Mehran University Research Journal of Engineering & Technology
Vol. 37, No. 4, 507-520 October 2018
p-ISSN: 0254-7821, e-ISSN: 2413-7219
DOI: 10.22581/muet1982.1804.06

# Comparison of Effects of Entropy Coding Schemes Cascaded with Set Partitioning in Hierarchical Trees

ALI IQBAL*, IMRAN TOUQIR*, ASIM ASHFAQ*, NATASHA KHAN* AND FAHIM ARIF**

RECEIVED ON 16.06.2017 ACCEPTED ON 13.11.2017

## ABSTRACT

**WT (Wavelet Transform) is considered as landmark for image compression because it represents a signal in terms of functions which are localized both in frequency and time domain. Wavelet sub-band coding exploits the self-similarity of pixels in images and arranges resulting coefficients in different sub-bands. A much simpler and fully embedded codec algorithm SPIHT (Set Partitioning in Hierarchical Trees) is widely used for the compression of wavelet transformed images. It encodes the transformed coefficients depending upon their significance comparative to the given threshold. Statistical analysis reveals that the output bit-stream of SPIHT comprises of long trail of zeroes that can be further compressed, therefore SPIHT is not advocated to be used as sole mean of compression. In this paper, wavelet transformed images have been initially compressed by using SPIHT technique and to attain more compression, the output bit streams of SPIHT are then fed to entropy encoders; Huffman and Arithmetic encoders, for further de-correlation. The comparison of two concatenations has been carried out by evaluating few factors like Bit Saving Capability, PSNR (Peak Signal to Noise Ratio), Compression Ratio and Elapsed Time. The experimental results of these cascading demonstrate that SPIHT combined with Arithmetic coding yields better compression ratio as compared to SPIHT cascaded with Huffman coding. Whereas, SPIHT once combined with Huffman coding is proved to be comparatively efficient.**

**Key Words: Wavelet Transform, Set Partitioning in Hierarchical Trees, Huffman and Arithmetic Coding.**

## 1. INTRODUCTION

Image compression plays a vital role in terms of saving storage space and reduction in transmission time. Wavelet Transform, a mathematical scheme developing rapidly in the field of image compression, is a successor of DCT (Discrete Cosine Transform) in which low frequencies are exploited efficiently and high frequencies are quantized coarsely [1]. In this technique translated and scaled version of particular wavelets are employed to decompose an image [2]. It is preferred because of its inherent property of being redundant and shift invariant [3].

Nowadays DWT (Discrete Wavelet Transformed) based various image compression techniques are being used to

Authors E-Mail: (ali.msee19@students.mcs.edu.pk, imrantqr@mcs.edu.pk, ur_sim@hotmail.com, nat.khan03@yahoo.com, fahim@mcs.edu.pk)
* Military College of Signals, National University of Science & Technology, Rawalpindi.
** Institute of Electrical, Electronic Engineers, Pakistan.

attain better PSNR and CR (Compression Ratio). One of the progressive encoding techniques to underline is EZW (Embedded Zero-Tree Wavelet) which was introduced by Jorome [4]. This is a very simple technique that generates an embedded bit stream as output [5]. In this scheme, higher energized coefficients are encoded first and encoding process can be terminated at any point when the distortion metric or a target rate is achieved [4].

SPIHT, an improved version of EZW, was proposed by Amir and William [6]. It is WT based image compression algorithm that generates an embedded bit stream, gives better PSNR and CRs for different types of gray scale images [7]. Although with SPIHT, better compression is achieved by searching more zero-trees and denoting them by separate tree root from the tree [8] yet memory requirements of SPIHT are large enough to handle [9-10]. It has also been observed that most of the values of wavelet coefficients are below the given threshold; hence output of SPIHT consists of a number of binary strings of zeros and ones that contains similarity and provides room for further compression [11]. This additional compression on the output stream of SPIHT can be achieved by using various types of entropy encodings.

In this paper output of SPIHT algorithm has been cascaded with two types of entropy encoders; Arithmetic and Huffman encoder, to evaluate their performance in terms of compression and efficiency. The paper has been organized in 6 sections in a way that Section-1introduces image compression using DWT and embedded encoding. Section-2 enunciates SPIHT algorithm in detail. Section 3 and 4 describe about the cascading of SPIHT with Arithmetic and Huffman coding respectively. This is followed by simulation and results in section-5. Section-6 concludes the paper.

## 2. SET PARTITIONING IN HIERARCHICAL TREES

SPIHT yields high PSNR than EZW because of a special symbol that indicates the significance of child nodes of significance parent, and separation of child nodes from second generation descendants [12-14]. After wavelet transformation of pictorial data, the decomposed image consists of sub-bands, where coefficients form a tree like structure. In Fig. 1 a SOT (Spatial Orientation Tree) is shown which exhibits the relationship between the sub-bands present in the pyramidal structure, where coefficients divided into four bands repeatedly. In this spatial orientation tree every node of the tree consists of coordinates values of pixel of the image. Every node further consists of either no or four branches and each branch depicts corresponding off-springs of same spatial location present in pyramid structure, represented by same color in Fig. 1. The peak level of the pyramidal structure comprises on LL band.

Only three sub-bands participate to establish the descendants i.e. only three pixels contribute to form off-springs in HH, HL and LH bands at similar spatial location. The remaining pixel out of these four is marked as star ‘*’, as shown in Fig. 1, and does not participate in establishing the descendants. Here we need to define a data structure based on sets:

H(x,y): Four Root coefficients in LL band at highest level of SSOT.

O(x,y): Set consisting offspring of coefficient at (x,y) i.e.

