Medical Image Compression Using a Novel Embedded Set Partitioning Significant and Zero Block Coding

Geetha Palanisamy[1] and Annadurai Samukutti[2]
[1]Department of CSE & IT, Anna University, India
[2]Government College of Engineering, India

Abstract: *Lossless compression with progressive transmission is playing a key role in telemedicine applications. The proposed lossless embedded progressive region based significant and zero block coding uses a recursive set partitioning procedure to sort subsets of wavelet coefficients by maximum magnitude with respect to thresholds that are integer powers of two. It exploits two fundamental characteristics of a wavelet transform – the well defined hierarchical structure and energy clustering in transform domain. The proposed low complexity coder is rooted from Set Partitioning Embedded Block (SPECK) and it is a distortion scalable coder which achieves excellent coding performance. Extensive experiments have verified that the proposed coder is competitive with other state of the coders such as set partitioning in hierarchical trees, SPECK, JPEG2000 on compression performance and computational complexity for standard test images.*

Keywords: *ESPSZBC, lossless compression, progressive transmission, SPECK, SPIHT.*

Received November 4, 2006; accepted November 24, 2006

1. Introduction

Clinical Picture Archiving and Communication Systems (PACS), and telemedicine networks require the storage and transmission of a large amount of medical image data, and efficient compression of these data is crucial. Applying Image compression reduces the storage requirements and network traffic, therefore improves efficiency. Several techniques have been proposed for image compression [11]. These can be classified into lossy and lossless techniques. Lossless compression techniques allow exact reconstruction of the original, but the achievable compression ratios are only of the order approximately 2:1. This paper addresses the following combined problem: (1) obtaining high compression ratio for the lossless coding of images, comparable to other methods [7, 5, 6]. (2) Achieving this in an embedded fashion, i.e., in such a way that all lower rate codes of an image are embedded at the beginning of the bitstream generated for the lossless coding. And (3) progressive transmission of the embedded bitstream. Solving this problem is important for many application domains. In medical area one cannot afford information loss due to compact coding, because lossy compression may destroy some of the information required during processing, or add artifacts that lead to erroneous interpretations. So lossless compression has been widely preferred by medical professionals [11], because it facilitates accurate diagnosis.

The standard wavelet compression techniques, even if lossless in principle do not reconstruct exactly the original image because of the rounding of the floating-point wavelet coefficients to integers. The use of the lifting scheme allows generating truly lossless integer-to-integer wavelet transforms. In recent years, new reversible integer wavelet transforms [2, 3, 9] can be developed using lifting scheme, have grown in popularity for lossless image compression. Effective and computationally simple techniques of transform based image coding have been realized using set partitioning and significance testing on hierarchical structures of transformed images. Said and Pearlman introduced such a technique in their Set Partitioning in Hierarchical Trees (SPIHT) algorithm [5]. It is a successful extension and improvement of Shapiro's Embedded Zero tree Wavelet (EZW) algorithm [7]. The SPIHT algorithm has since become a standard benchmark in image compression.

The SPIHT scheme employs an iterative partitioning or splitting of sets or groups of pixels, in which the tested set is divided when the maximum magnitude within it exceeds a certain threshold. When the set passes the test and is hence divided, it is said to be significant. Otherwise it is said to be insignificant. Insignificant sets are repeatedly tested at successively lower thresholds until isolated significant pixels are identified. This procedure sorts sets and pixels by their threshold of significance. The results of these so called significance tests describe the path taken by the coder to code the source samples. Since the binary outcomes

of these tests are put into the bit stream as "1" or "0", the decoder at the destination can duplicate the execution path of the encoder.

The principal of set partitioning and sorting by significance is the key to excellent coding performance with very low computational complexity. This recognition has spawned more algorithms in this category, among which are Amplitude and Group Partitioning (AGP) [4], SWEET [1], NQS [8], Set Partitioning Embedded bloCK (SPECK) [6], and the proposed Embedded Set Partitioning Significant and Zero Block Coding (ESPSZBC).

