A Efficient Memory No List Set Partitioned Embedded Block (NLSK) Wavelet Image Coding Algorithm for Low Memory Devices

Naimur Rahman Kidwai[1], M. Alam[1], Ekram Khan[2] and Rizwan Beg[3]

[1]*Department of ECE, Integral University, Lucknow, India*
[2]*Department of Electronics Engg. A.M.U, India*
[3]*Department of CSE, Integral University, Lucknow, India*

naimkidwai@gmail.com, malam_iu@rediffmail.com, ekhan67@gmail.com, rizwanbeg@gmail.com

Abstract

In this paper, a fast and memory efficient version of set partitioned embedded block (SPECK) image coder is proposed. Due to use of linked lists, SPECK algorithm, requires large run-time memory, making it unsuitable for memory constrained portable multimedia devices and WMSN's. The proposed coder replace linked list with fixed size static memory, to keep track of set partitions and coding information and use few markers to facilitate coding. Elimination of linked lists also reduces the memory access time and time involved in addition /deletion in memory, thereby making it faster than the original SPECK. Simulation results show that the proposed algorithm outperforms SPECK image coder in terms of memory requirement while retaining the coding efficiency and scalability property of the original SPECK. Proposed coder requires 0.75 bit per pixel state memory which is 9.38 % of the memory required to store the image. Therefore proposed NLSK coder is suitable for resource constrained portable hand held devices and WMSN's.

Keywords: *Image compression, wavelet image coder, SPECK, Listless image coder, memory efficient image coder, Block based coding*

1. Introduction

Use of image communication through wireless networks, using handheld mobile/ portable multimedia devices is increasing day by day. Also Wireless multimedia sensor network (WMSN's) require image transmission among its nodes and hubs over bandwidth limited wireless channels. However, these devices are constrained in terms of processing memory, battery life and processing power, and the real time transmission of images through these devices over wireless channels require an embedded (bit rate scalable) image compression algorithm that is not only efficient, but also requires minimal resources (processing memory, computational power and battery lifetime [1-5].

Among the available state of art image coders, JPEG2000 [6] and EBCOT [7] coders have increased computational complexity and has limited embedded property, while wavelet based zero tree coders [8-10] and zero block coders [11-14] are fully embedded coders with reduced complexity but with higher processing memory requirement. Among the wavelet based coders, set partitioning in hierarchal trees (SPIHT) [9] and set partitioning embedded block coder (SPECK) [12] coders are most suitable coders for resource constrained handheld devices and WMSN's [4, 15]. Both coders use data dependent list (dynamic state memory) to keep track of coefficients/block/sets are to be tested for their insignificance in coding process. Use of continuously growing data dependent lists in these coders not only result in variable and large memory requirement, but also necessitate the need for memory management as the list nodes are added, deleted, or moved from one list to another. This also results in significant increase in coding time due to multiple accesses of memory and memory management. For high resolution images or high bit rate coding of images using portable devices with limited resources (memory, computational power), these problems become more severe. As SPECK uses only two lists compared to three lists of SPIHT, SPECK coder is fast and requires less memory than SPHIT while coding efficiency is at par with SPIHT.

Wavelet based coders achieve compression by transformation of image using wavelet transform in a sub band structure and encoding of bit planes by aggregating a large number of insignificant coefficients either in form of zero-trees or zero-blocks and use linked-list (dynamic state memory) to keep track of sets and coefficients to be tested for their significance. Figure 1 shows the block diagram of wavelet based set partitioning coders. These coders require memory for wavelet transformation and use state memory in form of various lists to store state of sets and coefficients during coding.

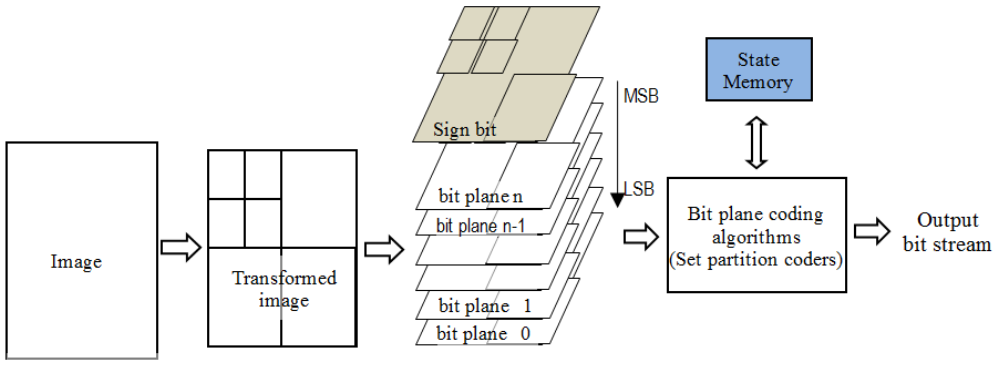


Figure1. Block Diagram of Wavelet based Bit Plane (set partitioning) Coders

Various listless versions of zero tree and zero block coders [16-20] have been suggested by researchers. These coders avoid use of variable and data dependent lists by using fixed size state table or markers. The markers are placed in state memory and are updated as per partitioning decisions in the coding process. Though these coder use fixed size memory, the memory requirement of theses coders are significantly high for low memory applications. The listless implementation of SPIHT coder NLS [17], uses 4 bit per coefficient marker, implementation in [18] uses 3 bit per coefficient marker, while listless versions of SPECK algorithm, LSK [19] uses 2 bit per coefficient marker, MLSK [20] uses 1 bit per coefficient marker to keep track of blocks/ trees and coefficients to be tested for significance in coding process, which is 128 KB, 96 KB, 64 KB and 32 KB respectively for a 512x512 image.

In this paper, we propose a no list version of SPECK algorithm for efficient coding of wavelet transformed images that uses 0.75 bit per coefficient state memory to keep track of blocks and coefficients to be tested for their significance. The proposed algorithm is referred as **N**o **L**ist **SPECK** coder (NLSK) due to the obvious reasons.

