STM JOURNALS

Journal of Artificial Intelligence Research & Advances
ISSN: 2395-6720(online)
Volume 3, Issue 3
www.stmjournals.com

Analysis of Image Coders

Shefa Hasnain*, Sadiya Bakhshi, Shazia Asif, Monauwer Alam
Electronics and Communication Department, Integral University, Lucknow, Uttar Pradesh, India

Abstract

In this paper we work on the performance analysis of well-known image compression algorithm i.e., wavelet based image compression which is Set Partition in Hierarchical Tree (SPIHT) and also no list SPIHT (NLS). As SPIHT uses three variable lists which require large hardware implementation and hence extra cost and high encoding and decoding time is required but NLS use marker instead of list which are placed on lower nodes of insignificant tree hence it requires less hardware and hence it is less complex then SPIHT and time of decoding and encoding is also inferior to SPIHT. The algorithm are implemented in MATLAB and compared using the parameter peak signal to noise ratio PSNR, encoding time and decoding time. Using the NLS technique the image obtained has less encoding as well as decoding time as compared to SPIHT but in context SNR SPIHT is better.

Keywords: *SPIHT, NLS, PSNR, Performance analysis, Bits per pixel*

****Author of Correspondence*** E-mail: shefahasnain@gmail.com

INTRODUCTION

The SPIHT algorithm is a generalization of the Embedded Zero Wavelet (EZW) algorithm proposed by Amir Said and William Pearlman [1]. In EZW we transmit a lot of information for little cost when we declare an entire subtree to be insignificant and all the coefficients in it with a zero tree label zero.

The SPIHT algorithms use a partitioning of the trees in a manner that tends to keep insignificant coefficient together in large subset. The partitioning decision are binary decision that are transmitted to the decoder, providing a significance map encoding that is more efficient than EZW.

SPIHT uses three variable lists

- List of significant pixel (LSP)
- List of insignificant pixels (LIP)
- List of insignificant set (LIS)

No List SPIHT (NLS) uses the same set structures and partitioning rules as SPIHT. The trees are tested for significance breadth first. Significance tests are related in a different order than SPIHT because SPIHT performs significance tests roughly breadth first, while NLS performs the tests strictly breadth first. Because the set splitting rules are the same, each coder produces the exact same output bits, though in a different order.

SPIHT ALGORITHM

We need to know about some data notation before describing the algorithms. SPHIT algorithm use data structure which is similar to that used by EZW algorithm but still it is not that much same. Again divide the coefficient into trees which formulate from the lowest resolution band. These coefficients are grouped into array of 2×2 matrixes which except for coefficient in band 1 are offspring's of a coefficient of a lower resolution band.

The coefficient which is in lower band resolution (Figure 1) is also divided into array of 2x2 matrixes. The coefficient which is in top most left corner of the array doesn't have any offspring.

Algorithm

Let Sn(t) is a function which decides the set of significance bit with respect to the threshold 2n.

Sn(t), if max(i,j){[Ci,j]} ≥ 2n
0, else

Whereas ci, j is the wavelet coefficient.