The ESPSZBC coder contains the following main features:

- It uses integer wavelet transform to decompose the original image I, which provides a multiresolution representation of I.
- The proposed coder posses the capability of progressive transmission and embeddedness.
- It is a fast codec, whose computational complexity is similar to SPECK.
- It generates an embedded distortion scalable bitstream which contains distinct subsets, Bq, such that $\bigcup_{k=0}^{q} B_k$ together represents the samples from all subbands at some reconstruction quality level, q. The embedded bitstream output from ESPSZBC can be transmitted progressively and truncated at any point to get an optimal or suboptimal representation of the original image X. The algorithm runs sequentially and can stop whenever a target bit rate or a target distortion is met.

ESPSZBC is different from SPIHT and EZW in that it does not use trees which span and exploit the similarity across different subbands of wavelet decomposition; rather like AGP, SWEET, and NQS it makes use of sets in the form of blocks of contiguous coefficients within subbands. The ESPSZBC coding scheme has its roots from primarily in the ideas developed in the SPECK [6] coding algorithm. Comparative run-times, as reported in [6], show that SPECK is 4.6 to 15.7 times faster than JPEG2000's VM 3.2A (Version model, 3.2 A), which is essentially the EBCOT coder [10], in encoding and 8.1 to 12.1 faster in decoding on the average over a set of four images and set of four rates, 0.25, 0.50, 1.0 and 2.0 bits per pixel. Meanwhile the reduction of PSNR from that of VM 3.2. A ranges only from a minimum of 0.45 dB for entropy coded versions to a maximum of 0.85 dB for nonentropy coded versions. Arithmetic coding is used to increase the compression ratio of the proposed approach. The rest of paper is organized as follows. Lifting and reversible integer wavelet transform is discussed in section 2 ESPSZBC coding methodology in section 3, ESPSZBC coding algorithm in section 4, experiments and results are presented in section 5, and the paper is concluded in section 6.

2. Lifting and Reversible Integer Wavelet Transforms

In most cases, the wavelet transform produces floating-point coefficients and although this allows perfect reconstruction of the original image in theory, the use of finite-precision arithmetic, together with quantization, results in a lossy scheme. Recently, new Integer wavelet transform, i.e., wavelet transforms that transform integer to integer have been introduced [2, 3, 9]. In [2], it was shown that an integer version of every wavelet transform with finite filters can be obtained using the lifting scheme of [9]. Lifting allows a wavelet transform to be computed quickly through a series of individual lifting steps and fully in place calculation. Integer wavelet transform is used to reduce the memory demands of the compression algorithm as integers are used instead of real numbers.

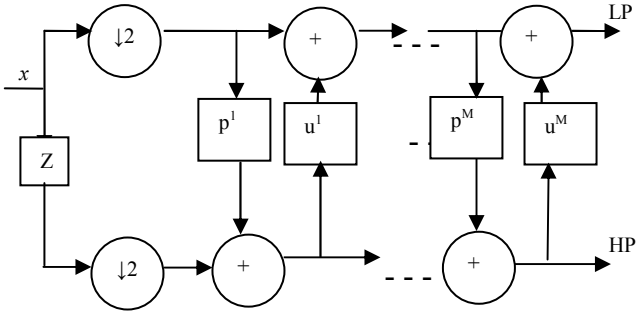


Figure 1. Forward wavelet transform using lifting.

Figure 1 depicts the forward wavelet transform using lifting. For the forward transform, the data set *x* is split into two disjoint subsets. There is no restriction on how the data should be split, or on the relative size of each of the subsets. The only thing we need is some procedure to join the subsets back into the original data set *x*. The easiest possibility is split the data *x* into even and odd samples. Next is a series of dual lifting and primal lifting steps, alternatively referred to as prediction and update steps. At the conclusion of these lifting steps, the even and odd samples have been replaced by

$$d_i[n] = d_{i-1}[n] - \left\lfloor \left(\sum_k p_i[k] s_{i-1}[n-k] \right) + 1/2 \right\rfloor \quad (1)$$

$$s_i[n] = s_{i-1}[n] - \left\lfloor \left(\sum_k u_i[k] d_{i-1}[n-k] \right) + 1/2 \right\rfloor \quad (2)$$

