I.J. Image, Graphics and Signal Processing, **2025, 6, 169-181**
Published Online on December 8, 2025 by MECS Press (http://www.mecs-press.org/)
DOI: 10.5815/ijigsp.2025.06.10

A Fast Output Generating Set Partitioning in Hierarchical Trees Coding for Medical Image Compression

Narayana Prakash S.*
Mangalore University/Department of Studies and Research in Electronics, Mangalore, 574199, India
E-mail: narayanap6@gmail.com
ORCID iD: https://orcid.org/0009-0005-5902-1070
*Corresponding Author

Airani Mohammad Khan
Mangalore University/Department of Studies and Research in Electronics, Mangalore, 574199, India
E-mail: asifabc@gmail.com
ORCID iD: https://orcid.org/0000-0001-8690-856X

Received: 15 December, 2024; Revised: 03 June, 2025; Accepted: 18 July, 2025; Published: 08 December, 2025

Abstract: In this paper, we have presented Discrete Wavelet Transform (DWT) based Fast Output Generating Set Partitioning in Hierarchical Trees (FOGSPIHT) algorithm for MRI brain image compression. The FOGSPIHT is scalable, faster, and robust algorithm. Image compression is an important technique that enables fast and high throughput imaging applications by reducing the storage space or transmission bandwidth. DWT transforms the image to get a set of coefficients that are used for efficient compression. The Set Partitioning In Hierarchical Trees (SPIHT) algorithm is an efficient algorithm used for DWT based image compression. The limitations of SPIHT coding are the complexity and memory requirements. To reduce the complexity, we propose the FOGSPIHT algorithm that works on the basic principles of SPIHT. The FOGSPIHT algorithm works on coefficients that are converted to bit planes. FOGSPIHT eliminates the comparison operations in the compression process of SPIHT by simple logical operations on bits. The values of Mean Square Error (MSE), Peak Signal to Noise Ratio (PSNR), and Structural Similarity Index Measure (SSIM) are calculated and plotted against Compression Ratio (CR). The result obtained with the FOGSPIHT algorithm is equal to or better than the SPIHT algorithm. The FOGSPIHT algorithm is faster which has reduced encoding and decoding time. The implementation of the FOGSPIHT algorithm with an 8x8 image DWT coefficient on FPGA requires the lower amount of resource and power requirements in comparison with the SPIHT algorithm.

Index Terms: DWT, Wavelet, Medical Image Compression, SPIHT, FOGSPIHT

1. Introduction

Digital image compression is the process of reducing the redundancies in digital images to store or transmit them efficiently [1]. Image in medicine plays an important role by providing visual representations of the internal organs of the human body [2]. Based on the reconstructed image from the compressed data, the image compression is classified as lossy and lossless [3]. Lossless image compression reproduces the original image visually and mathematically, but to obtain a large amount of compression, lossy techniques are preferred over lossless techniques [4]. The majority of image compression is performed with transform based lossy method [5, 6]. Discrete Wavelet Transform (DWT) is one of the transforms used in the process of image compression [7]. DWT provides high energy compaction and decorrelation compared with other transforms [8]. Another advantage of DWT is the provision of simultaneous space and frequency representation, which helps to achieve efficient compression [9]. The algorithms used to compress the DWT coefficients are Embedded Zerotree Wavelet (EZW), Set Partitioning In Hierarchical Trees (SPIHT), and Embedded Block Coding with Optimized Truncation (EBCOT) [10, 11, 12].

The SPIHT algorithm by A. Said and W. A. Pearlman is an efficient, embedded, progressive bitstream generation algorithm for image compression [11]. It uses a hierarchical tree structure for the compression process that uses both spatial and frequency domain properties of the image by making efficient use of the DWT. The SPIHT algorithm provides better compression in terms of Mean Square Error (MSE), Peak Signal to Noise Ratio (PSNR), and Structural

Similarity Index Measure (SSIM). Even though the compression is slightly less than EBCOT, SPIHT implementation on hardware is easier, which gained attraction towards SPIHT for hardware implementations [13].

Even though SPIHT has the advantage of becoming friendly for hardware implementation, it has limiting factors, such as memory requirement and complexity [14]. Various modifications were proposed in the literature to improve the performance of the SPIHT algorithm. Frederick W. Wheeler et. al. proposed SPIHT with no list [15]. Ahmed Abu-Hajar et. al. proposed a variant of the SPIHT algorithm which works with different options to provide better results [16]. Nikola Sprljan et. al. combined wavelet packet decomposition with the SPIHT algorithm to get better results than traditional SPIHT [17]. Yongseok Jin et. al. proposed a block-based SPIHT algorithm to overcome dynamic processing order [18]. Humberto de J. Ochoa Dom ńguez et. al. proposed a modified SPIHT algorithm for the reduction of comparison operations, thereby reducing execution time in comparison with the original SPIHT algorithm. The threshold for each subband is calculated after applying DWT [19].

The various modifications the SPIHT algorithm gives better performance either in performance using additional complexity or they provide efficient compression with a small compromise in performance. To handle this, we are using the SPIHT algorithm as the reference for this work. The SPIHT algorithm is a bitplane coding technique, but it does not completely explore the bitplane coding capability. To overcome this, we presented the Fast Output Generating Set Partitioning in Hierarchical Trees Coding (FOGSPIHT) algorithm, which works on the bitplanes. The wavelet transformed images are compressed with both the SPIHT algorithm and FOGSPIHT algorithm separately and the results obtained in both cases are compared in terms of MSE, PSNR, SSIM, and encoding and decoding time. The time and space complexities of the FOGSPIHT are analyzed. An image of 8x8 is compressed using the FOGSPIHT algorithm on an FPGA and compared with the SPIHT algorithm to highlight the advantage of FOGSPIHT over SPIHT.

2. Image Compression System

The simplified block diagram of the wavelet-based compression system is shown in Fig. 1. The image in the spatial domain is subjected to discrete wavelet transformation, quantization/thresholding and encoding, and entropy encoding to get the compressed bitstream [20]. In the decompression, the decoding operations are performed, followed by the inverse transformation to get the reconstructed image. The parameters used to analyze the performance of compression are CR, MSE, PSNR, and SSIM calculated using the expression (1), (2), (3), and (4) [21].

$$\mathrm{CR} = \frac{\text{Size of the original image}}{\text{Size of compreesed image}} \tag{1}$$

$$MSE = \frac{1}{NM}\sum_{0}^{N-1}\sum_{0}^{M-1}|f(x,y) - f'(x,y)|^2 \tag{2}$$

$$PSNR = 10 * log_{10}\frac{255^2}{MSE} \tag{3}$$

$$SSIM = \frac{(2\mu_x\mu_y + C_1)+(2\sigma_{xy}+C_2)}{(\mu_x^2+\mu_y^2+C_1)+(\sigma_x^2+\sigma_y^2+C_2)} \tag{4}$$

Where f(x,y) is the original image, $f^1(x,y)$ is the reconstructed image, and μ_x, μ_y, σ_x, σ_y, and σ_{xy} are the local means, standard deviations, and cross-covariance of the original and reconstructed images.