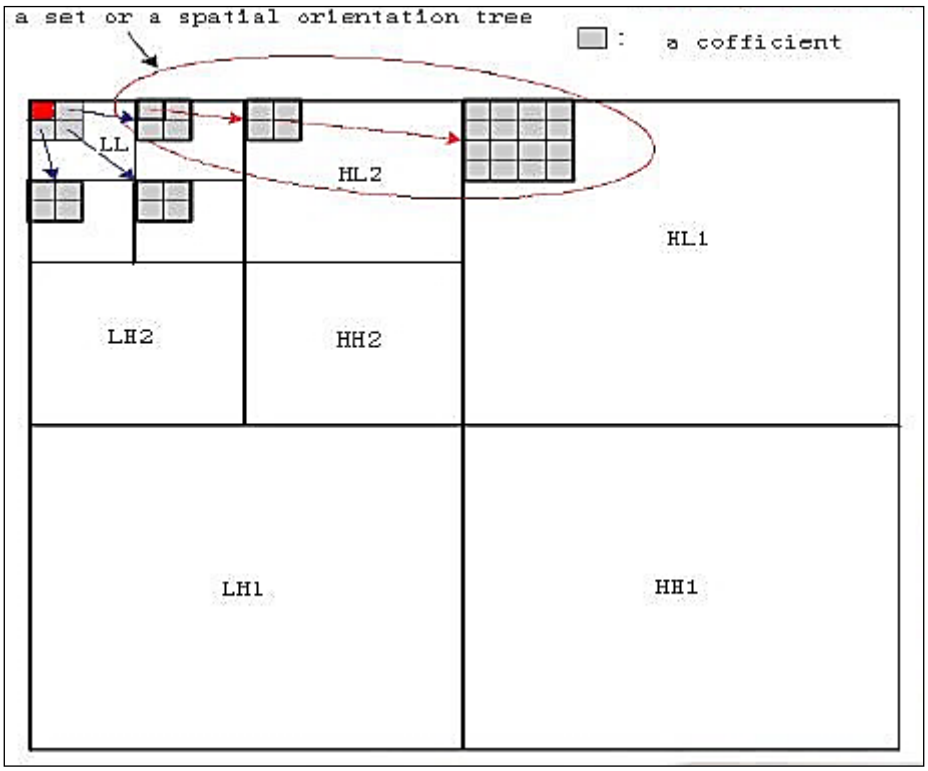

Fig. 1: *Hierarchal Tree.*

Initialization

1.1>output n=[log2max{[c(i,j)]}];

1.2>set LSP= 0 (empty)

1.3>set LIP=(i,j)€H;

1.4>set LIS=(i,j)€H, where D(i,j)≠0 set each entry in LIS as type A

Sorting Pass

2.1> For each entry (i,j) in the LIP do: Output Sn(i,j),

If Sn(i,j)=1 then move (i,j) to the LIP, and output the sign of cid

2.2> For each entry (i,j) in the LIS do;

2.3> if the entry is of type A then output Sn(D(i,j)),

If Sn(D(i,j))=1 then *For each (k,i)€o(i,j) do: Output Sn (k,i)

If Sn(k,i)=1 then

Add (k,i) to the LSP, output the sign of Ck,1 If Sn(k,1)=0 then

Add(k,i) to the end of the LIP,

If L(I,j)≠0 then

Move (i,j) to the end of the LIS, as an entry of type B, go to step 2.2.2) Otherwise remove entry (i,j) from the LIS

2.4> If the entry is pf type B Output Sn(L(i,j)),

*add each hm to the end of LIS as an entry of type A, *Remove(i,j) from the LSP

Refinement Pass

For each entry (i, j) in the LSP, except those included in the last sorting pass (i.e., with the same n), output the *n*th most significant bit of | ci,j |

Quantization Pass

Decrement *n* by 1 and go to Step 2.

NLS

There are three passes per bitplane. First, the insignificant Pixel (IP) pass which corresponds to SPIHT's LIP pass tests each lone insignificant pixel for significance [2]. The significant set (IS) pass, like SPIHT's LIS pass, tests each multiple-pixel set for significance, splitting and repeating as needed. Finally, the refinement (REF) pass, like SPIHT's LSP pass, refines pixels found significant in previous bitplanes.

State Table Markers

The following markers are placed in the 4 bit per coefficient state table marker, to keep track of the set partitions.

M*P: Each of these first four markers are for alone pixel.

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

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

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

MCP like MIP, but applied during partitioning in the IS pass so the new pixel set will be tested for significance immediately during the same IS pass, while MIP pixels are skipped.

MD: The pixel is the first (lowest index) child in a set consisting of all descendants of its parent.

MG: The pixel is the first (lowest index) grandchild in a set consisting of all grand descendants of its grandparent pixel, but not including the grandparent or the children of the grandparent.

MN*: The following markers are used on the leading node of each lower level of an insignificant tree. As the image is scanned, these markers indicate that the next block of pixels is insignificant.