The paper is organized as follows. Section 2 presents the overview of SPECK and LSK algorithm. Section 3 presents the proposed NLSK algorithm including the linear indexing scheme. Simulation results and related discussions are presented in Section 4 and finally the paper is concluded in Section 5.

2. Overview of SPECK and LSK Algorithm

SPECK [12] coder exploits intra sub band correlation of wavelet transform coefficients. It is bit plane coding algorithm and each pass comprises; sorting, and refinement. It uses two linked lists: list of insignificant sets (LIS) and list of significant pixels (LSP). The coding algorithm initializes with defining initial threshold, which depends on the maximum value in the wavelet coefficients (chosen as a power of two: $T = 2^n$, where n is bit plane number). List Then, the LIS is initialized with set (block) of type *S* which is the root of the pyramid (LL-sub band), and a set of type *I* which is the remaining part of the image. LSP is initialized empty.

In the sorting pass, the algorithm proceeds by testing the significance of type *S* blocks with the threshold and significance is encoded. If a set *S* is found to be significant, it is partitioned by using quad partitioning scheme. Significant set is partitioned into four equal sets; significant set is removed from LIS wile newly formed sets are added to LIS. A set of size 1x1 corresponds to single coefficient, and when a coefficient is found significant for the first time, corresponding sign bit is also coded and coefficient is send to LSP.

After processing the *S* sets of LIS, set *I* is tested for significance. If a set *I* is found to be significant, it is partitioned by yet another partitioning scheme, called the octave band partitioning. Significant *I* set is partitioned into four sets: three sets of type *S* and one of type *I* which is of reduced size. The size of each of these three *S* sets is the same as that of the chopped portion of Image resulting in *I* set. The significant *I* set is removed from LIS while newly formed sets are added to LIS. The octave band partitioning scheme exploits the hierarchical pyramidal structure of the sub band decomposition, and 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. Once the set *I* is partitioned by the octave band partitioning method, the three *S* sets are processed in the regular image-scanning order, after which the newly formed reduced *I* set is processed. Once all sets in LIS have been processed for a particular threshold, 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. The threshold is then reduced by a factor of two and the sequence of sorting and refinement passes is repeated for sets in the LIS against this lower threshold. For processing the sets in LIS, the sets are processed in increasing order of their size. The whole process is repeated until the desired bit rate is achieved.

Though SPECK is an efficient algorithm with reduced computational complexity than SPIHT, use of data dependent list (LIS and LSP) requires large run time data dependent memory and also adds to the computational complexity due to multiple accesses of memory and addition / deletion of nodes in memory. This limits application of SPECK coder in memory constrained environment such as handheld multimedia devices and WMSN's.

Listless SPECK (LSK) [19] is listless implementation of SPECK but it does not use *I* partition. State information of coefficient /block is kept in a fixed size array, with two bits per coefficient of image to enable fast scanning of the bit planes. In LSK, efficient skipping of blocks of insignificant coefficients is accomplished using a recursive zigzag image indexing scheme (linear indexing). Instead of indexing the array coefficients using two indices, two-dimensional array is converted in to a one-dimensional array using linear indexing. The following markers are placed in the 2 bit per coefficient state table mark, to keep track of the set partitions.

MIP : The pixel is significant or untested for this bit-plane.

MNP : The pixel is newly significant and will not be refined for this bit-plane.

MSP: The pixel is significant and will be refined in this bit plane.

MS2: The block of size 2×2, i.e., 4 elements to be skipped.

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

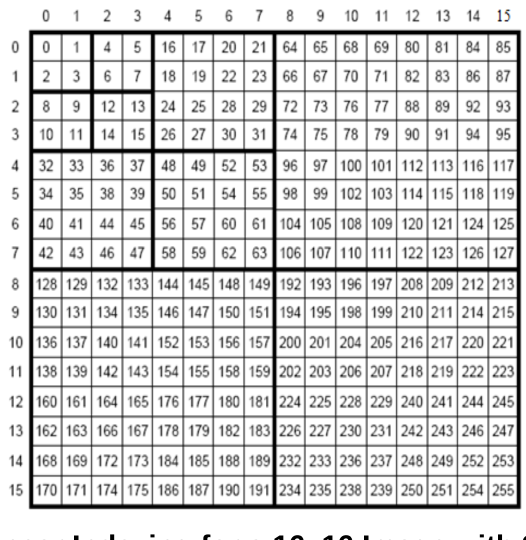
LSK is listless implementation of SPECK coder where state information of coefficient / block is kept in a fixed size array, with two bits per coefficient. For a 512×512 image decomposed to 5 levels, the memory required for markers is 64 KB while for image size of 1024 x 1024, the memory required for markers is 256 KB. For memory constrained environment such as handheld multimedia devices and WMSN's the amount of memory required is significantly higher. Also LSK does not use *I* sets of SPECK thereby generating more bits in earlier passes.

3. Proposed NLSK Algorithm

The proposed NLSK algorithm is a novel implementation of SPECK algorithm where state of sets during coding are stored in a fixed size of static memory. It uses dyadic wavelet transform with lifting scheme, which is a fast implementation of conventional wavelet transform and reduces memory requirement of the transform. The transformed image is then converted into linear index [16, 21]. The algorithm achieves functionalities of LIS and LSP are obtained by using small fixed size memory. The concept of linear indexing and detailed algorithm are discussed below.

3.1. Linear Indexing

The linear indexing allows transformation of two dimensional arrays into one dimensional array using Z scan order. Let wavelet transform of a (M x M) square image be $\{C_{i,j}\}$, and let r and c be the row and column indices of a particular wavelet coefficient. The linear index i of transformed image $\{C_i\}$, varying in the range of 0 to M^2-1, can be obtained by simply interleaving the bits of binary representation of r and c.