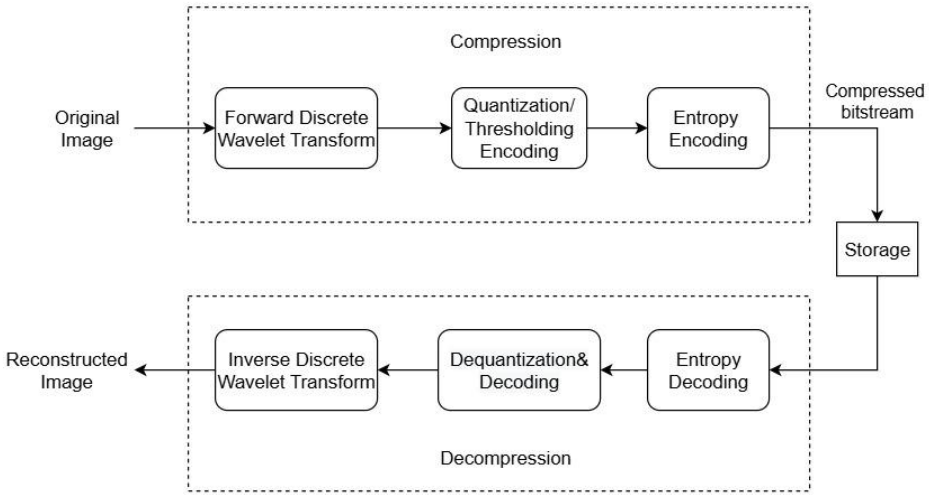


Fig. 1. Image compression flow diagram

2.1 Discrete Wavelet Transform (DWT)

The Discrete Wavelet Transform (DWT) is an image transform that converts the image from the spatial domain to the frequency domain using the basis function called wavelets. The basis function for DWT is not uniform, there exists a variety of wavelets in the literature. The output representation of DWT is a set of frequency components termed as coefficients, which are grouped into 4 different subbands. The NxN image is transformed using DWT to get approximate (LL), vertical (LH), horizontal (HL), and diagonal (HH) subbands[22]. The approximation subband contains detail coefficients (low frequency) and the remaining subbands contain finer coefficients (high frequency). Each subband is of size (N/4) x (N/4). Further, these subbands can be transformed for multiple levels and each subband is represented along with the level number. A two-level wavelet subbands are shown in Fig. 2.

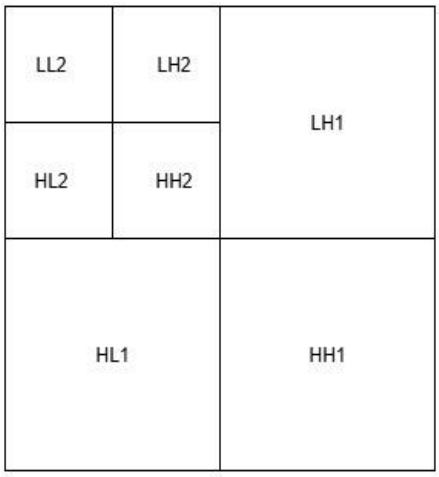


Fig. 2. Wavelet transform with two levels

2.2 Quantization/ thresholding, encoding, and entropy encoding

The process of quantization/thresholding reduces the quantization levels required to represent the coefficients and the quantized coefficients are encoded to get a large amount of compression. The entropy encoding is a lossless technique used to enhance the compression [23]. Most of the lossy compression algorithms use Huffman or Arithmetic coding for entropy coding [24, 25, 26].

EZW, SPIHT, and EBCOT are the 3 major progressive algorithms used in wavelet-based image compression. The summary of these three wavelet-based compression algorithms is shown in Table 1.

Table 1. Summary of wavelet-based compression algorithms.

Algorithm	Quantization/ thresholding	Coding	Entropy Encoding	Merits	Demerits	Ref.
EZW	The threshold is fixed in the beginning and the next iteration threshold is reduced by a factor of two. The process continued till the target bitrate is achieved or the threshold becomes 1.	The coefficients are encoded as positive symbol, negative symbol, zerotree, and isolated zero with dominant pass and subordinate pass.	Huffman coding is used to generate the bitstream from the coded symbols	Progressive image compression	Since the coding involves symbol generation and coding, the process is slower.	[6, 10]
SPIHT	Works with threshold same as EZW above.	The coefficient positions are arranged in various lists and bitstream is generated using sorting and refinement pass.	Huffman/ Arithmetic coding is performed to provide additional compression.	Progressive, Fast, and efficient bitstream generation. Entropy coding is optional since the bitstream is directly available after coding.	The coding depends on other subbands that make it a complex structure. The memory requirement is high.	[11, 13, 14, 27]
EBCOT	Quantization values are defined and the coefficients are quantized	Tier 1 followed by tier 2 coding is performed. In tier 1 bit plane coding (BPC) is performed.	Binary arithmetic coding (BAC) is performed as an intermediate step between BPC in tier 1 and tier 2 coding.	High compression ratio. Wavelet subbands are encoded independently.	Complex structure to implement on hardware.	[12, 13]

The advantage of the SPIHT algorithm for hardware implementation is its fast and efficient coding as compared with EZW and EBCOT. The implementation of the SPIHT algorithm on hardware is a complex and high memory requiring process. Another limitation of the SPIHT algorithm is the requirement for the presence of all the subbands of transformed coefficients throughout the encoding process. To overcome this, we have developed the FOGSPIHT algorithm, which generates the compressed bitstream using bitplanes of the transformed coefficients with faster encoding and decoding.

3. Overview of the SPIHT Algorithm

The arrangement of coefficients for SPIHT coding is shown in Fig. 3 [28]. The coefficients in the higher level subband are considered as the parent for the coefficients in the lower level subband. Each parent coefficient in the higher level subband will have 4 children in the lower level subband as shown in Fig. 3. This relation exists among the immediate subbands until there is no offspring left.

The coefficients are listed in four sets of coordinates [11]:

- O(i, j)-offspring node coordinate
- D(i, j)-descendant node coordinate
- H-all spatial orientation tree roots
- L(i, i) = D(i, j)- O(i, j).

O(i, j) except the lowest and highest subbands is given in the expression (5).

$$O(i,j) = \{(2i,2j),(2i,2j+1),(2i+1,2j),(2i+1,2j+1)\} \quad (5)$$

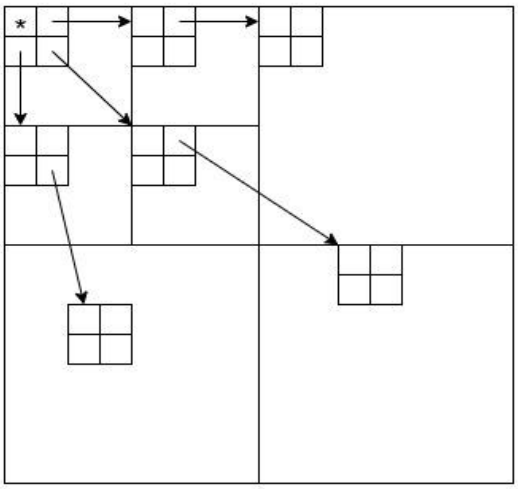


Fig. 3. The parent-offspring relationship between coefficients.