High pass (1) and low pass (2) coefficients respectively. It should be noted that although integers are transformed to integers, the coefficients *p(i)[k]* and *u(i)[k]* are not necessarily integers. Thus, computing the integer transform coefficients requires floating point operations. More detailed information on lifting can be found in [2, 3, 9]. In this paper Daubechies integer wavelet transform (2, 4) was taken (3), (4) and tested with the proposed coding technique. In equations (1), (2), (3), (4) and *x[n]* refers to the

original signal and s[n] refer to the low pass coefficient and, $d[n]$ refer to the high pass coefficients after applying the wavelet transform.

$$d[n] = x[2n+1] - \lfloor 1/2\,(x[2n] + x[2n+2]) + 1/2 \rfloor \quad (3)$$

$$s[n] = x[2n] + \left\lfloor \begin{matrix} 19/64\,(d[n-1] + d[n]) \\ -3/64\,(d[n-2] + d[n+1]) + 1/2 \end{matrix} \right\rfloor \quad (4)$$

The coding methodology of ESPSZBC is explained in the next section.

3. The ESPSZBC Coding Methodology

The ESPSZBC coding scheme has its roots from primarily in the ideas developed in the SPECK [6] coding algorithm. ESPSZBC uses integer wavelet transforms to decompose the original image X into different subbands, which is identical to hierarchical octave band decomposition. K level decomposition results in $3k+1$ subbands. The quad-tree structure and organization of subbands are shown in Figure 2.

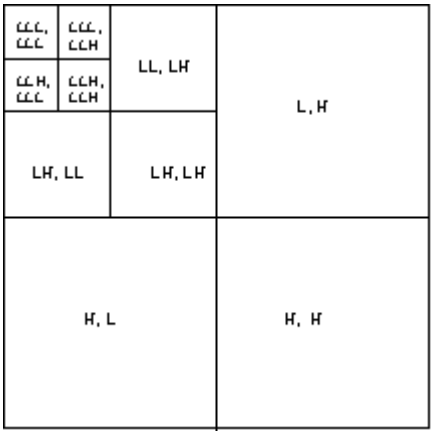


Figure 2. Quad tree structure and subband decomposition.

The transformed image is said to exhibit a hierarchical pyramidal structure defined by the levels of decomposition, with the topmost level being the root (LL subband). The finest pixels lie at the bottom of the pyramid while the coarsest pixels lie at the top (root) level. Image X is represented by an indexed set of transformed coefficients $\{C_{i,j}\}$, located at pixel position (i,j) in the transformed image. The terms coefficients and pixels will be used interchangeably. Pixels are grouped together in sets which comprise regions in the transformed image.

After analyzing the transformed pixels, most of the root level pixels (LL subband) found to be significant and most of the pixels lie at the bottom of the pyramid found to be insignificant. The SPECK coding method efficiently reducing the bits needed to code the insignificant pixels in the finest level of the pyramid. The ESPSZBC coding scheme not only minimizing the bits needed to code the finest level pixels and also it reducing the bits needed to code the significant pixels in the coarsest level. Following the ideas of SPIHT, a set of coefficients is significant with respect to T if

$$Max \left\{ \left| \sum_{(i,j) \in T} c_{i,j} \right| \right\} \geq 2^n \quad (5)$$

Otherwise, it is insignificant. The significance set function is defined as:

$$SG_n(T) = \left\{ \begin{matrix} 1, if\ 2^n \leq max_{(i,j) \in T} |c_{i,j}| \prec 2^{n+1} \\ 0, else \end{matrix} \right\} \quad (6)$$

The ESPSZBC algorithm makes use of rectangular regions of image. The regions or sets, henceforth referred to as sets of type S, can be of varying dimensions. The dimension of a set S depends on the dimension of the original image and the subband level of the pyramidal structure at which the set lie. The size of the set to be the cardinality of the set, i.e., the number of elements in the set

$$Size(S) = C(S) \equiv |S| \quad (7)$$

During the course of the algorithm, set of various sizes will be formed, depending on the characteristics of pixels in the original set. Note that a set of size 1 consists of just one pixel. The other type of sets used in the ESPSZBC algorithm is referred to as sets of type I. These sets are obtained by chopping off a small square region from the top left portion of a larger square region. A typical set I is illustrated in Figure 3. A set is always decomposed into S sets in a prescribed way, so as to progress through the transformed image from coarser to finer resolution subbands. The coding part of ESPSZBC always takes place on the S sets.