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	1	4	5	16	17	20	21	64	65	68	69	80	81	84	85
1	2	3	6	7	18	19	22	23	66	67	70	71	82	83	86	87
2	8	9	12	13	24	25	28	29	72	73	76	77	88	89	92	93
3	10	11	14	15	26	27	30	31	74	75	78	79	90	91	94	95
4	32	33	36	37	48	49	52	53	96	97	100	101	112	113	116	117
5	34	35	38	39	50	51	54	55	98	99	102	103	114	115	118	119
6	40	41	44	45	56	57	60	61	104	105	108	109	120	121	124	125
7	42	43	46	47	58	59	62	63	106	107	110	111	122	123	126	127
8	128	129	132	133	144	145	148	149	192	193	196	197	208	209	212	213
9	130	131	134	135	146	147	150	151	194	195	198	199	210	211	214	215
10	136	137	140	141	152	153	156	157	200	201	204	205	216	217	220	221
11	138	139	142	143	154	155	158	159	202	203	206	207	218	219	222	223
12	160	161	164	165	176	177	180	181	224	225	228	229	240	241	244	245
13	162	163	166	167	178	179	182	183	226	227	230	231	242	243	246	247
14	168	169	172	173	184	185	188	189	232	233	236	237	248	249	252	253
15	170	171	174	175	186	187	190	191	234	235	238	239	250	251	254	255

Figure 2. Linear Indexing for a 16x16 Image with 3-level DWT

The linear indexing scheme allows enlisting coefficients of sub bands consecutively in their wavelet coefficients pyramidal structure. Linear indexing efficiently supports the operations on coefficient positions, needed for tree-based/block-based algorithms with one operation instead of two. Figure 2 shows linear indexing of an 16x16 image with three level dyadic transform in Z scan order.

3.2. NLSK Algorithm

Consider an zero mean square image X of size (M, M) and $M=2^m$, that after L level of dyadic Lifting wavelet transform is read into the linear array $\{C_i\}$, using linear indexing of $M_{pix}=4^m$ coefficients. The transformed image $\{C_i\}$, exhibit a hierarchical pyramidal structure. The LL- band (root set) has $M_{pix}/4^L$ coefficients. NLSK uses set partitioning scheme of SPECK algorithm, the functionalities of LIS and LSP are obtained in NLSK by using a fixed size state memory *'SB'*(State of Blocks) to keep track of blocks of size 4 (2x2 blocks) to be tested for their significance.

The state memory *SB* of size $M_{pix}/4$ maps all possible nodes blocks of size 4 (2x2 block) and each element of *SB* stores state of the mapped block. States and its meaning are listed below.

'0' :	This node is not the beginning of any block set and is part of a larger block
MIB :	This node (representing the beginning of block) is untested for this bit plane.
MI :	This node is beginning of a *I* set
MNB :	This block of size 4 represented by the node is found significant in current pass.
MSB :	This block of size 4 represented by the node is found significant in previous passes and may also contain insignificant coefficient along with the refinement coefficient.
MRB :	All coefficients of this block of size 4 represented by the node has been found significant in previous passes and require refinement.

The encoder algorithm shown by Figure 3 is performed for each bit plane n starting with threshold $T=2^n$ and decrementing up to 0 or until a bit budget is achieved. Significance function used in NLSK algorithm is given in equation (1). Significance of a block B against a threshold $T= 2^n$ is given as,

$$S_n(B) = \begin{cases} 1, & if \quad T \le \max\limits_{i\in B}(|C_i|) < 2T \\ 0, & if \quad \max\limits_{i\in B}(|C_i|) < T \end{cases} \;\ldots\ldots \qquad (1)$$

The algorithm begins with initialization of fixed size memory *'SB'* of $N_{pix}/4$ elements with *SB*(0) set as MIB and *SB*($N_{pix}/4^{(L+1)}$) set as MI and rest elements set as zero. Threshold is initialized as $T=2^n$, where n is the highest bit plane number. Each pass of proposed NLSK algorithm consists of sorting pass and refinement pass.

In sorting pass, algorithm begins scanning state of elements of *SB*. *SB(k)='MIB'*, indicates a *S* block. The size of *S* block is determined by counting the consecutive elements of *SB* (representing blocks of size 4) whose state is '0'. The *S* block is tested against threshold and its significance is encoded. An insignificant *S* block is skipped and *SB* index is incremented accordingly. A significant block is partitioned using quad partition scheme by setting the corresponding elements of *SB* as 'MIB". Then quad block is tested against threshold. A significant block is quad partitioned till a block of size 4 (2x2 block) is reached. For a significant block of size 4, its state is updated as *'MNB'* and each coefficient of the block is tested against threshold and its significance is encoded. If a coefficient is found significant, its sign bit is encoded.

For *SB(k)= MI,* the corresponding *I* set is tested for significance and its significance is encoded. If *I* set is found insignificant, the sorting pass ends, A significant *I* set is partitioned by octave band partitioning and is effected by setting states of new *S* sets as *'MIB'* in *SB* ,and