{(2x,2y), (2x,2y+1), 2x+1,2y),(2x+1,2y+1)}

D(x,y): Set comprises on descendants of location (x,y)

L(x,y): D(x,y) – O(x,y)

The rules of set partitioning technique are employed in SPIHT algorithm. This technique has outperformed EZW. Both encoding and decoding are achieved using SPIHT algorithm. This algorithm works more effectively, as it does not require ordering information in explicit manner for transmission like other progressive transmission algorithms. In this algorithm three lists are employed which are continuously updated by encoder and decoder in order to enhance compression ratio:

LIS (List of Insignificant Sets)

LIP (List of Insignificant Pixels)

LSP (List of Significant Pixels)

Every element of these sets is represented by coordinate values of the pixel (x,y). Elements in LSP and LIP comprise on coordinates of single pixel while in case of LIS element is set L(x,y) or the descendants D(x,y). SPIHT segregates the decomposed wavelets into significant and insignificant partitions based on the following functions:

$$S_n(Z) = \begin{cases} 1 & \max_{(i,j\in z)}\left\{\left|c_{i,j}\right|\right\} \geq 2^n \\ 0 & \text{Otherwise} \end{cases} \quad (1)$$

Here Sn (Z) is a set of significant coordinates Z, and ci, j is the coefficient value at coordinate (i,j) [4].

Initially every pixel consider as insignificant pixel with reference to calculated threshold. This algorithm comprises of different passes, given as follows:

- Sorting pass
- Refinement pass
- Updating of Quantization Step Pass

These three passes recur unless encoder continues to send least significant refinement bits. Insignificancy of the pixels of LIP is checked during the sorting pass. During the sorting pass, the pixels greater than the given threshold for this particular pass, are termed to be significant. These significant pixels are then passed to the LSP. Similarly, pixels of set present in LIS are checked, if it is significant then partition it and finally remove from this list. Those subsets which have more than one entry are put in LIS. Similarly, single pixels are put in the LSP or LIP, according to the significance. In magnitude refinement pass, the significant pixels of LSP are encoded for most significant '$j^{th}$' position. After the above discussion, we can summarize the encoding algorithm as follows:

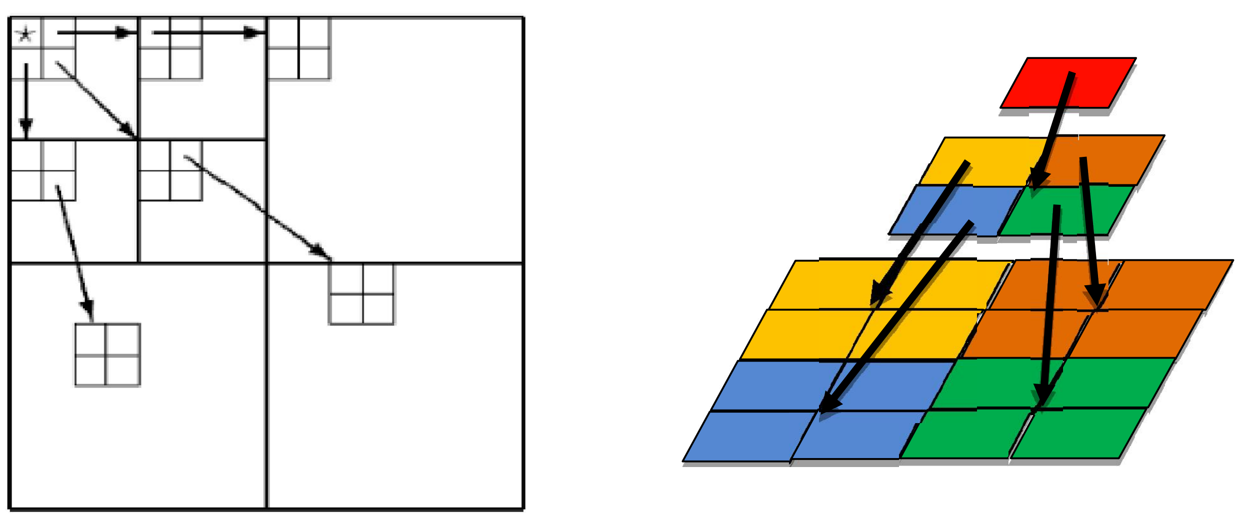

*FIG. 1. SPACIAL ORIENTATION TREEAND PYRAMIDAL STRUCTURE*

## 2.1 Initialization

Output $j = \lfloor \log_2 (\max_{(x,y)} \{| c_{x,y} |\}) \rfloor$

LSP = { }

LIP = {(x,y)∈H}

LIS = {D(x,y),(x,y)∈H}

## 2.2 Sorting Pass

Verify for significance of each entry of the LIP. Transmit '1' if the entry is found significant as compared to threshold and transmit '0' if insignificant. Remove the entry from the LIP if it is found significant and insert it in the LSP. Then verify for significance of each entry of the LIS with respect to threshold. If the entry is found significant, transmit its sign of significance, partition this entry depending upon the set if it is L(x,y) or D(x,y). Continue updating all three lists LSP, LIS and LIP relevant to their significance.

## 2.3 Refinement Pass

For all the entries present in the LSP, Transmit MSB found at $i^{th}$ location.

## 2.4 Renewing Quantization Step Pass

Decrease values of 'j' by 1 during this pass and repeat steps of sorting pass, refinement pass and quantization steps pass of the algorithm. Keep repeating unless 'j' becomes zero. At the decoder end same procedure is repeated in reverse order. The output of encoder is fed to the decoder. By employing further entropy coding for the output of the SPIHT encoder more compression is achieved.

## 2.5 Analysis of SPIHT

We illustrate the concept of this algorithm with the help of this example given in Tables 1-2.