Initially, target bitrate and threshold values are fixed. The coding process is done by considering the coefficients in three lists: List of Significant Pixels (LSP), List of Insignificant Pixels (LIP), and List of Insignificant Sets (LIS). The coding process is done through sorting and refinement passes iteratively by halving the threshold value.

A coefficient greater than the threshold value is termed significant in the sorting pass; the significance and sign are encoded immediately as binary values. Each LIP indicates individual insignificant coefficients. Each LIS points to the 4 children coefficients to be checked for significance. As the coefficients are termed significant, they are moved to LSP and the insignificant ones are moved to LIP. The coefficients that are already significant in previous iterations are refined using the refinement pass.

The process of sorting and refinement requires the entire coefficient for the process of comparison with the threshold.

4. FOGSPIHT Algorithm

The image is transformed using DWT with the biorthogonal 9/7 wavelet basis. The biorthogonal 9/7 wavelet is better suited for image compression applications. The compression process using the FOGSPIHT algorithm converts the coefficients into a continuous string of 0s and 1s called the bitstream.

4.1 Coding and decoding principles of the FOGSPIHT algorithm

In the FOGSPIHT algorithm, the transformed coefficients are split into sign plane, and magnitude bit planes from MSB to LSB (n, n-1, …, 1) as shown in Fig. 4. Positive and negative signs of the coefficients are represented as 0s and 1s, respectively in the sign plane.

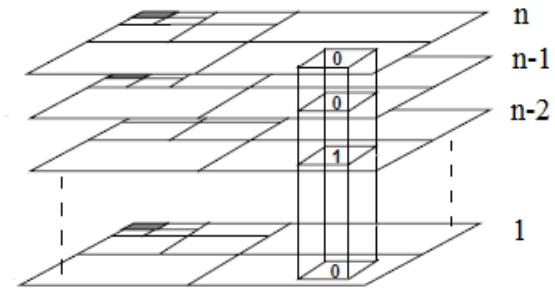


Fig. 4. Bit plane for wavelet of coefficients.

When a coefficient is represented in bitplanes, the coefficient becomes significant when the bit from the most significant bit (MSB) to the least significant bit (LSB) has the value of 1 for the first time. For example, the 10-bit binary representation of a coefficient having the value of 120 is 0001111000 (b9-b0). These 10 bits are the bitplane representation of the coefficient with value 120. The coefficient becomes significant at the 6^{th} bit, starting from the MSB, denoted as bitplane 6. A similar arrangement is done for all the NxN coefficients, which arranges them in n bitplanes. When a coefficient becomes significant, its sign is immediately added to the bitstream. In the coding process, bitplanes are considered from the most significant bitplane to the least significant bitplane. The consideration of the bitplane is based on its plane number. Here, it is just required to pass the significant coefficient's bitplane value to the bitstream, and immediately sign representation (positive or negative sign can be represented using 0&1 conventionally) is stored as the next bit in the bitstream. Therefore, the output generation in FOGSPIHT reduces to writing the bitplane value directly to the bitstream without requiring the comparison of the entire coefficient in comparison with the SPIHT algorithm. Checking the descendant validity is performed by the OR operation of all corresponding descendants. If any one of the descendants is 1, OR operation results in high output means descendants are valid. Whereas in the case of SPIHT algorithm, the largest coefficient among the descendants must be identified by the comparison operation.

The target bitrate is fixed in the beginning and the encoding process continues till the target bitrate is achieved. The sorting and refinement passes are performed iteratively to generate the compressed bitstream representing the image. The sorting pass identifies the 1st significant bit plane; both the significant bit and sign are encoded. Whereas the refinement pass encodes error information that appears in the succeeding bit planes.

The algorithm encodes the bitplanes by considering the coordinates under 3 lists namely List of Significant Pixels (LSP), List of Insignificant Pixels (LIP), and List of Insignificant Sets (LIS). The functioning of these lists are same as in the case of the SPIHT algorithm. The LSP holds the coordinate values of the bits from the bitplanes corresponding to the coefficients that have already become significant. LIP contains the coordinates corresponding to the insignificant coefficients. The LIS elements are termed as type A or type B, pointing to the bits from the next set of descendants of the same bitplane. Each LIS type A element corresponds to the descendants in the next generations till the end, whereas the type B elements hold similar information, except immediate next generation. If any of the four elements in the next generation descendants of LIS type A is/are significant, then they are moved to LSP. The insignificant elements are moved to LIP and the corresponding LIS element is marked as type B. If LIS type B descendants are found significant, then that element is deleted from the LIS list. This also adds four of its next-generation elements to LIS as type A.

At the beginning of the iteration, the bits corresponding to the position indicated by LIP are evaluated in order, and the values are written to the bitstream. The coordinate values of the bits found significant are moved to LSP. Similarly, the offspring bits indicated by LIS are evaluated in order, and coordinate values are moved to LSP or LIP based on the significance. The LIS is updated and new coordinates are appended at the end of the LIS.

Initially, all the bit planes representing the coefficients are considered insignificant. The bit planes from MSB to LSB are checked for significance in each iteration, called passes. In the sorting pass, the corresponding bit(i, j) will be written to the bitstream according to LIP or LIS evaluation as shown in Table 2. If the bit(i,j) =1, then its sign information is written to the bitstream, and the corresponding coordinate values are moved to LSP. LSP is used for the refinement pass of lower bitplanes of the coefficients that are significant in the previous sorting passes. The bits in the current bit plane are written to the bitstream corresponding to the previous LSP.

During the decoding process, bitstream bits are read one by one and placed in the appropriate bitplanes using LIP, LIS, and LSP lists. While decoding a particular bitplane, the next lower bitplane is also predicted at the same time, and later it is refined during its decoding. Once all the bits in the bitstream are completely inspected, the bitplanes and sign are packed together to get the coefficients, which are inverse DWT transformed to get the reconstructed image.

4.2 FOGSPIHT coding algorithm

The FOGSPIHT algorithm is given in the steps below:

Step 1) Initialization: Find the number of bitplanes n= ⌊log'2 (|Largest coefficient|)⌋. LSP with all bits as 0s, and bit(i, j) of bit plane n∈ H to the LIP. Coordinates of bit(i, j) with descendants including the first generation appended to the LIS, as type A entries.

Step 2) Sorting Pass: Read the nth bitplane and run the sorting pass as shown in Table 2. [Type A and type B indicate the descendants. Type A indicates all the descendants whereas type B indicates descendants except first generation offspring.]

Step 3) Refinement Pass:

For the previous LSP, output bit(i, j) to the bitstream in the current pass.

Step 4) Update n=n-1 and go to Step 2, which continues the coding process with the next bitplane.