the reduced *I* set if it exists, then state of new I set in the corresponding element of *SB* is set as '*MI*'. Then the testing of newly formed S blocks are done as explained above.
SB(k)=' MSB' indicates that the corresponding block has been found significant in previous passes It also indicates that, it contains coefficient requiring refinement and may contain coefficient requiring significance testing. The coefficient requiring refinement (whose magnitude is less than twice of threshold) is skipped and significance of other coefficients are encoded. If the coefficient is found to be significant, its sign bit is encoded. After all *S* blocks and *I* blocks are coded, the sorting pass ends and refinement pass begins.

```
Encoding algorithm
(1)Initialization:
Set n = $\lfloor \log_2(\max\{|C_i|\}) \rfloor$; T=$2^n$, initial threshold; L:transforms levels, $M_{pix}=4^m$; size of linear array
Initialize SB of size ($1xM_{pix}/4^L$) with SB(0)='MIB' , SB($4^{(m-L-1)}$)='MI' and rest elements set to '0'
(2) Sorting pass
k=1;
while k<$M_{pix}/4$
{    IF SB(k)='MIB' ; then
          i=4k; Count=1; temp_ind=j;
          while SB(temp_ind)=0; Count=Count+1; temp_ind=temp_ind+1;}
          λ=4*Count; For block $S_i^\lambda = \{C_j, i \le j < i+\lambda\}$, Output $S_n(S_i^\lambda)$
          if $S_n(S_i^\lambda)$=1
              if λ>4 ; Set SB(k+ λ/4), SB(k+2 λ/4), SB(k+3 λ/4), as 'MIB' marker
              else     Set SB(k)='MNB',
                  for j=0:3
                      Output $S_n(C_{(i+j)})$
                      if $S_n(C_{(i+j)})$=1; Output sign bit }}}
              k=k+1}
          `else   k=k+ λ/4}
     elseif SB(k)='MI'; a I set marker, then for set $I=\{C_j, i \le j < M_{pix}\}$, i=4k;  Output $S_n(I)$
              if $S_n(I)$=1 ;  Set SB(4k) as 'MI' and Set SB(2k) and SB(3k) as 'MIB'
              else k=$M_{pix}/4$}
     elseif SB(k)='MSB'; then
              i=4k;
              for j=0:3
                  if T≤$C_{(i+j)}$<2T;      Output $S_n(C_{(i+j)})$
                      if $S_n(C_{(i+j)})$=1; Output sign bit}}}
              k=k+1}
(3) Refinement Pass
for k=0:($M_{pix}/4$)-1
     if SB(k)='MNB';      SB(k)='MSB'
     elseif SB(k)='MSB'
          i=4k; Count_refin=0;
          for j=0:3
              if $C_{(i+j)}$≥2T;  Output nth MSB of $C_{(i+j)}$ and increment Count_refin}}
              if Count_refin=4, then SB(k)='MRB'}
     elseif SB(k)='MRB'
          for j=0:3
              if $C_{(i+j)}$≥2T;  Output nth MSB of $C_{(i+j)}$ }}}}
(3) Quantization step update
   Decrement n by 1 and go to step 2  }
```

Figure 3. Pseudo-code of NLSK Algorithm

In refinement pass all the element of state memory *SB* are scanned. State' *MNB'* indicates that the corresponding block of size 4 has been found significant in this bit plane and will not require refinement in this bit plane. The state of block is changed to *'MSB'*. The state 'MRB' indicates that all the four coefficients of the block has been found significant in previous passes and refinement bit are encoded, For block with State *'MSB'*, coefficient requiring refinement (whose magnitude is greater than twice of threshold), the refinement bit is encoded. If refinement bits are set for all the elements of the block, then state of the block is changed to *'MRB'*.

The proposed NLSK coder is symmetrical and the decoder of NLSK follows the same overall procedure as the encoder with some low-level changes. To decode, use input instead of output, and set the bits and sign of coefficients. The decoder performs mid-tread de-quantization for coefficients that are not fully decoded.

As proposed NLSK use the same set structures and partitioning rules as that of SPECK coder, each coder produces the exact same output bits in each pass, though in a different order. The reordering of bit stream occurs because SPECK performs significance tests roughly breadth first while NLSK, like LSK coder, performs the significance tests strictly breadth first. Thus for the NLSK slight degradation of PSNR of decoded image may occur, compared to that of original SPECK, if bit budget is exhausted in the middle of a bit plane as bit budget will also be used in refinement thereby reducing the new significant pixels identified. However, at the end of sorting pass, NLSK encodes same information as that of SPECK. Thus the NLSK generates embedded bit stream with progressive transmission and by using small size fixed state memory, thereby significantly reducing the computational cost involved in multiple memory access and addition and deletion in dynamic state memory or appending the markers in static lists.

3.3 State Memory Requirement

In this section, we compare the static memory requirement of proposed NLSK coder with LSK and SPECK coder. SPECK use two linked list to store the state information of coefficients/ blocks during coding there by requires a data dependent state memory (dynamic memory) that also necessitates need of memory management. The proposed NLSK coder uses a fixed size static memory to store state information of blocks to facilitate encoding. Here the memory required for storing the state information in NLSK and LSK and SPECK are estimated. The required memory size for NLSK and LSK is fixed while for SPECK coder, it depends on the number of entries in the corresponding lists; LIS and LSP.

In SPECK, LIS is implemented by using multiple lists depending on the size of the block including a single coefficient. The list LSP contains the address of significant coefficient. Let

N_{LIS} = Total number of block in LIS

N_{LSP} = Number of nodes in LSP

b = Number of bits to store addressing information

Then, the total required memory size (due to linked lists) in SPECK is given by

$$M_{SPECK} = \{b(N_{LIS} + N_{LSP})\} \quad \text{bits} \qquad (2)$$

LSK uses a fixed size static memory *mark* of size equal that of coefficients array to store state of each node. Only two bit markers are used to define a node state. For a image size (M x M), memory required for LSK is

$$M_{LSK} = 2M^2 \quad \text{bits} \qquad (3)$$

The proposed NLSK uses a static memory *SB* of size equal to one fourth of coefficients array to store state of a node (a 2 x2 block of block of 4 coefficients). Only three bits are used to define a node state. For an image size (M x M), state memory required for NLSK coder is

$$M_{NLSK}=3M^2/4 \qquad \text{bits} \qquad (4)$$

For SPECK coder, in worst case all the coefficients may be in either LIS or in LSP. Therefore, for an image size of 512x512, worst case state memory required for SPECK is 576 KB, for LSK it is 64 KB and for proposed NLSK coder it is only 24 KB. Therefore the proposed NLSK coder (No list implementation of SPECK) is memory efficient and requires small fixed size state memory in comparison to other implementation LSK.