**TABLE 1. THE ORIGINAL MATRIX**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 34.2329 | 22.9106 | 8.0819 | -9.5783 | 2.4702 | 9.6024 | 17.4720 | 20.9260 |
| 3.1444 | 0.0473 | -10.7578 | 5.7983 | 15.2621 | 5.7212 | -6.8773 | -26.2526 |
| -9.4979 | -7.2971 | 8.1126 | 10.0352 | 10.4049 | 3.1472 | -12.044 | -15.2028 |
| 6-7.7991 | 1.9334 | 12.7445 | 13.1993 | 11.3390 | 6.9783 | 4.1331 | 5.5305 |
| 12.7476 | 13.6053 | 24.2530 | 24.4590 | 21.2853 | 16.8028 | 13.9673 | 14.3350 |
| 0.4514 | 7.7005 | 16.9633 | 23.2157 | 20.2790 | 16.1249 | 13.9673 | 14.3350 |
| 7.9368 | -4.9201 | 2.2329 | 0.4116 | 29.5710 | 22.8518 | 22.6126 | 6.1349 |

**TABLE 2. DWT COEFFICIENTS OF ARBITRARY DATA SET**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 34.2329 | 22.9106 | 8.0819 | -9.5783 | 2.4702 | 9.6024 | 17.4720 | 20.9260 |
| 3.1444 | 0.0473 | -10.7578 | 5.7983 | 15.2621 | 5.7212 | -6.8773 | -26.2526 |
| -9.4979 | -7.2971 | 8.1126 | 10.0352 | 10.4049 | 3.1472 | -12.044 | -15.2028 |
| 6-7.7991 | 1.9334 | 12.7445 | 13.1993 | 11.3390 | 6.9783 | 4.1331 | 5.5305 |
| 12.7476 | 13.6053 | 24.2530 | 24.4590 | 21.2853 | 16.8028 | 13.9673 | 14.3350 |
| 0.4514 | 7.7005 | 16.9633 | 23.2157 | 20.2790 | 16.1249 | 13.9673 | 14.3350 |
| 7.9368 | -4.9201 | 2.2329 | 0.4116 | 29.5710 | 22.8518 | 22.6126 | 6.1349 |

The encoder transmits the bit-stream after one pass and then the decoder decodes it. For Threshold =2n and n=5 (according to the previously stated explanation). So the output bit stream is '11100011100010000001010110000'. It is evident from result that there is large number of series of '0' situation. Statistical analysis reveals that in most digital image that '000' comes with the probability larger than ¼. At this stage further compression can be achieved by concatenating Entropy coding schemes with SPIHT.

## 3. CASCADING WITH ARITHMETIC CODING

To understand Arithmetic coding following concept should be kept in mind:

In this algorithm variable length of Symbols are encoded using variable length code-words.

A single Arithmetic code-word is allocated to all the symbols in the message jointly.

The symbol and code-word has no one to one correspondence.

The code-word exist from the class of real numbers [0,1] with one open interval. As the coding progresses this interval becomes smaller and smaller.

A flowchart in Fig. 2 shows how algorithm runs. To perform cascading initially the output bit-stream of the SPIHT is first symbolized i.e. all bit stream is divided in sets of three bits forming $2^3$ symbols given as 000, 001, 010, 011, 100, 101, 110 and 111. Then the probabilities of these symbols are calculated. These symbols are then fed to the Arithmetic coder block, which generate an arithmetic code-word.

During the symbolization of the SPIHT output bit-stream there remains one, two or three bits. The information of these remaining bits is added to the Bit Header as shown in Table 3.

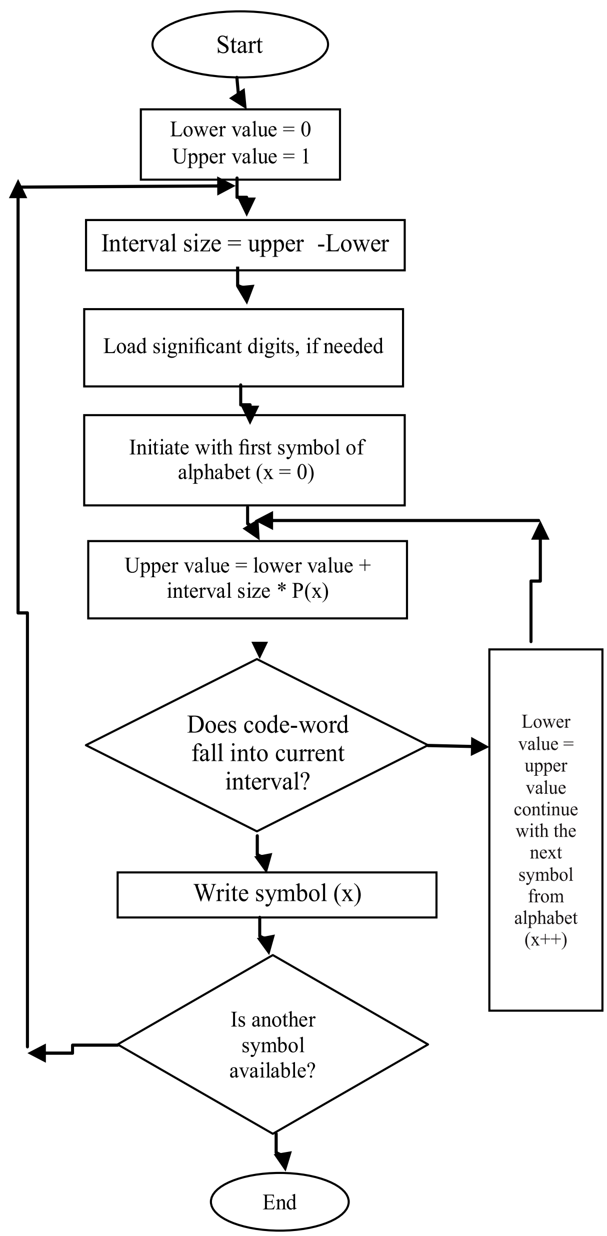


*FIG. 2. FLOWCHART FOR ARITHMETIC CODING*

**TABLE 3. BIT HEADER OF CASCADING SPIHT AND ENTROPY CODING**

| Two Bits Representation of number of Reaming Bits | Entropy Coding Generated Bit-Stream | Remaining Bits |
|---|---|---|

# 4. CASCADING WITH HUFFMAN CODING

Huffman coding is dependent upon the frequency of appearance of pixels in an image. It has the following features:

In this algorithm variable length code-words allot fixed length of symbols [15].