5. Results and Discussion

5.1 Evaluation of FOGSPIHT algorithm in terms of MSE, PSNR, SSIM vs CR

The FOGSPIHT algorithm is tested using MRI images and the results are presented in this section. The images are transformed using DWT and compressed using the FOGSPIHT algorithm followed by Huffman encoding. The same experiment is performed with the SPIHT algorithm. Table 3, Table 4 and Table 5 present the values of MSE, PSNR and SSIM of 4 different MRI images respectively with both SPIHT and FOGSPIHT algorithms. Fig. 5, shows the graphical representation of MSE vs CR for different images, Fig. 6 shows the graphical representation of PSNR vs CR and Fig. 7 shows the graphical representation of SSIM vs CR for those images. The measured values of MSE, PSNR and SSIM from the above figures and tables show that the results obtained by the FOGSPIHT algorithm are equal to or marginally better than the SPIHT algorithm.

FOGSPIHT consists of direct writing of bit(i, j) to the bitstream, whereas SPIHT contains the comparison of the coefficient(i, j) with the threshold value. The FOGSPIHT algorithm works only with the plane number, corresponding bitplane values and the sign plane. It does not require the threshold value to be set at the beginning of each pass. In the case of finding LIS descendant validity check, the process is reduced to bitwise OR operation in FOGSPIHT instead of comparison and finding the largest number.

Table 2. Sorting Pass for FOGSPIHT.

- For bitplane n, bit (i,j) corresponding to LIP
- Output bit (i,j) to bitstream
- If bit (i,j)=1
  - ❖ Output sign (i,j) to bitstream
  - ❖ Append coordinate (i,j) to LSP
- For each entry (i,j) in the LIS
- If LIS entry type A
  - ❖ Perform OR operation of bits corresponding to D(i,j)
  - ❖ If OR (bits(D(i,j))) =1
    - ➢ For each (k,l) in O(i,j)
    - ➢ Output bit(k,l)
    - ➢ If bit(k,l)=1
      - ✓ Append (k,l) to LSP
      - ✓ Output sign(k,l)
    - ➢ Else
      - ✓ Append (k,l) to LIP
  - ❖ If L(i,j) = Φ
    - ➢ Remove (i,j) from LIS
  - ❖ Else append (i,j) to the end of LIS as type B
- Else if LIS entry type B
  - ❖ Perform OR operation of bits corresponding to L(i,j)
  - ❖ If OR (bits(L(I,j)))=1
    - ➢ Append (k,l) in O(i,j) to the end of LIS as type A
    - ➢ Remove (i,j) from LIS

In the decoding process, the bitstream is the input to the decoder and the bitplanes are modified according to the values of the bitstream with LSP matrix, LIP, and LIS as the same order in the encoder.

Table 3. MSE vs CR for image 1, image 2, image 3, and image 4.

	MSE (Image 1)		MSE (Image 2)		MSE (Image 3)		MSE (Image 4)	
CR	SPIHT	FOGSPIHT	SPIHT	FOGSPIHT	SPIHT	FOGSPIHT	SPIHT	FOGSPIHT
1.596	1.555	1.555	1.629	1.629	0.384	0.308	0.376	0.311
2.656	15.190	15.190	16.537	16.537	0.777	0.756	0.701	0.673
3.976	49.874	49.874	56.243	56.243	2.294	2.258	1.584	1.584
5.290	92.322	92.322	102.413	102.413	4.708	4.708	3.219	3.219
7.904	151.846	151.846	167.083	167.083	13.516	13.516	8.349	8.349
9.852	189.724	189.724	208.927	208.927	22.981	22.981	13.422	13.422
13.071	230.125	230.125	259.040	259.040	39.796	39.796	24.294	24.294

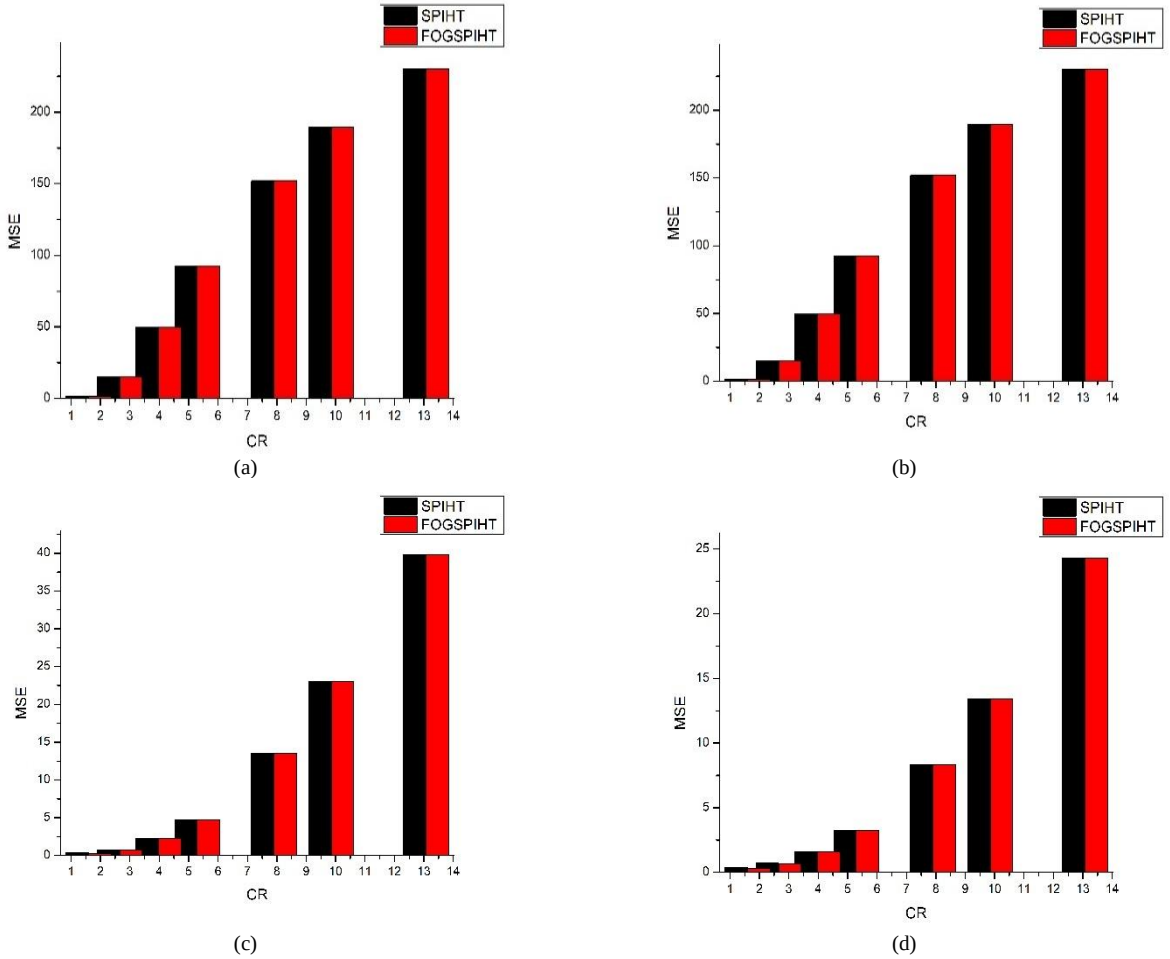


Fig. 5. MSE vs CR results: a) image 1; b) image 2; c) image 3; d) image 4.

Table 4. PSNR vs CR for image 1, image 2, image 3, and image 4.