3.4 Computational Complexity

Proposed NLSK coder is no list implementation of SPECK coder. In SPECK algorithm, quad partitioning / octave partitioning results in addition of four new entries, deletion of one previous entry in LIS, and also necessitates memory management. In addition multiple access of linked list adds to the computation complexity. Proposed NLSK coder and LSK coder use a fixed size static memory to keep track of set partitioning resulting in complexity reduction of appending of memory, multiple accesses of memory and memory management. In comparison to LSK, proposed coder records state of blocks of size 4 (2x2 block) instead of coefficients (1x1 block) thereby reducing the computational cost involved in memory access approximately by one fourth. Though in significant block of size 4 in which all the four coefficients are not found significant, coefficients are compared with threshold and twice of threshold. However the computation cost involved in two comparisons (for block with state MSB) is quite less than the saving of computational cost involved in multiple memory access. Therefore the proposed NLSK coder is fast in comparison to LSK and SPECK.

4. Simulation Results

The coding efficiency, memory requirement, of the NLSK coder is evaluated and compared with SPECK, and LSK coder on many grey scale classical test images (8 bits/pixel). A dyadic wavelet decomposition of five levels using bi-orthogonal 4.4 filters with lifting scheme is used to transform the images. Floating point, transform coefficients are quantized to the nearest integers, and read into the linear array using linear indexing. All the coders are simulated using MATLAB and executed on a PC with Intel CPU T 2080 @ 1.73 GHz having 512 MB RAM. All the coders are implemented without arithmetic or context based coding. The test images are binary encoded once up to the last bit plane and are decoded at different bit rates from the same embedded bit-stream.

4.1 Coding Efficiency

Coding efficiency of proposed NLSK coder for image ‘LENA’ is compared with SPECK and LSK coder and the plot of coding efficiency, measured in terms of peak signal to noise ratio (PSNR) versus bit rate is given in Figure 4. It can be observed from the figure that coding efficiency of the NLSK is similar (slightly higher) to that of LSK. It should be noted that at the end of each bit plane coding gain obtained in NLSK and SPECK coders are identical, while for bit rates somewhere in middle of a bit plane (pass), the efficiency of NLSK is slightly lower than that of SPECK. This is because NLSK strictly breadth first search while SPECK does roughly breadth first search and process smaller sets first, resulting in slight reduction of PSNR for bitrates in the mid of bit plane. It can be also observed that reduction of PSNR of NLSK with respect to SPECK in middle of bit plane is less for lower bit rates.

The coding efficiency of Proposed NLSK, SPECK, and LSK coder for various classical test images are given in Table 1. It can be observed from the table that PSNR of NLSK coder is at par with SPECK (slightly lower at some points). The maximum reduction of PSNR with SPECK is 0.3 dB for CAMERAMAN, and 0.4 dB for LENA, LOSTLAKE, MAN. The reduction in coding gain at some points is due to bits reordering in proposed NLSK for bit rates exhausted in middle of a pass. In comparison to LSK coder, PSNR of proposed NLSK coder is slightly higher. The increment of PSNR with LSK is 0.08 dB for CAMERAMAN, LENA, 0.15 dB for LOSTLAKE, and 0.01 dB for MAN. This gain is because if use of *I* block in NLSK. In early passes use of *I* block result in saving of several bits resulting in gain of PSNR of NLSK.

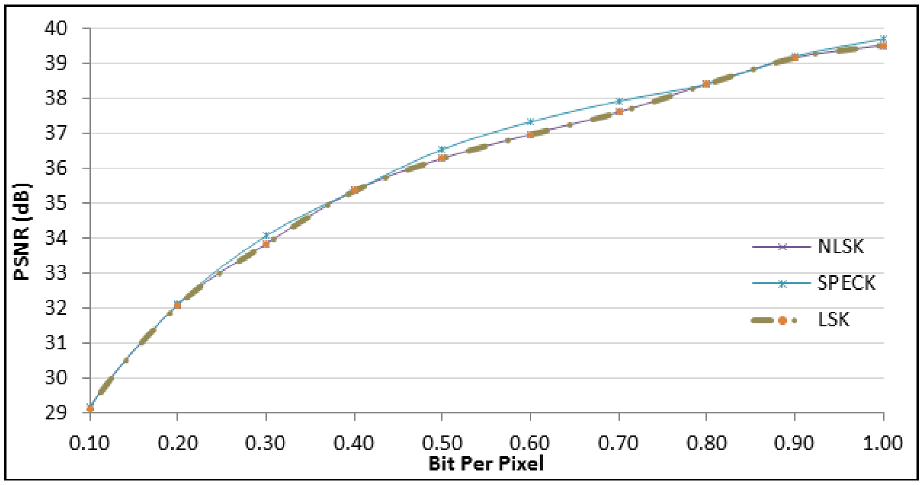


Figure 4. PSNR vs Bit rate for NLSK, SPECK and LSK for LENA (512x512)

Table 1. Coding Efficiency of Proposed NLSK, LSK and SPECK Coder

BPP	CAMERAMAN (256 x256)			LENA (512 x 512)			LOSTLAKE (512 x 512)			MAN (1024 x1024)		
	NLSK	SPECK	LSK	NLSK	SPECK	LSK	NLSK	SPECK	LSK	NLSK	SPECK	LSK
0.05	20.83	20.84	20.75	26.34	26.38	26.30	22.79	22.79	22.64	25.08	25.13	25.07
0.10	22.78	22.82	22.74	29.16	29.18	29.14	24.24	24.41	24.21	27.09	27.12	27.09
0.20	25.31	25.30	25.28	32.10	32.11	32.08	26.34	26.46	26.33	29.61	29.71	29.61
0.30	26.90	27.08	26.88	33.84	34.07	33.83	27.61	27.73	27.59	31.32	31.39	31.32
0.40	28.27	28.38	28.24	35.37	35.37	35.36	28.78	28.79	28.75	32.41	32.58	32.41
0.50	30.07	30.08	30.04	36.29	36.55	36.29	29.56	29.72	29.55	33.44	33.48	33.43
0.60	30.99	31.11	30.97	36.98	37.34	36.97	30.22	30.50	30.21	34.38	34.41	34.37
0.70	31.80	32.07	31.78	37.62	37.93	37.61	30.80	31.20	30.79	34.93	35.10	34.93
0.80	32.96	33.07	32.93	38.43	38.42	38.42	31.65	31.81	31.64	35.41	35.70	35.40
0.90	34.14	34.14	34.11	39.18	39.22	39.17	32.35	32.34	32.33	35.81	36.21	35.81
1.00	35.06	35.19	35.05	39.54	39.71	39.54	32.95	33.03	32.94	36.34	36.64	36.33

* Above results are without context coding or arithmetic coding

The visual performances of NLSK coder and SPECK and LSK coder are shown in Figure 5 for the classical test images at o.125 bit per pixel. In subjective evaluation, the reconstructed images appear to be similar for all the coders.