MN2: The pixel is the first grandchild of a MD set. This pixel and its 16 (4 by 4) 1st cousins can be skipped.

MN3: The pixel is the first great grandchild of a MD or MG set. This pixel and its 64 (8 by 8) 2nd cousins can be skipped.

MN6: The pixel is the first 6th generation descendant of a MD or MG set. This pixel and its 4096 (64 by 64) 5th cousins can be skipped.

INSIGNIFICANT PIXEL PASS [3-8]

I = 0, while I < 1 start the IP pass
If mark [I] = MIP insignificant pixel
Output (d = (val[i] AND s))send bit
if d
output (sign[i]) send the sign
Mark[i] = MNP pixel now newly significance
i=i+l move to next pixel
e l s e
i = i + skip[mark[z]] move past set/block

INSIGNIFICANT SET PASS

i = 0, while i< I start the IS pass
If mark[i] = MD a set of descendants
Output(d = (dmax[Li/4J] ANDs))
if d if the set is significant
mark[i] = mark[i + I] = MCP split into four children
mark[i+2] = mark[i+3] = MCP (will test these next)
mark[4i] = MG and the grand descendants
no increment here so new MCP pixels processed next
else
i = i + 4 move past the siblings in this set
else if mark[i] = MG a set of grand descendants
output (d = (gmax[Li/16]] ANDs))
if d
if the set is significant
mark[z] = mark[i + 41 = MD split into four sets
mark[i+8] = mark[i+l2] = MD (will test these next)
mark the borders of these new sets
push(i),push(i + 4), push(i + 8), push(i + 12)
no increment here so new MD sets processed next
else
i = i + 1 6 move past the cousins in this set
else if mark[i] = MCP an insignificant pixel
output(d = (val[i] ANDs))
if d if the pixel is significant
output (sign[i]) send the sign of the pixel
mark[z] = MNP the pixel is now newly significant
else
mark[i] = MIP the pixel is insignificant
i = i + l move past this pixel
else
i = i + is skip[mark[z]] move past set/block

REFINEMENT PASS

i = 0, while z < I start the refinement pass

If mark[i] = MSP	significant pixel	mark[i] = MSP	significant pixel in
output (val[i] AND s)	refine the pixel	next plane	
i = i + l	move past the pixel	i = i + l	move past the pixel
Else if mark[i] = MNP	newly significant	else	
pixel		i = i + skip[mark[i]	move past set/block