	PSNR in dB (Image 1)		PSNR in dB (Image 2)		PSNR in dB (Image 3)		PSNR in dB (Image 4)	
CR	SPIHT	FOGSPIHT	SPIHT	FOGSPIHT	SPIHT	FOGSPIHT	SPIHT	FOGSPIHT
1.596	46.215	46.215	46.010	46.010	52.293	53.248	52.381	53.200
2.656	36.315	36.315	35.946	35.946	49.226	49.347	49.672	49.853
3.976	31.152	31.152	30.630	30.630	44.525	44.593	46.133	46.133
5.290	28.478	28.478	28.027	28.027	41.402	41.402	43.054	43.054
7.904	26.317	26.317	25.901	25.901	36.822	36.822	38.915	38.915
9.852	25.350	25.350	24.931	24.931	34.517	34.517	36.853	36.853
13.071	24.511	24.511	23.997	23.997	32.132	32.132	34.276	34.276

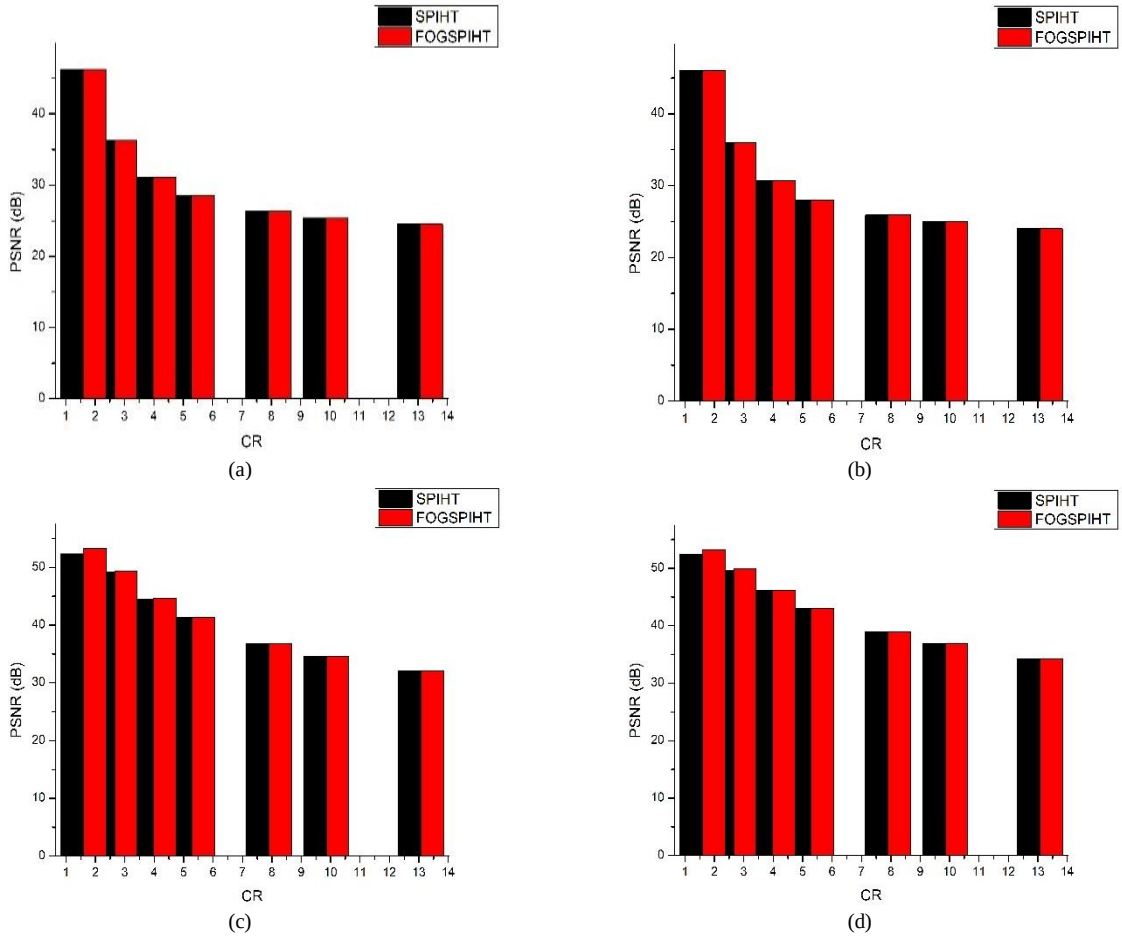
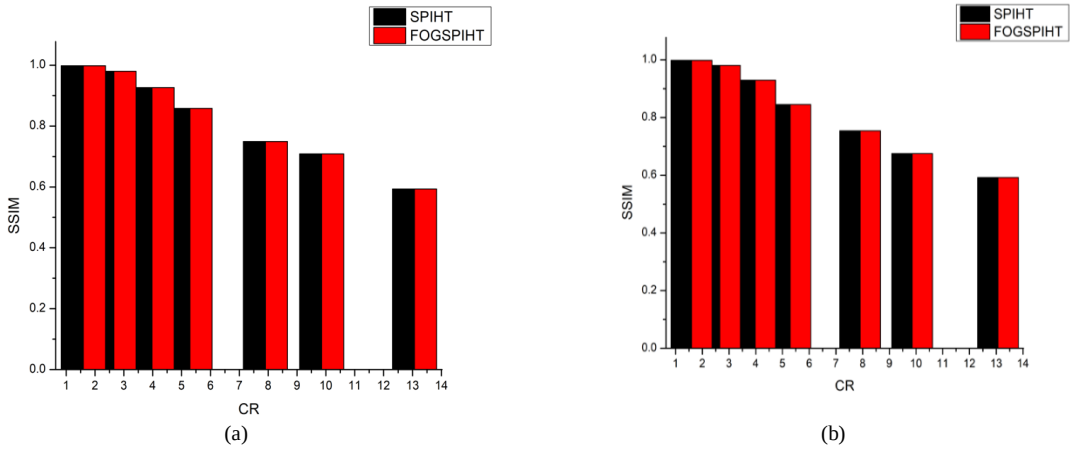


Fig. 6. PSNR vs CR results: a) image 1; b) image 2; c) image 3; d) image 4.