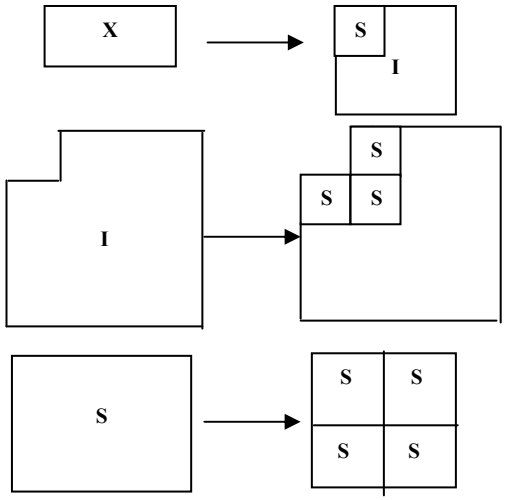


Figure 3. Three rules for partitioning different type sets.

To encode a set S, the ESPSZBC algorithm follows closely the methodology used in the SPIHT algorithm [5]. The difference lies in the sorting pass where instead of using spatial orientation trees for

significance testing, here sets of type *S* is used. The motivation behind this is to exploit the clustering of energy found in transformed images and concentrate on those areas of the set which have high energy. This ensures that pixels with high information content are coded first.

Similar to SPECK, in ESPSZBC two linked lists - List of Insignificant Sets (LIS) and List of Significant Pixels (LSP) are maintained. LIS contains sets of type *S* of varying sizes which have not yet been found significant against threshold *T*. LSP contains pixels which are found to significant against threshold *T*. In SPECK an array of smaller lists of type LIS, each containing sets of type *S* of fixed size, instead of using a single large list having sets *S* of varying sizes. These smaller lists are ordered by size from smallest single pixel sets first to largest set last (bottom).This ordering is the functional equivalent of separating the LIS into two lists, a List of Insignificant Points (LIP) and a List of Insignificant Multipoint Sets (LIS) as done in SPIHT. Because of sorting the sets, processing (subband) order is changed, so partial ordering of the transform coefficients by magnitude is maintained in SPIHT and SPECK algorithm. But in ESPSZBC sorting is not used, the sets are processed in the order stored in LIS, so the progressive (subband) ordering is maintained. The ESPSZBC algorithm is given in the next section.

4. The ESPSZBC Algorithm

Coding methodology is defined in previous section. Outline of the ESPSZBC algorithm is presented in pseudo-code. It consists of four steps: the initialization step, sorting pass, refinement pass, and the quantization step. First an Image *X* that has undergone an appropriate subband transformation. The image *X* consists of transformed coefficients $\{C_{i,j}\}$, located at pixel positions (*i, j*). Such an image exhibits a hierarchical pyramidal structure having subbands at different levels of its decomposition. The topmost band is the root of the pyramid. The algorithm starts by partitioning the image into two sets: set *S* which is root of the pyramid, and set *I* which is everything that is left of the image after taking out the root, as shown in Figure 3. To start the algorithm, set *S* is added to the LIS. The coefficient $C_{i,j} \epsilon X$ is significant if $| C_{i,j} | \geq$ threshold.

The outline of the full coding algorithm is as follows:

1. *Initialization*
   Output $n=\lfloor log2(\max_{(i,j)} |c_{i,j}|) \rfloor$
   Partition image transform X into two sets S≡ root, and R≡ X-S (see Figure 3)
   Add S to LIS and set LSP= Φ
   Set the List of Significant Points (LSP) as empty. Set the roots of similarity trees in the List of Insignificant Sets (LIS).
2. *Sorting pass*
   For each set S ϵ LIS do ProcessS(S).
   If I ≠ Ø, ProcessI.
   Distribute the appropriate indices of the coefficients to the LIS, and LSP.
3. *Refinement pass*
   For each entry in the LSP significant for higher n, send the n^{th} *most significant bit to the decoder.*
4. *Quantization Step*
   Decrement n by one and return to step 2 until the specified bit rate is reached.