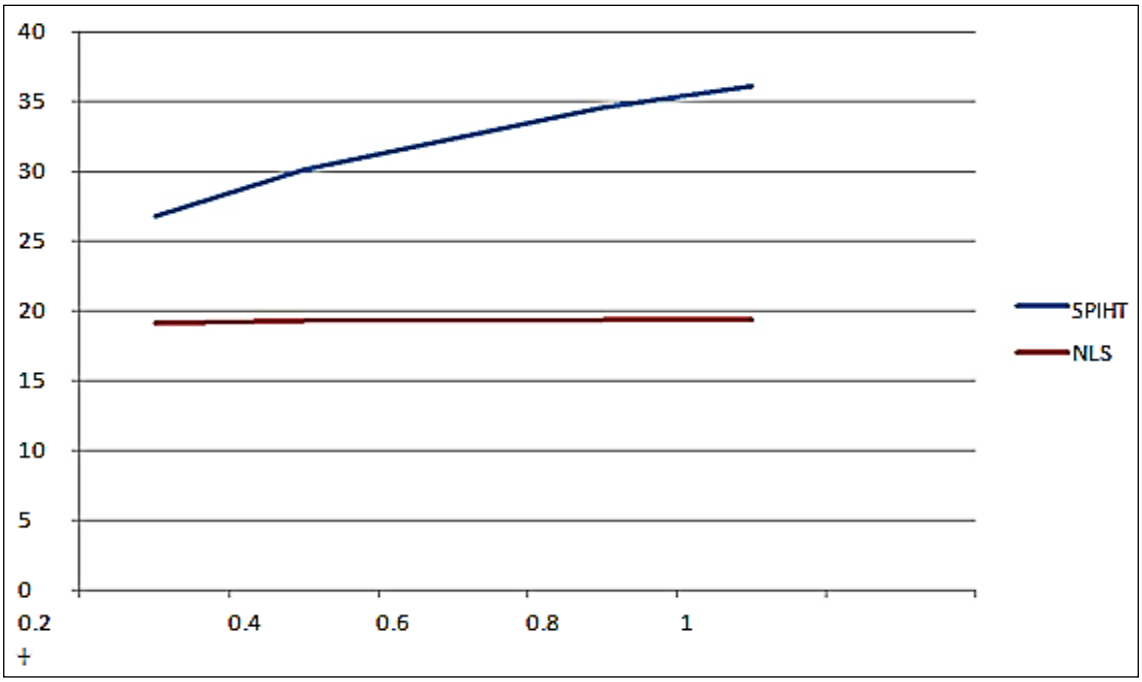

Fig. 2: *(PSNR vs. bpp).*

Table 1: *Coding Results for the 256 by 256 Grayscale Lena Image.*

SPIHT					**NLS**		
Encoding Time (sec) at bpp =2	**bpp**	**Decoding Time (sec)**	**PSNR**	**Encoding Time (sec) at bpp=2**	**bpp**	**Decoding time (sec)**	**PSNR**
19.725085	0.2	1.117604	26.8119	4.36214	0.2	0.007496	19.2697
17.175709	0.4	2.08771	30.1318	4.977572	0.4	0.012853	19.3424
17.488831	0.6	3.172648	32.4001	4.458534	0.6	0.019154	19.3744
16.807072	0.8	4.166350	34.6284	5.007638	0.8	0.022779	19.3990
17.044163	1.0	5.439317	36.1065	4.521988	1.0	0.07786	19.4074

RESULTS AND CONCLUSION

Coding results for the 256 by 256 grayscale lena image is plotted in Figure 2. The blue line is PSNR at different bit per pixel for (bpp) for SPIHT whereas red one is for NLS if we compare the encoding time of NLS and SPIHT at fix bpp we can see that NLS takes very less time than SPIHT and also in encoder when we encode image at different bpp then we can see that the encoding time of NLS is very less than SPIHT but in context of PSNR ratio (Table 1) SPIHT is more better than NLS.

REFERENCES

1. Tung CL, Chen TS, Wang WH, *et al.* A New Improvement of SPIHT Progressive Image Transmission, *IEEE 5th Int. Symposium Multimedia Software Eng*. 2003.
2. Zhu J, Lawson S. Improvements to SPIHT for Lossy Image Coding, *The 8th IEEE International Conference on Electronics, Circuits and Systems*, 2001.
3. A. Said, and W. Pearlman, “A New Fast and Efficient Image Codec Based on Set

Partitioning in Hierarchical trees", IEEE Trans. Circuits Syst. Video technology. vol. 6, no. 3, pp. 243-250. Jun. 1996.

4. Gopi ES. *Algorithm Collections for Digital Signal Processing using MATLAB*, Springer, 2007.
5. Minghe H, Cuixiang Z. Application of Improved SPIHT for Multispectral Image Compression, *5th Int. Conf. On Comp. Sci. &Edu*, Hefei, China. Aug. 24–27, 2010.
6. Jin Y, Lee HJ. A Block-Based Pass-Parallel SPIHT Algorithm, *IEEE T Circ Syst Video Tech*. Jul. 2012; 22(7): 1064–1075p.
7. Zhu L, Yang YM. Embedded Image Compression Using Differential Coding and Optimization Method, *7th Int. Conf. Wireless Comm. Net. And Mobile Computing*, 2011.
8. Khan E, Ghanbari M. Error Detection And Correction of Transmission Errors in SPIHT Coded Images, *IEEE International Conference on Image Processing*. 2002; 2: 689–692p.

Cite this Article
Shefa Hasnain, Sadiya Bakhshi, Shazia Asif *et al.* Analysis of image coders. *Journal of Artificial Intelligence Research & Advances*. 2016; 3(3): 26–30p.