This technique decode symbols uniquely i.e. no code word contains the prefix of previous code-word. Fig. 3 represents the flow of algorithm.

Cascading of SPIHT with Huffman coding is done in the same way as was done with Arithmetic coding. Initially Symbolization is done on SPIHT output bit-stream by using a combination of three bits each and then that symbol is further given to Huffman Coder block. Table 4 is drawn for the Lena Image of 512x512 at 0.5 bpp. That shows the symbols with their probabilities and allotted code-words. This cascading results with the same bit header as discussed in Table 3.

# 5. SIMULATION AND RESULTS

After the successful cascading of both entropy coding techniques with SPHT some measurements are

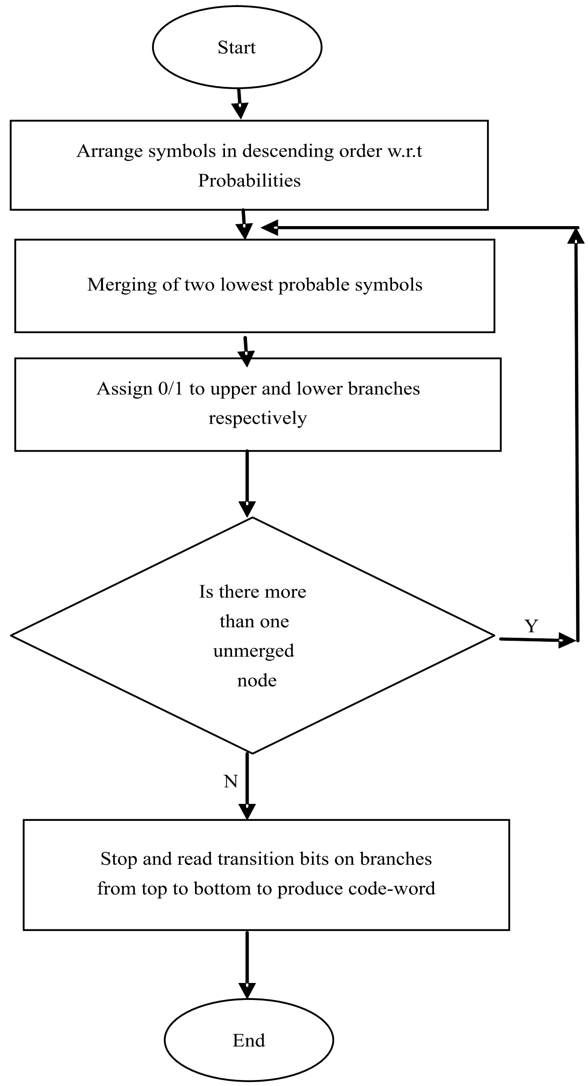


*FIG. 3. FLOWCHART OF HUFFMAN CODING*

**TABLE 4. SYMBOLS WITH PROBABILITIES AND CODE-WORDS**

| Symbols | Probabilities | Code Words |
|---|---|---|
| 000 | 0.24 | 00 |
| 001 | 0.14 | 100 |
| 010 | 0.12 | 101 |
| 011 | 0.1 | 110 |
| 100 | 0.15 | 111 |
| 101 | 0.08 | 0011 |
| 110 | 0.1 | 010 |
| 111 | 0.07 | 0110 |

evaluated in order to compare the performance of both cascading. In order to carry out calculation Lena image is used at different resolutions i.e. 256x256 and 512x512.

Tables 5-8 enunciate that bit saving capability and compression ratio, after cascading SPIHT with Arithmetic encoding, are improving gradually with the increase in bit rates for two different resolution of Lena image as compared to those of SPIHT cascaded with Huffman. Whereas results of Table 5 exhibit that SPIHT concatenation with Huffman encoding is more time efficient. The results are also demonstrated with the help of Figs 4-7.

**TABLE 5. BITS SAVING CAPABILITY OF BOTH ALGORITHMS**

| Bits Saving Capability | | | | |
|---|---|---|---|---|
| Rate | Lena 256x256 | | Lena 512x512 | |
| | SPIHT with Huffman | SPIHT with Arithmetic | SPIHT with Huffman | SPIHT with Arithmetic |
| 0.1 | 296 | 332 | 1149 | 1290 |
| 0.2 | 558 | 648 | 1888 | 2154 |
| 0.3 | 700 | 819 | 2908 | 3284 |
| 0.4 | 993 | 1186 | 3358 | 3838 |
| 0.5 | 1233 | 1437 | 4153 | 4810 |
| 0.6 | 1282 | 1527 | 4359 | 5150 |
| 0.7 | 1522 | 1816 | 3716 | 4851 |
| 0.8 | 1658 | 1989 | 4167 | 5570 |
| 0.9 | 1820 | 2156 | 5093 | 6676 |
| 1.0 | 1849 | 2218 | 5520 | 7224 |