Table 5. SSIM vs CR for image 1, image 2, image 3, and image 4

	SSIM (Image 1)		SSIM (Image 2)		SSIM (Image 3)		SSIM (Image 4)	
CR	SPIHT	FOGSPIHT	SPIHT	FOGSPIHT	SPIHT	FOGSPIHT	SPIHT	FOGSPIHT
1.596	0.998	0.998	0.998	0.998	0.997	0.998	0.996	0.997
2.656	0.980	0.980	0.980	0.980	0.993	0.994	0.995	0.995
3.976	0.926	0.926	0.929	0.929	0.986	0.986	0.988	0.988
5.290	0.858	0.858	0.845	0.845	0.971	0.971	0.980	0.980
7.904	0.749	0.749	0.754	0.754	0.942	0.942	0.962	0.962
9.852	0.708	0.708	0.675	0.675	0.929	0.929	0.944	0.944
13.071	0.594	0.594	0.592	0.592	0.896	0.896	0.923	0.923



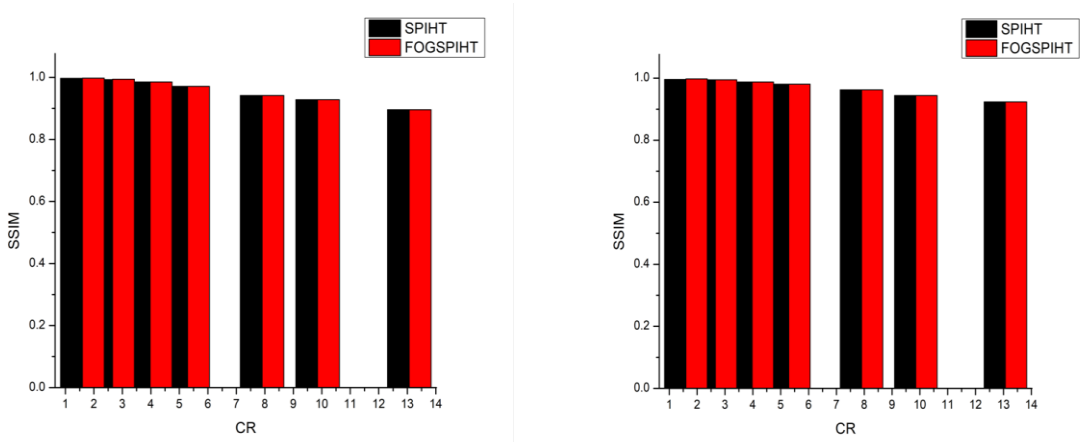


Fig. 7. SSIM vs CR results: a) image 1; b) image 2; c) image 3; d) image 4.

5.2 The encoding and decoding time of the FOGSPIHT algorithm

The experiments are performed on the system with an Intel i5 processor with a clock frequency of 2.40GHZ with 8GB RAM. The encoding and decoding times are presented in Table 6. Fig. 8 shows the respective plots of encoding time and decoding time of the SPIHT and FOGSPIHT algorithms. The measured encoding and decoding time values show that FOGSPIHT is faster than SPIHT. The SPIHT coding requires all the coefficients throughout the process, whereas the FOGSPIHT coding requires only a single bitplane thereby reducing the memory requirement.

Table 6. Encoding and decoding time vs CR for image 1.

CR	Encoding Time (in seconds)		Decoding Time (in seconds)	
	SPIHT	FOGSPIHT	SPIHT	FOGSPIHT
1.596	3.276	1.600	2.863	1.590
2.656	1.855	1.156	1.649	0.974
3.976	1.113	0.803	0.866	0.631
5.290	0.766	0.610	0.643	0.462
7.904	0.517	0.422	0.340	0.289
9.852	0.417	0.347	0.270	0.227
13.071	0.400	0.332	0.238	0.207

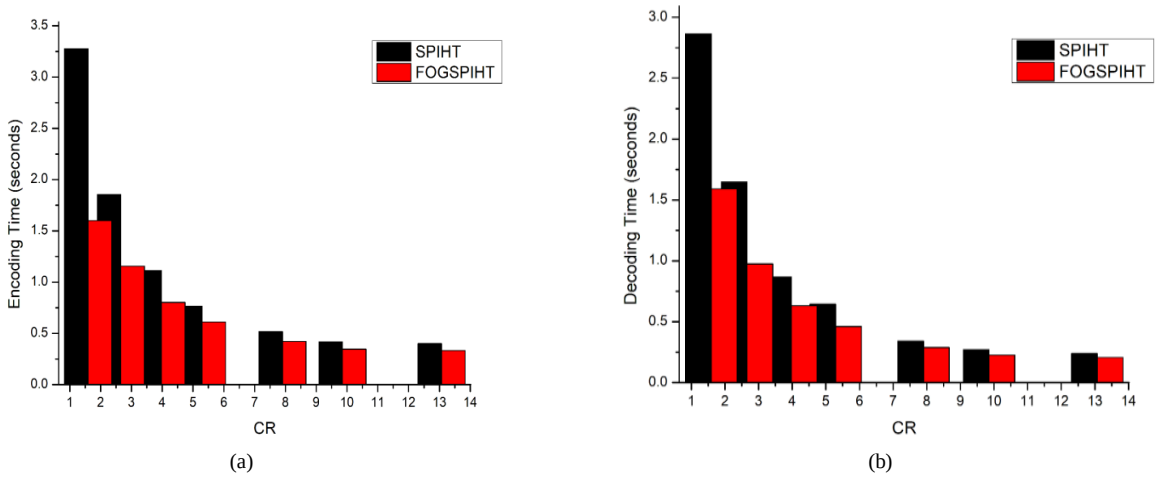


Fig. 8. Encoding and decoding time vs CR for image: 1 a) encoding time; b) decoding time.

The images used in the test, the compressed version using SPIHT, FOGSPIHT are shown in Fig. 9 for the mentioned compression ratio. The images reconstructed with SPIHT and FOGSPIHT have no visual differences.

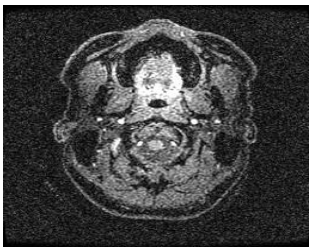
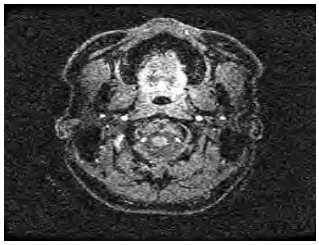
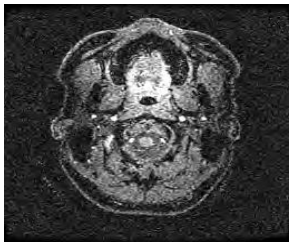
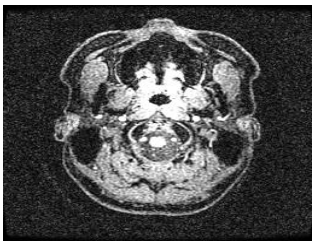
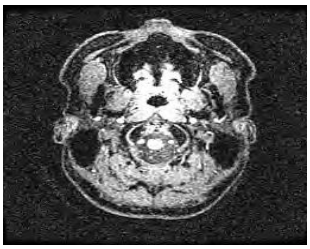
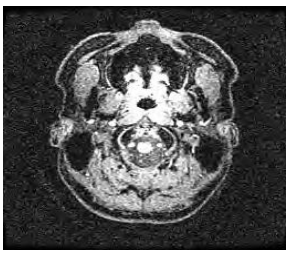
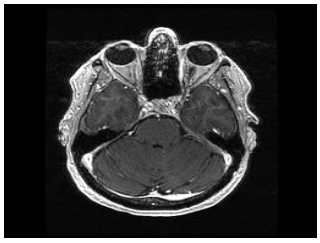
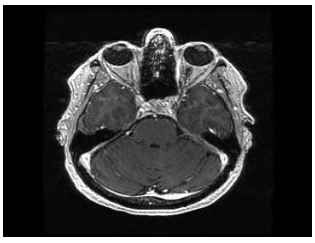
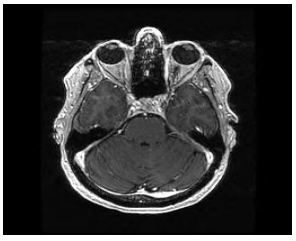
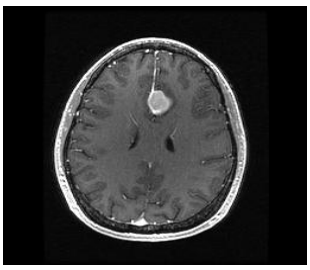
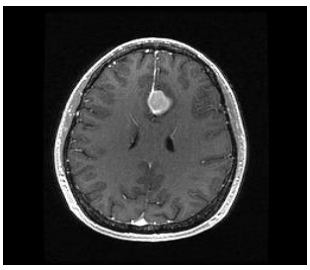
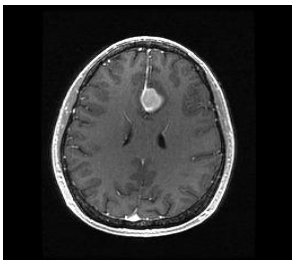
Image and CR	Original image	SPIHT compression	FOGSPIHT compression
Image1, CR=7.90			
Image 2, CR = 5.29			
Image 3, CR=3.97			
Image 4, CR=2.65			