Functions used by the ESPSZBC algorithm are given below:

Procedure ProcessS(S)
1. Output $SG_l(\epsilon)nS$
2. If $SG_l(\epsilon)nS$ *=1 then*
  If S is a pixel then
   Output the sign of S and add S to LSP.
   Else CodeS(S)
  If S ϵ LIS , then remove S from LIS
   Else
   If S ϵ LIS, then add S to LIS
3. Return.

Procedure CodeBlockS(S)
1. If S is in LL subband then
  Output SG(S)
  If SG(S)= 1 (Check all the elements in S is significant and positive) then Output SG (S) and all the elements in the set is added To LSP
  Else partition S into four equal subsets O(S)
  (shown in Figure 3)
  For each set $S_i \epsilon O(S)$ *(i = 0, 1, 2, 3)*
    Output $SG_n(S_i)$
    If $SG_n(S_i)$ *=1 then*
      If S_i *is a pixel, output its sign and add* S_i *to LSP*
        Else CodeBlockS (S_i)
    Else add S_i *to LIS*
2. Else (S is in HL or LH or HH)
  Partition S into four equal subsets O(S) (shown in Figure 3)
3. For each set $S_i \epsilon O(S)$ *(i = 0, 1, 2, 3)*
    Output $SG_n(S_i)$
    If $SGn(S_i)$ *=1*
      If S_i *is a pixel, output its sign and add* S_i *to LSP*
      Else CodeBlockS (I_s)
        Else add S_i *to LIS*
4. Return

Procedure ProcessI ( )
1. Output $SG_n(I)$
2. if $SG_n(S_i)$ *=1*

then CodeI ()
3. Return

Procedure CodeI ( )
1. Partition I into four sets – three S_i and one I (Shown in Figure 3)
2. Foreach of the three sets Si (i = 0, 1, 2) ProcessS (S_i)
3. ProcessI( )
4. Return

4.1. Quadtree Partitioning in ESPSZBC

Set *S* in LIS is processed by testing it for significance against the threshold $n=n_{max,}$ not significant it stays in the LIS. If *S* is significant and it belongs to LL subband, then check whether all the elements in the set is significant and positive, then all the elements in the set is moved to LSP. Otherwise it is quadrisected, i.e., partitioned into four subsets, each having size approximately one-fourth the size of the parent set S (function *CodeBlockS*). If *S* is significant and it not belongs to LL subband, it is quadrisected, i.e., partitioned into four subsets. Figure 3 gives an illustration of this partitioning process. In the procedure *CodeBlockS*, each of these offspring sets O(S) is tested for significance for same *n* and, if significant, is quadrisected once more. If no significant, it is added to the LIS.

Each significant subset is, in turn, treated as a set of type S and processed recursively, via *ProcessS* and *CodeBlockS*, until all the sets are reached. The pixel/sets that are found insignificant during this selection process are added to LIS to be tested later against the next lower threshold. The binary result of every significance test is sent to the code bitstream. Whenever some of the elements in the set of size greater than one are significant, tests of four offspring sets follow, so that the binary significance decision paths map onto a quadtree. The motivation for this so-called quadtree partitioning of such sets achieves three goals: (1) to quickly identify the areas of high energy (magnitude) in the set *S* and code them first; (2) to locate structured groups of coefficients that are all above thresholds, so as to minimize the number of bits needed for their representation; and (3) to locate structured groups of coefficients that are below a decreasing sequence of magnitude thresholds, so as to limit the number of bits needed for their representation.

4.2. Octave Band Partitioning in ESPSZBC

At this stage of the algorithm, all current sets of type *S* have been tested against *n*. The set *I* is processed next, by testing it against the same threshold *n* (procedure *ProcessI*). If it is found to be significant, it is partitioned by yet another partitioning scheme-the octave band partitioning. Figure 3 gives an illustration of this partitioning scheme. Set *I* is partitioned into four sets-three sets of type *S* and one of type *I* (procedure *CodeI*). The size of each of these three sets *S* is the same as that of the chopped portion of *X*. The new set *I* that is formed by this partitioning process is now reduced in size.