**TABLE 6. PSNR PERFORMANCES OF BOTH ALGORITHMS**

| PSNR Performance (Bpp) | | | | |
|---|---|---|---|---|
| Rate | Lena 256x256 | | Lena 512x512 | |
| | SPIHT with Huffman | SPIHT with Arithmetic | SPIHT with Huffman | SPIHT with Arithmetic |
| 0.1 | 24.45 | 24.45 | 28.13 | 28.13 |
| 0.2 | 27.11 | 27.11 | 31.02 | 31.02 |
| 0.3 | 28.73 | 28.73 | 32.95 | 32.95 |
| 0.4 | 30.44 | 30.44 | 34.30 | 34.30 |
| 0.5 | 31.71 | 31.71 | 35.56 | 35.56 |
| 0.6 | 32.66 | 32.66 | 36.53 | 36.53 |
| 0.7 | 33.76 | 33.76 | 37.36 | 37.36 |
| 0.8 | 34.86 | 34.86 | 38.15 | 38.15 |
| 0.9 | 35.74 | 35.74 | 38.83 | 38.83 |
| 1.0 | 36.47 | 36.47 | 39.65 | 39.65 |

Application of both cascading algorithms on different types of low and high contrast images have also substantiated above the drawn inferences. Tables 9-10 contain the results of two test images, 128x128 Lena and 256x256 Obama, of both high and low contrasts. The original and reconstructed images are given at Figs. 8-9.

Embedded coding requires least multiplication and division with focus on comparison, thus it brings efficiency. The complexity is proportional to the number of compressed bits. It is better in complexity as compared to segmentation based image coding, therefore widely adopted. In order to verify the concluding statement, a

**TABLE 7. COMPRESSION RATIOS OF BOTH ALGORITHMS**

| Compression Ratio | | | | |
|---|---|---|---|---|
| Rate | Lena 256x256 | | Lena 512x512 | |
| | SPIHT with Huffman | SPIHT with Arithmetic | SPIHT with Huffman | SPIHT with Arithmetic |
| 0.1 | 1.0473 | 1.0534 | 1.0458 | 1.0518 |
| 0.2 | 1.0445 | 1.0520 | 1.0374 | 1.0428 |
| 0.3 | 1.0369 | 1.0435 | 1.0384 | 1.0436 |
| 0.4 | 1.0394 | 1.0474 | 1.0331 | 1.0380 |
| 0.5 | 1.0391 | 1.0459 | 1.0327 | 1.0381 |
| 0.6 | 1.0337 | 1.0404 | 1.0285 | 1.0339 |
| 0.7 | 1.0343 | 1.0412 | 1.0207 | 1.0272 |
| 0.8 | 1.0327 | 1.0394 | 1.0203 | 1.0273 |
| 0.9 | 1.0318 | 1.0379 | 1.0221 | 1.0291 |
| 1.0 | 1.0290 | 1.0350 | 1.0215 | 1.0283 |

**TABLE 8. ELAPSED TIME FOR BOTH ALGORITHMS**

| Elapsed Time (Sec) | | | | |
|---|---|---|---|---|
| Rate | Lena 256x256 | | Lena 512x512 | |
| | SPIHT with Huffman | SPIHT with Arithmetic | SPIHT with Huffman | SPIHT with Arithmetic |
| 0.1 | 1.7089 | 3.3623 | 7.7691 | 9.0000 |
| 0.2 | 4.0136 | 8.4732 | 14.9828 | 21.2069 |
| 0.3 | 5.1116 | 6.5069 | 22.7861 | 31.4346 |
| 0.4 | 6.2057 | 7.4415 | 36.1099 | 41.3791 |
| 0.5 | 7.2749 | 10.7266 | 46.4404 | 51.4574 |
| 0.6 | 10.3345 | 12.1818 | 64.1089 | 74.2677 |
| 0.7 | 12.0107 | 14.2440 | 72.5772 | 89.4216 |
| 0.8 | 13.6070 | 16.9060 | 107.5733 | 106.6014 |
| 0.9 | 16.6915 | 18.8235 | 124.0462 | 120.7071 |
| 1.0 | 18.7386 | 24.2221 | 148.4652 | 151.2966 |

lot of statistical analyses have been carried out by taking a number of images of various sizes and resolutions. Results of the few images like Obama, Baboon, Pepper, bird and couple have been displayed in the ensuing Tables 11-14 and Fig. 10.