Fig. 9. Images and their compressed versions.

5.3 Time and space complexity comparison for FOGSPIHT and SPIHT algorithms

The time and space complexities of both the SPIHT and FOGSPIHT algorithms are shown in Table 7. In time complexity, logN and the number of bitplanes n result in the same values. Therefore, both algorithms have the same values of time and space complexities. But FOGSPIHT is faster, which requires lower encoding and decoding time.

Table 7. Time and space complexities of SPIHT and FOGSPIHT.

Algorithm	Time complexity	Space complexity	Example execution time at CR=3.976	
			Encoding Time (in seconds)	Decoding Time (in seconds)
SPIHT	O(NlogN)	O(N)	1.113	0.866
FOGSPIHT	O(Nxn)	O(N)	0.803	0.631

5.4 Implementation of FOGSPIHT algorithm with 8x8 image on FPGA

The FOGSPIHT algorithm is implemented on Xilinx Artix 7 FPGA, considering an image of size 8x8 pixels. The 8x8 pixels are transformed using DWT and the coefficients are used as the input for both SPIHT and FOGSPIHT algorithms. The implementation details, consisting of lookup table (LUT), flipflop (FF), and power requirements, are shown in Table 8.

As the number of passes increased, the compression ratio goes on decreasing. This is because we are coding more and more bitplanes to the compressed bitstream. In this example, we used 4 passes which means we encoded 4 bitplanes starting from the MSB. The compressed bitstream generated using FOGSPIHT with an FPGA is the same as that obtained with the software method.

The FOGSPIHT algorithm requires a lower number of LUTs and lower power requirement to perform the same operation in comparison with the SPIHT algorithm. The FOGSPIHT algorithm eliminates the requirement of comparison operations by OR operation/direct bitstream writing. For example, if there are 100 coefficients, to conclude that any one of them is greater than the threshold, it is required to identify the largest number among them and compare it with the threshold. This is a mathematical operation that requires more resources. This method is used in SPIHT algorithm. The FOGSPIHT algorithm works with the bitplanes and a particular bitplane is used for generating the compressed bitstream. To perform the same for 100 coefficients with OR operation, it is just required to perform the OR operation of those coefficients' particular bitplane bits that we are encoding. This reduces the resource and power requirement required by the FOGSPIHT algorithm and this makes the FOGSPIHT algorithm a better choice for hardware implementation over the SPIHT algorithm.

Table 8. FPGA implementation details for 8x8 image.

Pass number	Synthesis/ Implementation	LUT		FF		Power (in milliwatts)			
		SPIHT	FOGSPIHT	SPIHT	FOGSPIHT	SPIHT	FOGSPIHT	% of reduction	
								LUT	Power
Pass1	Synthesis	42	52	18	18			-	-
	Implementation	42	52	18	18	19.31	17.758	-	-
Pass 2	Synthesis	41	45	18	18			-	-
	Implementation	41	45	18	18	8.734	10.650	-	-
Pass 3	Synthesis	157	122	59	59			22.30	
	Implementation	156	122	59	59	30.874	14.090	21.80	50
Pass 4	Synthesis	8018	4649	149	149			42.128	
	Implementation	7931	4634	149	149	353.538	233.277	41.572	34.127

6. Conclusion

In this work, a scalable, faster, and robust algorithm FOGSPIHT is proposed for MRI image compression based on Discrete Wavelet Transform (DWT). The values of Compression Ratio (CR), Mean Square Error (MSE), Peak Signal to Noise Ratio (PSNR), Structural Similarity Index Measure (SSIM) and encoding and decoding time are measured. The FOGSPIHT algorithm is faster compression and decompression algorithm than SPIHT. The measured values of MSE, PSNR, and SSIM with FOGSPIHT in comparison with those obtained with the SPIHT algorithm are equal to or better. Also, the visual appearance in both cases does not have differences. The implementation of FOGSPIHT on FPGA requires lower resources and power requirement. The comparison operations in SPIHT are reduced to writing the bit directly to the compressed bitstream or bitwise OR operation of descendants.

The FOGSPIHT can be implemented on FPGA for full image with inclusion of error handling capabilities with real world imaging scenarios.

References

[1] S. Dhawan, "A Review of Image Compression and Comparison of its Algorithms," Int. J. Electron. Commun. Technol., vol. 2, no. 1, pp. 22–26, 2011.

[2] V. P. B. Grover, J. M. Tognarelli, M. M. E. Crossey, I. J. Cox, S. D. Taylor-robinson, and M. J. W. Mcphail, "Magnetic Resonance Imaging: Principles and Techniques: Lessons for Clinicians," J. Clin. Exp. Hepatol., vol. 5, no. 3, pp. 246–255, 2015, doi: 10.1016/j.jceh.2015.08.001.

[3] P. Sreenivasulu and S. Varadarajan, "An Efficient Lossless ROI Image Compression Using Wavelet-Based Modified Region Growing Algorithm," J. Intell. Syst., vol. 29, no. 1, pp. 1063–1078, 2020, doi: 10.1515/jisys-2018-0180.

[4] D. A. Koff and H. Shulman, "An overview of digital compression of medical images: can we use lossy image compression in radiology?," Can. Assoc. Radiol. J. - J. l'Association Can. des Radiol., vol. 57, no. 4, pp. 211–217, 2006, [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/17128888

[5] A. Kaur and M. Goyal, "A Review: ROI based image compression of medical images," Int. J. Adv. Res. in …, vol. 4, no. 6, pp.9–12, 2014, [Online]. Available: http://www.ijarcsms.com/docs/paper/volume2/issue11/V2I11-0035.pdf

[6] E. Kofidis, N. Kolokotronis, A. Vassilarakou, S. Theodoridis, and D. Cavouras, "Wavelet-based medical image compression," in Future Generation Computer Systems, 1999, pp. 223–243. doi: 10.1016/S0167-739X(98)00066-1.