The idea behind this partitioning scheme is to exploit the hierarchical pyramidal structure of the subband decomposition, where it is more likely that energy is concentrated at the top most levels of the pyramid and as one goes down the pyramid, the energy content decreases gradually. If a set *I* is significant against some threshold *n*, it is more likely that the pixels that cause *I* to be significant lie in the top left regions of *I*. These regions are decomposed into sets of type *S*, and are put next in line for processing. In this way, regions that are likely to contain significant pixels are grouped into relatively smaller sets and processed first, while regions that are likely to contain insignificant pixels are grouped into a large set. A single bit may be enough to code this large region against the particular threshold. Hence, once the set *I* is partitioned by the octave band partitioning method, the three sets *S* are processed in the regular image scanning order, after which the newly formed reduced set *I* is processed.

It should be noted that processing the set *I* is in recursive process, and depending on the characteristics of the image, at some point in the algorithm, the set *I* will cover only the lower most level of the pyramidal structure. When, at this stage, the set *I* tests significant against some threshold, it will be broken down into three sets *S* but there will be no new reduced set *I*. To be precise, the new set *I* will be an empty set. Hence, the functions *ProcessI* and *CodeI* will have no meaning in the ESPSZBC algorithm after this event. This octave band partitioning is done during the initial sorting pass at the highest significance level $n=n_{max}$. Once all the sets have been processed for a particular threshold *n*, the refinement pass is initiated which refines the quantization of the pixels in the LSP, i.e., those pixels that had tested significant during the previous sorting passes. Once this is done, the threshold is lowered and the sequence of sorting and refinement passes is repeated until the desired rate is achieved or, in case of lossless or nearly lossless compression, all the thresholds up to the last, corresponding to n=0, are tested.

4.3. Processing Order of Sets

Once pass 1 is occurred, sets of type S of varying sizes are added to LIS. During the next lower threshold, these are processed sequentially. Because of processing the sets sequentially, the order of importance (first process LL subband then LH, HL, HH) is maintained in ESPSZBC. But in SPECK the sets are processed in increasing order of their size. So in SPECK the order of importance is not maintained.

The decoder uses the same mechanism as the encoder. It receives significance test results from the coded bitstream and builds up the same list structure during the execution of the algorithm. Hence, it is able to follow the same execution path for the significance tests of the different sets, and reconstructs the image progressively as the algorithm proceeds.

5. Experiments and Results

Two of the error metrics used to compare the various image compression techniques are the Mean Square Error (MSE) and the Peak Signal to Noise Ratio (PSNR). The MSE is the cumulative squared error between the compressed and the original image, whereas PSNR is a measure of the peak error. The mathematical formula for the two is:

$$Error\ E = Original\ image - Reconstructed\ Image \quad (8)$$

$$MSE = E / (Size\ of\ the\ image) \quad (9)$$

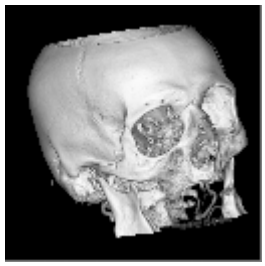
$$PSNR = 20 * \log_{10}\left(255/\sqrt{MSE}\right) \quad (10)$$

The proposed lossless compression algorithm is applied to various images of size 128 × 128 which is digitized to 8 bits/pixel is shown in Figures 4(a), 5(a), 6(a). Part (b) and (c) in Figures 4, 5, and 6 are decomposed and reconstructed images. This algorithm is tested for various images and the Compression Ratio (CR) of SPIHT, SPECK, proposed approach, and proposed approach with arithmetic coding are compared and shown in Table 1. These results show that the proposed algorithm offers excellent CR and also high computational speed.