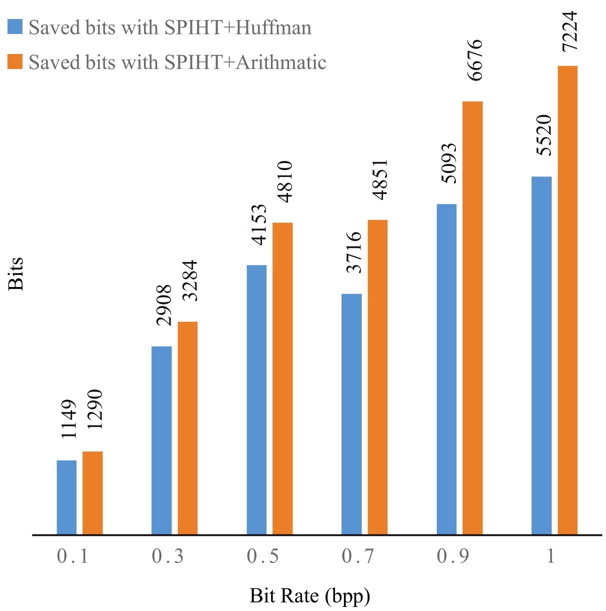


*FIG. 4. HISTOGRAM FOR BITS SAVING CAPABILITY AT GIVEN RATES FOR LENA 512*

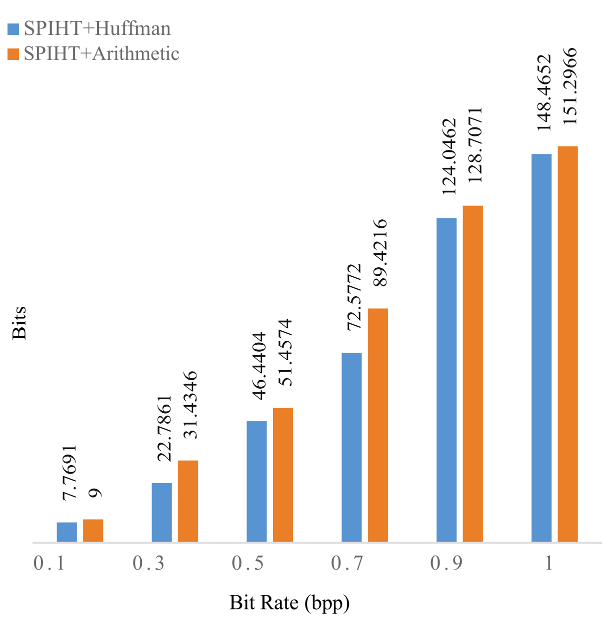


*FIG. 5. HISTOGRAM FOR ELAPSED TIME AT GIVEN RATES FOR LENA 512*

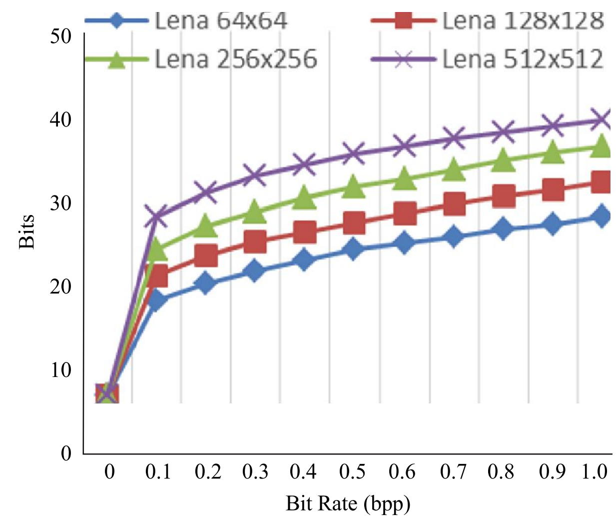


*FIG. 6. PSNR PERFORMANCE FOR BOTH CASCADING AT GIVEN RESOLUTION*

*FIG. 7(a). ORIGINAL*

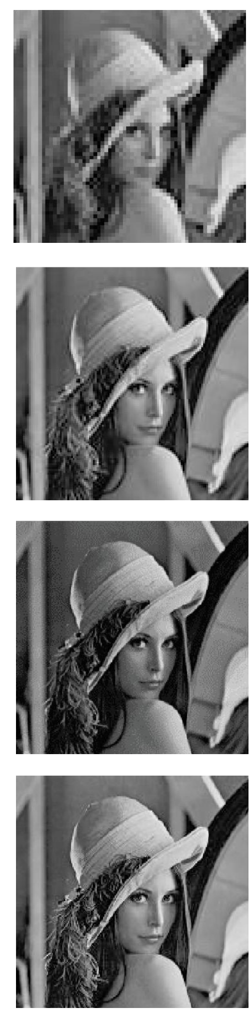

*FIG. 7(b). RECONSTRUCTED IMAGES OF LENA AT VARIOUS RESOLUTIONS 64X64, 128X128, 256X256 AND 512X512*

**TABLE 9. ELAPSED TIME FOR BOTH ALGORITHMS FOR LOW AND HIGH CONTRAST IMAGES**

| Elapsed Time (Sec) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Rate | Low Contrast | | | | High Contrast | | | |
| | Lena128x128 | | Obama 256x256 | | Lena128x128 | | Obama 256x256 | |
| | SPIHT with Huffman | SPIHT with Arithmetic | SPIHT with Huffman | SPIHT with Arithmetic | SPIHT with Huffman | SPIHT with Arithmetic | SPIHT with Huffman | SPIHT with Arithmetic |
| 0.1 | 1.9297 | 3.3432 | 4.0748 | 4.8686 | 1.7847 | 2.0522 | 4.3770 | 4.8594 |
| 0.2 | 1.8107 | 2.4664 | 6.3330 | 7.8396 | 1.6868 | 2.1747 | 6.4340 | 7.6480 |
| 0.3 | 2.4242 | 2.9887 | 9.4621 | 11.8295 | 2.4661 | 3.0754 | 9.3433 | 11.0217 |
| 0.4 | 3.1893 | 3.6106 | 12.1901 | 15.1210 | 3.2125 | 3.8484 | 12.4466 | 14.9772 |
| 0.5 | 3.7576 | 5.8504 | 15.7590 | 18.9981 | 3.8118 | 4.6898 | 15.5798 | 18.4035 |
| 0.6 | 4.6173 | 5.9920 | 21.2084 | 22.7926 | 4.4995 | 5.4369 | 18.3400 | 21.6120 |
| 0.7 | 5.2586 | 6.7626 | 22.1881 | 39.3621 | 5.6181 | 6.3819 | 22.0686 | 27.1575 |
| 0.8 | 5.8512 | 7.5879 | 26.1461 | 32.3699 | 6.4207 | 7.2186 | 25.1197 | 31.5276 |
| 0.9 | 7.0841 | 8.7920 | 32.2327 | 35.0577 | 6.3065 | 7.9643 | 27.9584 | 33.0500 |
| 1.0 | 7.6785 | 9.0754 | 32.4269 | 36.5067 | 7.5575 | 8.7126 | 31.2239 | 36.6862 |

