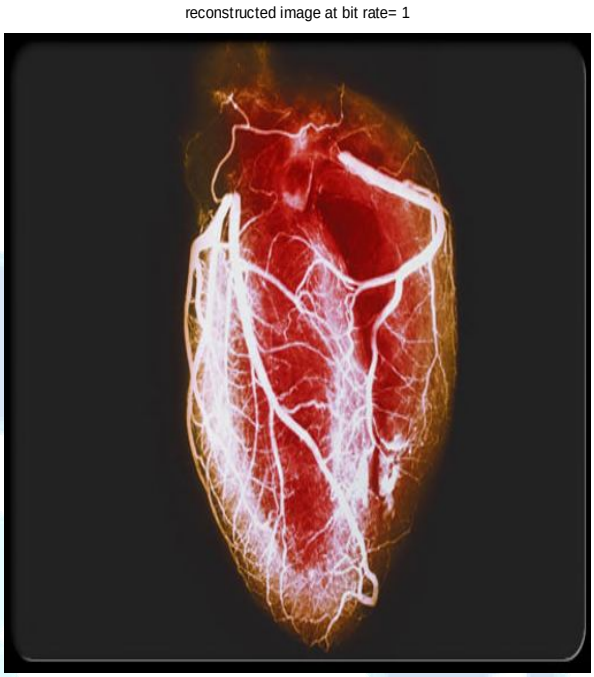

Fig. 2(f) Reconstructed Image at bitrate =1

We can see that for bit rate 0.0625 reconstruction of the image is blurred. However the blurring has been decreased significantly as we move from bit rate 0.0625 to bit rate 1 that is, from fig. 2(b) to fig.2(f). However we observed that the image reconstructed from bit rate 0.25 is very much similar to that of the original image fig, 2(a).To analyze the similarity between the reconstructed image and the original image we have considered two parameters that are PSNR and CR obtained for three different planes viz. R, G, B for all the above mentioned bit rates given in following table:

Table 1. Analytical Results for Image compression for image heart.jpg.

Bit Rate	R	G	B	Average value
.0625	PSNR = 29.0021 CR = 128	PSNR = 26.8883 CR = 127.98	PSNR = 28.4326 CR = 127.99	PSNR = 28.1076 CR = 127.99
.125	PSNR = 31.1587 CR = 64	PSNR = 29.8257 CR = 64	PSNR = 29.4107 CR = 64	PSNR = 30.1317 CR = 64
.25	PSNR = 34.7756 CR = 32	PSNR = 33.8756 CR = 32	PSNR = 33.5586 CR = 32	PSNR = 34.069 CR = 32
.5	PSNR = 40.1346 CR = 16	PSNR = 39.5307 CR = 16	PSNR = 39.2756 CR = 16	PSNR = 39.649 CR = 16
1	PSNR = 48.197 CR = 8	PSNR = 47.6962 CR = 8	PSNR = 47.3008 CR = 8	PSNR = 47.731 CR = 8

However our main objective is to provide high degree of image compression . For justifying this we have also shown the CR for the reconstructed images as shown in fig.4.1.
Since the compression is performed for R, G, B planes so to analyse the result we have calculated the average values of PSNR and Cr. We can see that the average PSNR varies from 28 to 47 and the corresponding CR varies from 8 to 128 times.
However the acceptable PSNR values are nearly 31 to35 and we can see that the image at bit rate 0.25 is giving PSNR of 34.069 with corresponding CR of 32 which is appropriate value for our reconstructed image after compression.

5. Conclusion:

In this work we have demonstrated the application of an advance wavelet transform based image compression technique using SPIHT coding algo. And is tested for several color images belonging to different file format. The performance of our algo. Is tested for different bit rates.
And finally results are tabulated in terms of PSNR and CR for justifying the performance of our proposed compression algo. It has been observed that the best suitable PSNR is in the range of28 to 47 and corresponding CR between 8 to 128. Which is appropriate value to reconstruct the original

image as per the human perception and to retrieve thr max. Necessary information from it.

Hence it can be concluded the our algorithm is capable of reconstructing the color image successfully at different bit rates .And so suitable bit rate can be used to get the desired reconstruction.

In future we can explore performance of our algo. At various decomposition level of the wavelet transform.

There are advanced versions of DWT known as fractal based wavelets and LWT, that can also be applied to enhance the performance of our proposed algorithm.

References:

[1] Keun-hyeong Park and HyunWook Park, "Region-of-Interest Coding Based on Set Partitioning in Hierarchical Trees" IEEE transactions on circuits and systems for video technology, vol. 12, no. 2, February 2002.

[2] William A. Pearlman et. al., "Efficient, Low-Complexity Image Coding with a Set-Partitioning Embedded Block Coder" 2004 - ieeexplore.ieee.org.

[3] Hala H. Zayed et. al., "3D Wavelets with SPIHT Coding for Integral Imaging Compression" IJCSNS International Journal of Computer Science and Network Security, VOL.12 No.1, January 2012.

[4] S. Sridevi et. al., "Medical Image Compression Based On Set Partitioning In Hierarchical Trees Using Quantized Coefficients Of Self Organizing Feature Map For Mr Images" Journal of Theoretical and Applied Information Technology 10th October 2013. Vol. 56 No.1.

[5] Monauwer Alam and Ekram Khan, "listless highly scalable set partitioning in hierarchical
trees coding for transmission of image over heterogenous networks" International Journal of Computer Networking, Wireless and Mobile Communications (IJCNWMC), Vol.2, Issue 3 Sep 2012 36-48

[6] Mahesh Chandra et. al., "A Multiple Description Coding Method Based On Set Partitioning In Hierarchical Tree Algorithm For High Definition Image" International Journal of Electrical and Electronics Engineering Research (IJEEER) Vol. 3, Issue 1, Mar 2013, 55-60.

[7] VenkateshVC et al., "SPHIT Algorithm combined with Variable length encoder to enhance the performance of the image compression technique" International Journal Of Engineering Sciences & Management, 3(2),April-June, 2013.

[8] A. Sreenivasa Murthy et. al., "Performance Analysis Of Set Partitioning In Hierarchical Trees (Spiht) Algorithm For A Family Of Wavelets Used In Color Image Compression" ICTACT Journal On Image And Video Processing, November 2014, Volume: 05, Issue: 02.

[9] Meenu Roy and N.Kirthika, "Design of Coder Architecture for Set Partitioning in Hierarchical Trees Encoder" International Journal of Current Engineering and Technology Vol.4, No.3 (June 2014).