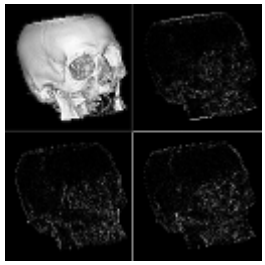
After analyzing the decomposition results of various wavelet transforms, in most of the transforms all the coefficient in the first quadrant are found to be positive. The PSNR ratio of both SPECK and the proposed ESPSZBC are same with a significant increase of compression ratio in ESPSZBC and it is shown in Table1. In this paper Daubechies integer wavelet transform (2, 4) is used. The image is further compressed by using arithmetic coding. If the size of the images is larger the proposed algorithm offers excellent CR.

6. Conclusion

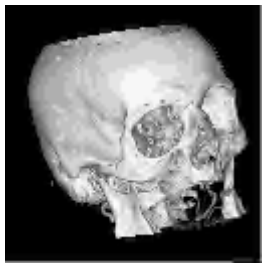
Medical imaging generates vast amount of data. To preserve these data, lossless compression is widely preferred, because it allows the exact reconstruction of the original image. The proposed lossless embedded progressive region based significant and zero block coding uses a recursive set partitioning procedure to sort subsets of wavelet coefficients by maximum magnitude with respect to thresholds that are integer powers of two.



(a) Original Image

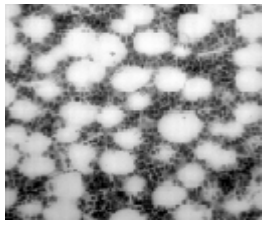


b) Decomposed Image

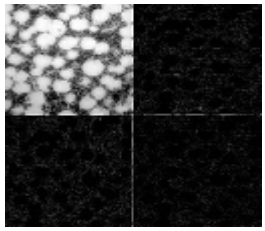


(c) Reconstructed image

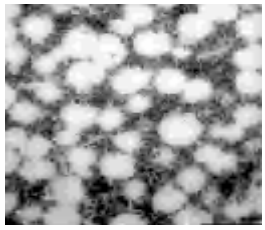
Figure 4. Proposed ESPSZBC compression algorithm of brain image result of (2, 4) transform CR=7.22.



(a)



(b)



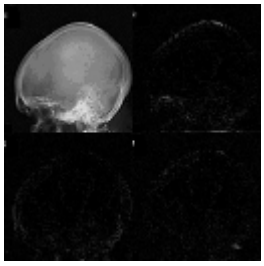
(c)

Figure 5. Proposed ESPSZBC compression algorithm of skull image result of (2, 4) transform CR=9.46.

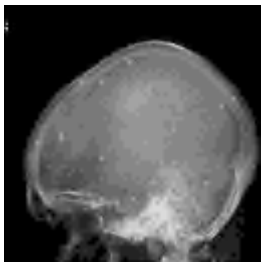
Table 1. Comparison on CR and PSNR of proposed approach with other methods.

Gray scale Images	SPIHT		SPECK		Proposed approach		Proposed approach with arithmetic coding
	CR	PSNR	CR	PSNR	CR	PSNR	CR
Saturn (512 x512)	13.50	(46.96)	17.45	46.97	**28.06**	46.97	**38.17**
Brain (256 x256)	6.95	(42.49)	7.78	42.46	**9.38**	42.46	**12.54**
Mandrill (256 x256)	6.33	(36.58)	6.69	36.54	**8.08**	36.54	**9.78**
Graygoldhill(256 x256)	6.76	(41.67)	7.45	41.65	**8.62**	41.65	**9.98**
Saturn (256 x256)	12.22	(44.92)	15.81	44.88	**20.21**	44.88	**26.43**
Willis (Mono chr256 x256)	9.74	(73.94)	10.51	77.90	**11.85**	77.90	**13.08**
Chest (128x128)	7.28	(40.39)	7.93	40.39	**10.10**	40.39	**12.82**
Skull (128x128)	7.92	(41.79)	8.75	41.79	**11.13**	41.79	**13.79**
Foot wrist (128x128)	6.34	(41.64)	6.91	41.64	**10.03**	41.64	**11.98**
Bonemarr (128x128)	7.20	(38.24)	7.71	38.24	**9.47**	38.24	**10.91**
IC (128X128)	4.67	(40.78)	4.87	40.78	**6.09**	40.78	**7.12**
ORTHO (64 x64)	4.75	(38.78)	4.90	38.78	**5.11**	38.78	**6.80**
Blood cell (64 x64)	7.99	(36.08)	8.31	36.12	**9.01**	36.12	**10.60**

(a)



(b)



(c)

Figure 6. Proposed ESPSZBC compression algorithm of skull image result of (2, 4) transform CR=11.2

It exploits two fundamental characteristics of a wavelet transform the well defined hierarchical structure and energy clustering in transform domain. An excellent compression ratio is achieved and also the quality of the image is found to be good.