[7] S. G. Mallat, "A theory for multiresolution signal decomposition: The wavelet representation," Fundam. Pap. Wavelet Theory, vol. 2, no. 7, pp. 494–513, 2009, doi: 10.1109/34.192463.

[8] K. Bhatt, T. Kumar, S. Hassan, and S. Kimothi, "2D Discrete Wavelet Transformation (2D-DWT) for Nanoscale Morphological Analysis," Prabha Mater. Sci. Lett., vol. 2, no. 2, pp. 140–153, 2023, doi: 10.33889/pmsl.2023.2.2.010.
[9] M. C. H. Zerva, V. Christou, N. Giannakeas, A. T. Tzallas, and L. P. Kondi, "An Improved Medical Image Compression Method Based on Wavelet Difference Reduction," IEEE Access, vol. 11, pp. 18026–18037, 2023, doi: 10.1109/ACCESS.2023.3246948.
[10] J. M. Shapiro, "Embedded image coding using zerotrees of wavelet coefficients," IEEE Trans. Signal Process., vol. 41, no. 12, pp. 3445–3462, 1993, doi: 10.1109/78.258085.
[11] A. Said and W. A. Pearlman, "a New , Fast , and Efficient Image Codec Based on Set Partitioning in Introduction - Image," IEEE Trans. CIRCUITS Syst. VIDEO Technol., vol. 6, no. 3, pp. 243–250, 1996.
[12] D. Taubman and M. Marcellin, "High performance scalable image compression with EBCOT," Int. Conf. Image Process., vol. 3, pp. 344–348, 2002.
[13] M. R. Lone and N. U. D. Hakim, "A novel hardware-efficient spatial orientation tree-based image compression algorithm and its field programmable gate array implementation," Turkish J. Electr. Eng. Comput. Sci., vol. 27, no. 5, pp. 3823–3836, 2019, doi: 10.3906/elk-1903-14.
[14] C. K. S. Budhiraja, "Improvements of SPIHT in Image Compression- Survey," Int. J. Emerg. Technol. Adv. Eng., vol. 3, no. 1, pp. 652–656, 2013.
[15] F. W. Wheeler and W. A. Pearlman, "SPIHT image compression without lists," ICASSP, IEEE Int. Conf. Acoust. Speech Signal Process. - Proc., vol. 4, pp. 2047–2050, 2000, doi: 10.1109/ICASSP.2000.859236.
[16] A. Abu-Hajar and R. Sankar, "Wavelet based lossless image compression using partial SPIHT and bit plane based arithmetic coding," ICASSP, IEEE Int. Conf. Acoust. Speech Signal Process. - Proc., vol. 4, pp. 3497–3500, 2002, doi: 10.1109/ICASSP.2002.5745408.
[17] N. Sprljan, S. Grgic, and M. Grgic, "Modified SPIHT algorithm for wavelet packet image coding," Real-Time Imaging, vol. 11, no. 5-6 SPEC. ISS., pp. 378–388, 2005, doi: 10.1016/j.rti.2005.06.009.
[18] Y. Jin and H. Lee, "A Block-Based Pass-Parallel SPIHT Algorithm," vol. 22, no. 7, pp. 1064–1075, 2012, doi: 10.1109/TCSVT.2012.2189793.
[19] H. de J. Ochoa Domínguez, O. O. Vergara Villegas, and V. G. Cruz Sanchez, "Modified set partitioning in hierarchical trees algorithm based on hierarchical subbands," J. Electron. Imaging, vol. 24, no. 3, pp. 033004-(1-12), 2015, doi: 10.1117/1.jei.24.3.033004.
[20] O. Khalifa, "Wavelet Coding Design for Image Data Compression," Int. Arab J. Inf. Technol., vol. 2, no. 2, pp. 118–127, 2005.
[21] M. K. SD. Kasute, "ROI Based Medical Image Compression," Int. J. Sc. Res. Netw. Secur. Commun., vol. 5, no. 1, pp. 6–11, 2017.
[22] R. C. Gonzalez and R. E. Woods, "Wavelets and Multiresolution Processing," in Digital Image Processing, 2008, pp. 466–477.
[23] A. M. Raid, W. M. Khedr, M. A. El-dosuky, and A. Wesam, "Jpeg Image Compression Using Discrete Cosine Transform - A Survey," Int. J. Comput. Sci. Eng. Surv., vol. 5, no. 2, pp. 39–47, 2014, doi: 10.5121/ijcses.2014.5204.
[24] D. A. Huffman, "A Method for the Construction of Minimum-Redundancy Codes," Proc. I.R.E., vol. 40, no. 9, pp. 1098–1101, 1952, doi: 10.1109/JRPROC.1952.273898.
[25] P. G. Howard and J. S. Vitter, "Analysis of arithmetic coding for data compression," Inf. Process. Manag., vol. 28, no. 6, pp. 749–763, 1992, doi: 10.1016/0306-4573(92)90066-9.
[26] Y. T. Chen and D. C. Tseng, "Wavelet-based medical image compression with adaptive prediction," Comput. Med. Imaging . Graph., vol. 31, no. 1, pp. 1–8, 2007, doi: 10.1016/j.compmedimag.2006.08.003
[27] A. A. Noor Huda Ja'afar and S. I. Safie, "A State Table SPHIT Approach for Modified Curvelet-based Medical Image Compression," Int. J. Online Biomed. Eng., vol. 20, no. 1, pp. 89–103, 2024, doi: https://doi.org/10.3991/ijoe.v20i01.41363.
[28] M. Beladgham, A. Bessaid, M. Abdelmounaim, and T. Abdelmalik, "Improving quality of medical image compression using biorthogonal CDF wavelet based on lifting scheme and SPIHT coding," Serbian J. Electr. Eng., vol. 8, no. 2, pp. 163–179, 2011, doi: 10.2298/sjee1102163b.

Authors' Profiles

Narayana Prakash S. is a research scholar in the Department of Studies and Research in Electronics at Mangalore University, Karnataka, India -574199. He has graduated M.Sc. (Electronics) and B.Sc. (Physics, Mathematics, and Electronics) from the same university. His area of research includes image processing, embedded systems.

Dr. Airani Mohammad Khan is a senior professor in the Department of Studies and Research in Electronics at Mangalore University, Karnataka, India -574199. He was awarded Ph.D. by Mangalore University, Karnataka, India, 574199 in 2006. He is an excellent researcher, academician, and able administrator. He has teaching experience of over 30 years. He guided students in various capacities. His research areas include artificial intelligence, biomedical image processing, embedded system design, image / data processing, algorithm development, IC fabrication technology, and computer networks. He published over 150 articles in reputed journals.

How to cite this paper: Narayana Prakash S., Airani Mohammad Khan, "A Fast Output Generating Set Partitioning in Hierarchical Trees Coding for Medical Image Compression", International Journal of Image, Graphics and Signal Processing(IJIGSP), Vol.17, No.6, pp. 169-181, 2025. DOI:10.5815/ijigsp.2025.06.10