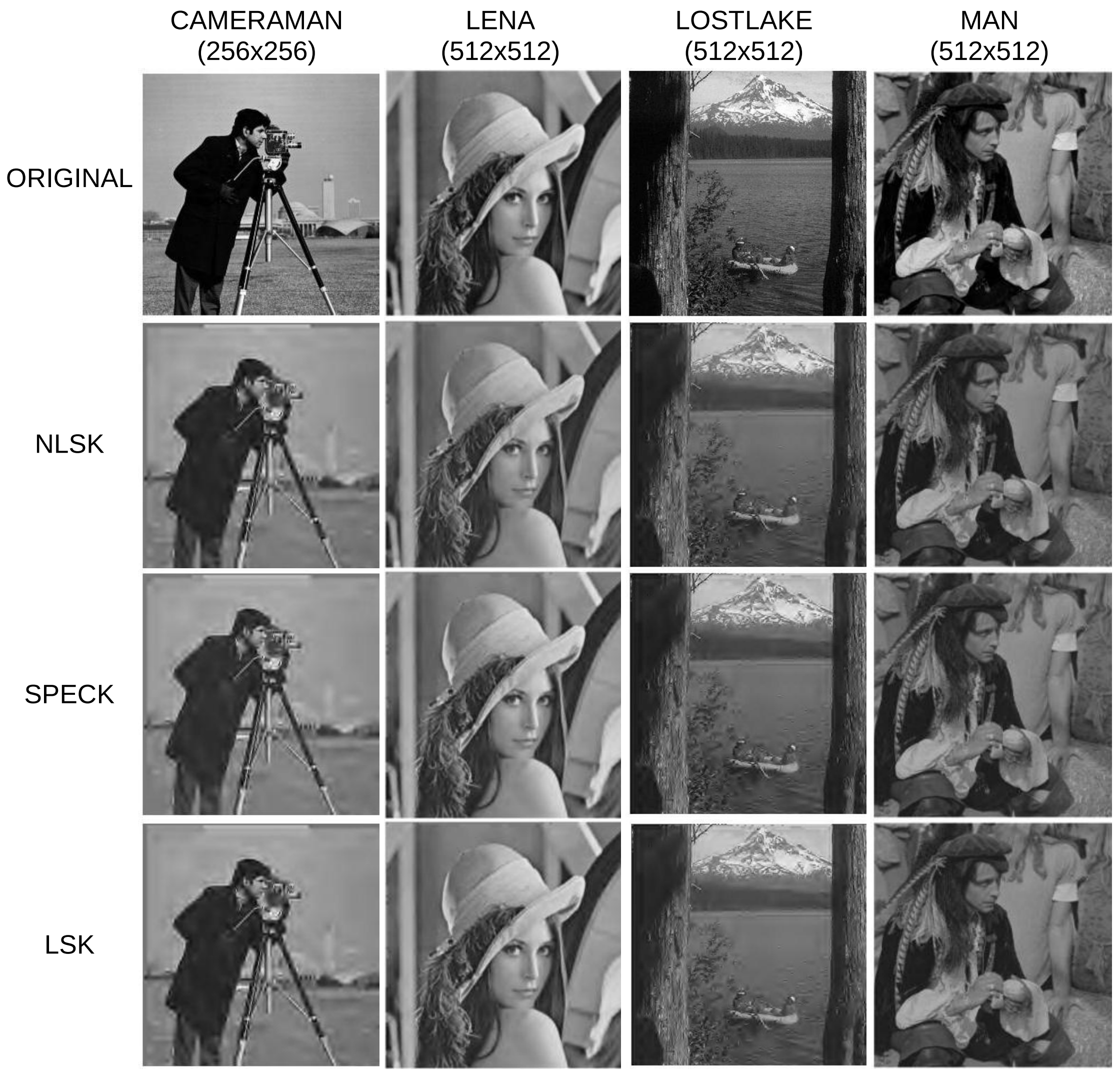


Figure 5. Decoded Images for NLSK, SPECK and LSK at 0.125 bit per pixel

4.2 State Memory Requirement

The memory requirements of proposed NLSK, SPECK and LSK coder are compared in Table 2 for classical test images of different sizes. SPECK coder requires data dependent variable (dynamic) state memory which depend on the bit rate for a typical image, while LSK and proposed NLSK coder uses fixed size static memory. It can be observed from the table that proposed NLSK coder outperforms others in respect of processing (state) memory. LSK require 2 bit per coefficient static memory while NLSK require 0.75 bit per coefficient (3 bit per block of 4 coefficient) to store state information .

Table 2. State Memory Requirement (KB)

BPP	CAMERAMAN (256x256)			LENA (512x512)			LOSTLAKE (512x512)			MAN (1024 x1024)		
	NLSK	SPECK	LSK	NLSK	SPECK	LSK	NLSK	SPECK	LSK	NLSK	SPECK	LSK
0.05	6.0	2.6	16.0	24.0	10.3	64.0	24.0	14.3	64.0	96.0	51.6	256.0
0.1	6.0	5.4	16.0	24.0	23.0	64.0	24.0	23.6	64.0	96.0	113.8	256.0
0.2	6.0	11.5	16.0	24.0	46.7	64.0	24.0	42.4	64.0	96.0	197.7	256.0
0.3	6.0	13.8	16.0	24.0	58.4	64.0	24.0	70.7	64.0	96.0	274.7	256.0
0.4	6.0	20.0	16.0	24.0	87.9	64.0	24.0	100.4	64.0	96.0	359.2	256.0
0.5	6.0	25.0	16.0	24.0	88.5	64.0	24.0	100.4	64.0	96.0	510.7	256.0
0.6	6.0	25.2	16.0	24.0	113.5	64.0	24.0	123.0	64.0	96.0	551.1	256.0
0.7	6.0	30.0	16.0	24.0	147.0	64.0	24.0	153.1	64.0	96.0	569.8	256.0
0.8	6.0	36.8	16.0	24.0	182.7	64.0	24.0	188.1	64.0	96.0	676.8	256.0
0.9	6.0	42.1	16.0	24.0	182.7	64.0	24.0	216.8	64.0	96.0	815.1	256.0
1	6.0	42.1	16.0	24.0	182.7	64.0	24.0	216.8	64.0	96.0	968.8	256.0