References

[1] Andrew J., "A Simple and Efficient Hierarchical Image Coder," *in Proceedings of the IEEE International Conference on Image Processing (ICIP)*, vol. 3, pp. 658-661, 1997.

[2] Calderbank R., Daubechies I., Sweldens W., and Yeo B-L., "Wavelet Transforms that Map Integers to Integers," *Technical Report*, Department of Mathematics, Princeton University, 1996.

[3] Daubechies I. and Sweldens W., "Factoring Wavelet Transforms into Lifting Steps," *Journal of Fourier Analysis and Applications,* vol. 4, no. 3, 1998.

[4] Said A. and Pearlman W., "Low-Complexity Waveform Coding via Alphabet and Sample Set Partitioning in Hierarchical Trees," *in Proceedings of the IEEE Transactions on Circuits and Systems for Video Technology*, vol. 3024, pp. 25-37, 1997.

[5] Said A. and Pearlman W., "A New, Fast and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees," *in Proceedings of the IEEE Transactions on Circuits and Systems for Video Technology*, vol. 6, pp. 243-250, 1996.

[6] Said A. and Pearlman W., "Efficient, Low-Complexity Image Coding with a Set-Partitioning Embedded Block Coder," *in Proceedings of the IEEE Transactions on Circuits and Systems for Video Technology*, vol. 14, no. 11, pp. 1219-1235, 2004.

[7] Shapiro J., "Embedded Image Coding Using Zerotrees of Wavelet Coefficients," *in Proceedings of the IEEE Transactions on Signal Processing*, vol. 41, pp. 3445-3462, 1993.

[8] Spring J., Andrew J., and Chebil F., "Nested Quadratic Splitting," *Document in IEEE Explorer ISO/IEC/JTS1/SC29*, WG1. N1191, July 1999.

[9] Sweldens W., "The Lifting Scheme: A Custom-Design Construction of Biorthogonal Wavelets," *Journal of Applied and Computational Harmonic Analysis*, vol. 3, no. 2, pp. 186-200, 1996.

[10] Taubman D., "High Performance Scalable Image Compression with EBCOT," *in Proceedings of the IEEE Transactions Image Processing*, vol. 9, pp. 1158-1170, 2000.

[11] Wong A., Zaremba L., Gooden D., and Huang H., "Radiologic Image Compression: A Review," *in Proceedings of the IEEE*, vol. 83, pp. 194-219, 1995.

Geetha Palanisamy is a lecturer in Anna University, Chennai. She received her BE degree in computer science and engineering from Anna University and ME degree in computer science and engineering from Government College of Engineering in 2002. She was a research scholar in Computer Science Department, Government College of Technology. Geetha has more than 6 years of experience in teaching and research. Her current research interest includes signal and image processing, processing of moving images, and computer vision.

Annadurai Samukutty is a professor of computer science and engineering at Government College of Technology, Coimbatore. He received his BE degree in electronics and communication engineering and ME degree in power systems in the year 1975 and 1981, respectively. He also received another ME degree in computer science and engineering from Bharathiar University in the year 1990. He received his PhD degree in computer science and engineering from Anna University in the year 1996. He was a design engineer for two years in a leading electrical company prior to joining his teaching profession. Now, he is an eminent professor in the field of computer science and engineering and had more than 25 years of experience in teaching field. His areas of interests include pattern recognition, digital image processing, neural networks, parallel computing, fuzzy systems and computer networks. He is a co-author for the book "Network Programming on Hands". Dr. Samukutty is internationally known for his areas of interest, having published over 60 articles dealing with image compression and segmentation, multitasking, and face recognition. Currently, he is holding the honorary post of Principal in Government College of Engineering, Tirunelveli.