**TABLE 10. BIT SAVING CAPABILITY FOR BOTH ALGORITHMS FOR LOW AND HIGH CONTRAST IMAGES**

| Bits Saving Capability | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Rate | Low Contrast | | | | High Contrast | | | |
| | Lena128x128 | | Obama 256x256 | | Lena128x128 | | Obama 256x256 | |
| | SPIHT with Huffman | SPIHT with Arithmetic | SPIHT with Huffman | SPIHT with Arithmetic | SPIHT with Huffman | SPIHT with Arithmetic | SPIHT with Huffman | SPIHT with Arithmetic |
| 0.1 | 78 | 80 | 358 | 405 | 65 | 73 | 224 | 291 |
| 0.2 | 104 | 115 | 629 | 734 | 143 | 165 | 599 | 711 |
| 0.3 | 205 | 220 | 967 | 1100 | 189 | 204 | 834 | 1011 |
| 0.4 | 270 | 300 | 1190 | 1379 | 262 | 299 | 1311 | 1550 |
| 0.5 | 281 | 315 | 1384 | 1599 | 343 | 380 | 1555 | 1826 |
| 0.6 | 331 | 386 | 1636 | 1895 | 410 | 455 | 1765 | 2087 |
| 0.7 | 388 | 453 | 1990 | 2287 | 463 | 514 | 2089 | 2476 |
| 0.8 | 484 | 555 | 2103 | 2417 | 540 | 598 | 2288 | 2707 |
| 0.9 | 489 | 579 | 2300 | 2662 | 588 | 652 | 2382 | 2817 |
| 1.0 | 510 | 615 | 2420 | 2814 | 679 | 748 | 2874 | 3326 |

*FIG. 8(a). ORIGINAL*

*FIG. 8(b). RECONSTRUCTED IMAGES OF LENA 128X128 AND OBAMA 256X256 (HIGH CONTRAST)*

*FIG. 9(a). ORIGINAL*

*FIG. 9(b). RECONSTRUCTED IMAGES OF LENA 128X128 AND OBAMA 256X256 (LOW CONTRAST)*

**TABLE 11. BITS SAVING CAPABILITY FOR BOTH ALGORITHMS FOR OTHER IMAGES**

| Rate | Bits Saving Capability | | | | | |
|---|---|---|---|---|---|---|
| | Obama 256x256 | | Baboon 256x256 | | Pepper256x256 | |
| | SPIHT with Huffman | SPIHT with Arithmetic | SPIHT with Huffman | SPIHT with Arithmetic | SPIHT with Huffman | SPIHT with Arithmetic |
| 0.1 | 363 | 418 | 289 | 345 | 308 | 329 |
| 0.2 | 579 | 678 | 441 | 604 | 659 | 735 |
| 0.3 | 918 | 1065 | 734 | 980 | 851 | 970 |
| 0.4 | 1084 | 1305 | 985 | 1219 | 1130 | 1307 |
| 0.5 | 1324 | 1557 | 1152 | 1408 | 1455 | 1654 |
| 0.6 | 1611 | 1884 | 977 | 1311 | 1804 | 2052 |
| 0.7 | 1819 | 2154 | 768 | 1249 | 2000 | 2308 |
| 0.8 | 2051 | 2424 | 859 | 1431 | 2277 | 2618 |
| 0.9 | 2303 | 2710 | 918 | 1590 | 2341 | 2692 |
| 1.0 | 2403 | 2850 | 1497 | 2155 | 2538 | 2928 |

**TABLE 12. BITS SAVING CAPABILITY FOR BOTH ALGORITHMS FOR OTHER IMAGES**

| Bits Saving Capability | | | |
|---|---|---|---|
| Bird 256x256 | | Couple 256x256 | |
| SPIHT with Huffman | SPIHT with Arithmetic | SPIHT with Huffman | SPIHT with Arithmetic |
| 265 | 297 | 235 | 291 |
| 496 | 552 | 525 | 639 |
| 757 | 843 | 605 | 739 |
| 1091 | 1214 | 668 | 876 |
| 1286 | 1434 | 924 | 1174 |
| 1498 | 1696 | 1174 | 1449 |
| 1594 | 1817 | 1171 | 1501 |
| 1452 | 1727 | 1188 | 1584 |
| 1599 | 1901 | 1384 | 1824 |
| 2018 | 2362 | 1681 | 2162 |

**TABLE 13. ELAPSED TIME FOR BOTH ALGORITHMS FOR OTHER IMAGES**

| Rate | Elapsed Time (Sec) | | | | | |
|---|---|---|---|---|---|---|
| | Obama 256x256 | | Baboon 256x256 | | Pepper256x256 | |
| | SPIHT with Huffman | SPIHT with Arithmetic | SPIHT with Huffman | SPIHT with Arithmetic | SPIHT with Huffman | SPIHT with Arithmetic |
| 0.1 | 4.1459 | 4.9219 | 4.8016 | 5.5692 | 4.1266 | 5.0121 |
| 0.2 | 8.1190 | 10.9493 | 7.5964 | 8.2429 | 6.2027 | 8.9535 |
| 0.3 | 9.8259 | 24.9735 | 10.4751 | 11.9745 | 9.4517 | 16.9925 |
| 0.4 | 14.1936 | 28.3171 | 13.3187 | 15.3194 | 15.2937 | 18.9351 |
| 0.5 | 15.8551 | 25.7068 | 17.0699 | 18.8527 | 15.4710 | 25.8023 |
| 0.6 | 18.3362 | 28.1613 | 19.5157 | 24.2435 | 19.7451 | 25.5108 |
| 0.7 | 23.6568 | 36.6132 | 23.9589 | 29.6304 | 21.6035 | 30.0418 |
| 0.8 | 25.0496 | 37.7163 | 27.9735 | 32.9852 | 24.5320 | 35.8612 |
| 0.9 | 28.2171 | 55.0322 | 32.6721 | 41.7283 | 31.5261 | 35.6338 |
| 1.0 | 30.6170 | 44.2331 | 35.0099 | 40.9646 | 34.3293 | 33.3989 |