*Memory to store wavelet coefficients is excluded in above results

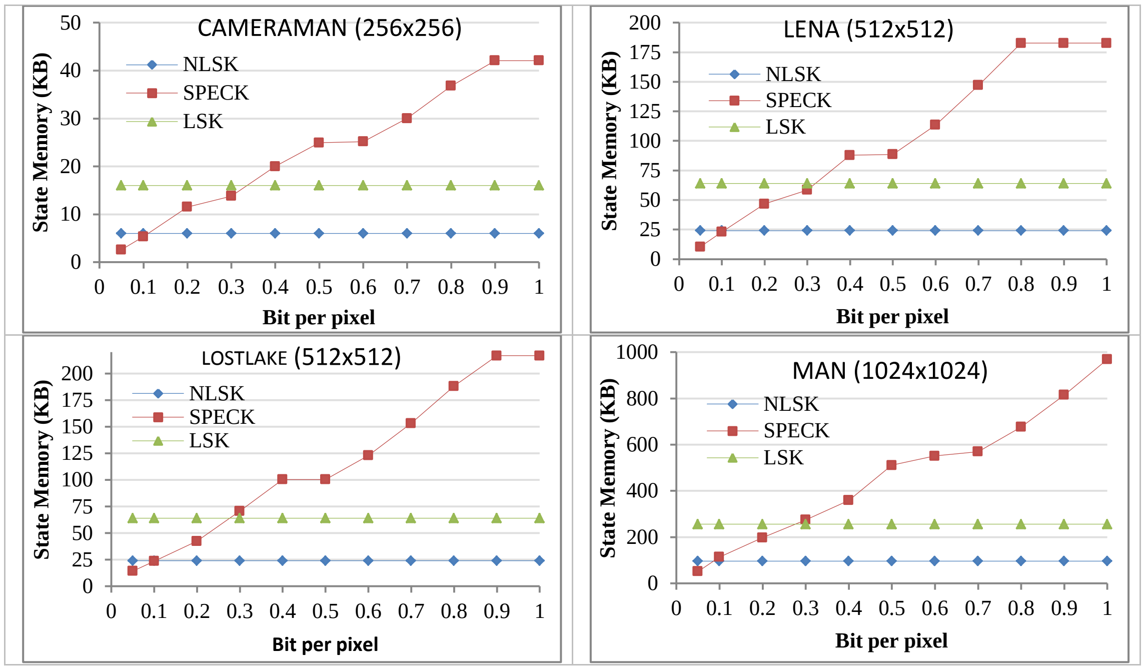


Figure 6. State Memory Requirement of Various Test Images

Figure 6 shows the state memory required for the coding of images CAMERAMAN, LENA, LOSTLAKE and MAN. From the figure it is evident that in terms of processing memory, LSK outperforms SPECK coder for bit rates higher than 0.32 bit per pixel. While proposed NLSK coder being memory efficient, outperforms SPECK and LSK coder for bit all bit rates higher than 0.1 bit per pixel. For low resolution image CAMERAMAN, proposed coder requires only 6 KB state memory, while LSK requires 16 KB memory. For high resolution image MAN, the proposed coder requires 96 KB state memory while LSK requires 256 KB memory. Therefore proposed NLSK coder is more suitable for low memory applications.

5. Conclusions

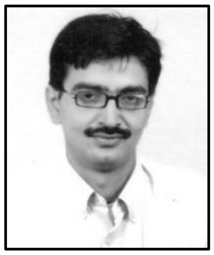
In this paper we have proposed a novel implementation of listless SPECK using fixed size markers array (0.75 bit per pixel) which is 9.38% of the memory required to store the image. It is observed that proposed NLSK coder is low memory implementation of SPECK with reduced computational complexity, while coding efficiency is at par with that of SPECK. NLSK coder generates same size of bit stream as that of SPECK in a pass but in different order. Bit ordering results in slightly lower coding gain during sorting pass, but at the end of each bit plane coding gain is exactly identical. Proposed coder offers coding gain similar (slightly higher) to LSK, but requires only 37.5% of the state memory required in LSK. Due to low memory requirement and low computational complexity, the NLSK coder is suitable for resource constrained applications such as portable camera, PDAs and wireless multimedia sensor networks.

References

[1] D. Lee, H. Kim, M. Rahimi, D. Estrin and J. D. Villasenor, “Energy efficient image compression for resource-constrained platforms”, IEEE Trans. Image Process, vol. 18, No. 9, **(2009)**, pp. 2100-2113.
[2] I. F. Akyildiz, T. Melodia and K. R. Chowdhury, “A survey on wireless multimedia sensor networks”, Computer Networks (Elsevier) Journal, vol. 51, no.4, **(2007)**, pp. 921-960.
[3] I. F. Akyildiz, T. Melodia and K. R. Chowdhury, “Wireless Multimedia Sensor Networks: Applications and Testbeds”, Proceedings of IEEE, vol. 96, no. 10, **(2008)**, pp. 1588-1605.
[4] L. W. Chew, L. -M. Ang and K. P. Seng, “Survey of image compression algorithms in wireless sensor networks”, International Symposium on Information Technology Malaysia, **(2008)**, pp. 1-9.
[5] S. Misra, M. Reisslein and G. Xue, “A Survey of Multimedia Streaming in Wireless Sensor Networks”, IEEE Communication Surveys & Tutorials, vol. 10, no. 4, **(2008)**, pp. 18-39.
[6] D. Santa-Cruz, R. Grosbois and T. Ebrahimi, “JPEG 2000 performance evaluation and assessment”, Signal Processing: Image Communication, vol. 17, **(2002)**, pp. 113–130.
[7] D. Taubman, “High performance scalable image compression with EBCOT”, IEEE Trans. Image Process., vol. 9, **(2000)**, pp. 1158–1170.
[8] J. M. Shapiro, “Embedded image coding using zerotrees of wavelet coefficients”, IEEE Trans. Signal Process., vol. 41, **(1993)**.
[9] A. Said and W. A. Pearlman, “A new fast and efficient codec based on set partitioning in hierarchical trees”, IEEE Trans. Circuits Syst. Video Technol., vol. 6, **(1996)**, pp. 243–250.
[10] Z. Xiong, K. Ramachandran and M. T. Orchard, “Space-Frequency quantization for wavelet image coding”, IEEE Trans. Image Process., vol. 6, **(1997)**, pp. 677–693.
[11] C. Chrysalis, A. Said, A. Drukarev, A. Islam and W. A. Pearlman, “SBHP- A low complexity wavelet coder”, IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP’00), vol. 4, **(2000)**, pp. 2035–2038.
[12] W. A. Pearlman, A. Islam, N. Nagaraj and A. Said, “Efficient low complexity image coding with set-partitioning embedded block coder”, IEEE Trans. Circuits Syst. Video Technol., vol. 14, **(2004)**, pp. 1219–1235.
[13] S. T. Hsiang and J. W. Woods, “Embedded image coding using zero blocks of subband/ wavelet coefficients and context modeling”, IEEE Int. Symp. Circuits and Systems (ISCAS’00), vol. 3, **(2000)**, pp. 662–665.
[14] H. Arora, P. Singh, E. Khan and F. Ghani, “Memory efficient image coding with embedded zero block-tree coder”, Proc. IEEE Int. Conf. Multimedia and Expo2004 (ICME’04), vol. 1, **(2004)**, pp. 679–682.
[15] R. Sudhakar, Ms R. Karthiga and S. Jayaraman, “Image Compression using Coding of Wavelet Coefficients– A Survey”, ICGST-GVIP Journal, vol. 5, no. 6, **(2005)**, pp. 25-38.
[16] W. K. Lin and N. Burgess, “Listless zerotree coding for color images”, In Proc. of the 32nd Asilomar Conf. on Signals, Systems and Computers, vol. 1, **(1998)**, pp. 231-235.

[17] F. W. Wheeler and W. A. Pearlman, "SPIHT image compression without lists", IEEE conference on acoustics, speech and signal processing (ICASSP2000), vol. 4, **(2000)**, pp. 2047-2050.
[18] H. Pan and W. C. Siu, "A fast and low memory image coding algorithm based on lifting wavelet transform and modified SPIHT", Signal Processing: Image Communication, vol. 23, no.3, **(2008)**, pp. 146-161.
[19] M. V. Latte, N. H. Ayachit and D. K. Deshpande, "Reduced memory listless SPECK image compression", Elsevier's Journal of Digital Signal Processing, vol. 16, issue 6, **(2006)** November, pp. 817-824.
[20] N. R. Kidwai, E. Khan and R. Beg, "A Memory Efficient Listless SPECK (MLSK) Image Compression Algorithm for Low Memory Applications", International Journal of Advanced Research in Computer Science, vol. 3, no. 4, **(2012)**, pp. 209-215.
[21] G. Seetharaman and B. Zavidovique, "Z-trees: Adaptive pyramid algorithms for segmentation", in Proc. of the International Conf. on Image Processing (ICIP-98), **(1998)**, pp. 294-298.

Authors



Naimur Rahman Kidwai did B.Sc.Engg. from Aligarh Muslim University, India, MBA from Jamia Millia Islamia University, Delhi, and MTech (Digital Communication) from UPTU, Lucknow. Presently he is working as Assistant Professor, Integral University, Lucknow, India. He has 15 years of experience which include 12 years of teaching experience. His area of expertise is signal processing, communication systems, and image processing. He has many valuable publications to his credit

Monauwer Alam received his B.Sc(Engg) degree in Electronics and Communication from Magadh University Bodh Gaya, India in 2000 and and MTech (Digital Communication) from UPTU, Lucknow India. Presently he is working as Associate Professor, Integral University, Lucknow. He has 12 years of experience. His research interest include image compression and image enhancement. India

Ekram Khan obtained B. Sc. Engineering (1991)and M. Sc. Engineering (1994) degrees from Z. H. College of Engineering and Technology AMU Aligarh (India) and Ph. D. (2003) degree from University of Essex, UK, all in Electronics Engineering. He has joined the department of Electronics Engineering as a Lecturer in 1993 where currently he is working as a Professor. He was a recipient of the prestigious Commonwealth Scholarship to pursue his Ph. D. degree from University of Essex, UK. He has authored and co-authored over 70 papers in refereed academic journals and international conference proceedings. His areas of research are image/video coding, video transmission over wirelessand IP networks, video streaming. He is recipient of IBM research award for best disruptive idea in 2002. Dr. Khan spends summers of 2005 and 2006 as academic visitor to University of Essex (funded by Royal Society, UK). He has delivered many invited talks at national and international levels. He is a member of IEEE (USA), life member of IETE (India) and life member of SSI (India).

Rizwan Beg has completed M.Tech. and Ph.D in Computer Sc. & Engineering. Presently he is working as Professor and Head, Department of Computer Sc. & Engg., Integral University, Lucknow, India. He has 16 years of experience which include 14 years of teaching experience. His area of expertise is Software engineering, software quality and software project management. He has published many of the valuable research papers in various national and international journals. He is associate editor in chief of the journal Advancement in computing technology and is member of editorial board of various other national/ international journals.