**TABLE 14. ELAPSED TIME FOR BOTH ALGORITHMS FOR OTHER IMAGES**

| Elapsed Time | | | |
|---|---|---|---|
| Bird 256x256 | | Couple 256x256 | |
| SPIHT with Huffman | SPIHT with Arithmetic | SPIHT with Huffman | SPIHT with Arithmetic |
| 4.1948 | 6.2311 | 4.8834 | 5.0736 |
| 6.4673 | 8.0934 | 6.6779 | 8.1793 |
| 9.5165 | 14.0227 | 9.6772 | 11.6693 |
| 13.2360 | 18.3097 | 13.0355 | 16.4412 |
| 18.3087 | 23.9866 | 16.4284 | 19.6296 |
| 18.9635 | 27.5071 | 19.2215 | 23.1156 |
| 21.3482 | 30.9340 | 22.8776 | 27.8283 |
| 26.3532 | 38.5624 | 27.5107 | 34.0176 |
| 29.4676 | 42.2578 | 31.0229 | 36.5118 |
| 34.0882 | 43.3389 | 34.9890 | 40.0505 |

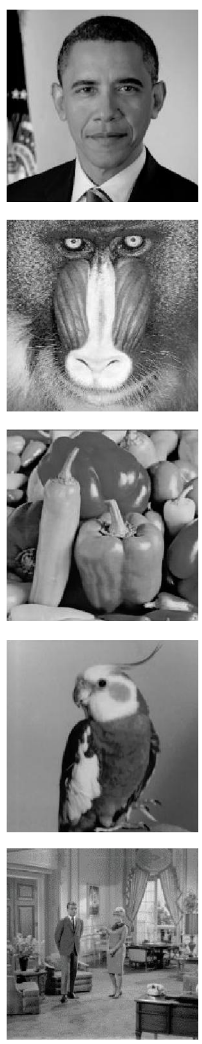

*FIG. 10(a). ORIGINAL*

*FIG. 10(b). RECONSTRUCTED IMAGES OF COUPLE, BIRD, OBAMA, PEPPER AND BABOON*

# 6. CONCLUSION

In this paper SPIHT technique has been combined with two schemes of entropy encoding for sparse representation of the image. It has been observed that SPIHT technique once cascaded with Arithmetic encoding performs better in terms of bits saving capability and CR but at the cost of computational complexity, as compared to SPIHT cascaded with Huffman encoding which is computationally efficient. Here, PSNR performance is preserved for both algorithms however, it increases with the increase in resolution of the image on a given rate in bpp.

# ACKNOWLEDGEMENTS

Authors take this opportunity to extend their profound gratitude to Military College of Signals, National University of Sciences & Technology, Rawalpindi, Pakistan, for research funding and anonymous Reviewers/Experts, for their valuable comments and feedback.

# REFERENCES

[1] Jandge, K., and Raja, R., "Image Compression Based on Discrete Wavelet and Lifting Wavelet Transform Technique", International Journal of Science, Engineering and Technology Research, Volume 3, No. 3, 2014.

[2] Antonini, M., Barlaud, M., and Pierre, M., "Image Coding using Wavelet Transform", IEEE Transactions on Image Processing, Volume 1, No. 2, pp. 205-220,1992.

[3] Jain, S., and Rani, J., "Image Enhancement Using Wavelets", International Journal of Innovative Research in Computer and Communication Engineering, Volume 2, No. 5, pp. 4131-4136,2014.

[4] Shapiro, J.M., "Embedded Image Coding using Zero-Tree of Wavelets Coefficients", IEEE Transactions on Signal Processing, Volume 41, No. 12, pp. 3445-3462, 1993.

[5] Shi, Y.Q., and Sun, H., "Image and Video Compression for Multimedia Engineering Fundamentals, Algorithms and Standards", Chapter-8, pp. 176-178, CRC Press, New York, 2011.

[6] Said, A., and Pearlman, W.A., "A New Fast and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees", IEEE Transactions on Circuits and Systems for Video Technology, Volume 6, No. 3, pp. 243-250, 1996.

[7] Dave, J., and Jain, P., "Study and Implementation of SPIHT and Modified SPIHT Algorithm for Image Compression", Volume 5, No.1, pp. 69-72, 2017.

[8] Kaur, C., and Budhiraja, S., "Improvements of SPIHT in Image Compression- Survey", International Journal of Emerging Technology & Advanced Engineering, Volume 3, No. 1, pp. 652-656, 2013.

[9] Kiran, R.H.K., and Paramesha, "Design and Implementation of HAAR Wavelet Transform and Stripe Logic Based Modified SPIHT Algorithm for Medical Imaging", International Journal of Engineering Development and Research, Volume 5, No. 1, pp. 347-356, 2017.

[10] Ahir, T., and Satyanarayana, R.V.S., "Contemporary Affirmation of SPIHT Improvements in Image Coding", Global Journal of Computer Science & Technology Graphics and Vision, Volume 13, No. 9, pp. 26-34, 2013,

[11] Li,W., Pang, Z.P., and Liu, Z.J.,"SPIHT Algorithm Combined with Huffman Encoding", IEEE Transaction on Intelligent Information Technology & Security Informatics, pp. 341-343, China, 2010.

[12] Basavanthaswami, V., and Somasekhar, T., "Image Compression using SPIHT", International Journal of Innovative Research in Computer & Communication Engineering, Volume 5, No. 2, pp. 1839-1844, 2017.

[13] Ghanbari, M., "Standard Codecs Image Compression to Advanced Video Coding", British Library, London Institute of Information &Technology, Chapter-4, pp. 80-97, 2011.

[14] Dodla, S., Raju, Y.D.S., and Mohan, K.V.M., "Image Compression using Wavelet and SPIHT Encoding Scheme", International Journal of Engineering Trends & Technology, Volume 4, No. 9, pp. 3863-3865, 2013.

[15] Sirikanth, S., and Meher, S., "Compression Efficiency for Combining Different Embedded Image Compression Techniques with Huffman Encoding", International Conference on Communication and Signal Processing, pp. 816-820, India, 2013.