SSRG International Journal of Electronics and Communication Engineering
ISSN: 2348-8549/ https://doi.org/10.14445/23488549/IJECE-V11I12P107

Volume 11 Issue 12, 71-82, December 2024

Original Article

Curvelet Transform Based Hyperspectral Image Compression with Listless Set Partitioned Compression Algorithm for Unmanned Aerial Vehicle Image Sensor

Vinod Kumar Tripathi[1], Shrish Bajpai[2]

[1,2]*Electronics & Communication Engineering Department, Faculty of Engineering & Information Technology, Integral University, Lucknow, Uttar Pradesh, India.*

[2]*Corresponding Author : shrishbajpai@gmail.com*

Received: 12 October 2024 Revised: 16 November 2024 Accepted: 06 December 2024 Published: 30 December 2024

Abstract *- Over the past several years, we have witnessed remarkable progress in hyperspectral (HS) images taken by unmanned aerial vehicles. The HS image is very high in spectral resolution in many narrow contiguous bands. The HS image compression becomes necessary to effectively handle large amounts of remote sensing data for storage and communication purposes. In recent years, many compression algorithms have been proposed to achieve a high compression ratio, but they either suffer from coding efficiency or coding memory or coding complexity. Transform-based Hyperspectral Image Compression Algorithm (HSICA) exploits the correlation that exists between the pixels or frames of the HS image and works with both types of compression. The wavelet transform-based set partitioned HSICAs are a special type of transform based HSICAs that use data-dependent link lists or image size-dependent state tables to track the significance of sets or coefficients and have better compression performance than other HSICAs due to the exploitation of the HS image redundancies. The proposed compression algorithm 3D-Low Memory Zerotree Coding (3D-LMZC) uses the curvelet transform to improve directional elements and better the ability to represent edges and other singularities along curves. The objective of the proposed HSICA is to achieve a high compression ratio while simultaneously representing HS images at a variety of scales and directions. This will allow for the provision of compressed HS images of a high quality. The results of the experiments reveal that the suggested approach has a low coding memory demand, and compared to other state-of-the-art compression algorithms, it achieves an increase in coding gain of approximately 5%.*

Keywords *- Curvelet transform, Hyperspectral image compression, Multiresolution, Set partitioned compression algorithm, Zerotree coding.*

1. Introduction

Hyperspectral Imagery (HSI) collects electromagnetic spectrum information (spatial and spectral) in many continuous and narrow spectral bands from ultraviolet to infrared wavelength. The HS image is 3D data (spatio-spectral cube) with 2D spatial information and 1D spectral signature [1, 2]. Due to this strong resolving power for fine spectra, the HS images are utilized extensively in a variety of contexts such as archaeological analysis, anomaly detection, biomedicine, cultivation, climatology, change area detection, drug identification, meteorology, security, food safety inspection and control, medical, mining, pollution monitoring, remote sensing, oceanography etc. [3, 4]. The HS image sensors generate data that must be compressed before it communicates to the receiver station. Thus, the image compression process of HS image becomes necessary to save data transmission bandwidth, reduces demand of sensor memory, lower energy consumption, and minimise data processing time [5]. The HSICA aims to improve the sensor performance by reducing computational complexity and removing unwanted data redundancy, achieving high coding efficiency and diminishing coding memory requirement. The image Compression Ratio (CR) is a unit less parameter, a ratio of the memory required by the uncompressed HS image to the reconstructed HS image. High data redundancy gives high CR, while low data redundancy gives low CR [6].

The compression algorithms can be classified based on the coding process or HS image data loss. The HS image data loss approaches include lossy compression, lossless compression, and near-lossless compression, while based on the coding process, can be further categorized into predictive coding, vector quantization, compressive sensing, transform coding, unmixing based compression, neural network based, and machine learning [7]. There is no loss of data for the

lossless compression, while for near-lossless, there is very less data loss. Data loss happens for the lossy compression but depends upon the coding process and bit rate.

In view of the categorization of the HSICAs based on the coding complexity, the predictive coding-based HSICA has low complexity. In contrast, neural networks and machine learning-based HSICA have high coding gain with very high coding complexity.

In the predictive coding based HSICA, one or more predictors are used to calculate the residual errors in the spectral dimension of the HS image and the errors are coded by any entropy coding methods (arithmetic coding, Huffman coding, variable length coding, etc.) [8]. The vector quantization, known as the dictionary-based or code book method, has high coding complexity and gain. It has four steps (block coding, training set generation, codebook generation, and quantization). The codebook is different for each HS image sensor. This method is useful when a large amount of HS data must be transmitted, as the codebook generation is performed only once. Thus, it saves time with high coding gain for huge HS data [9, 10]. Compressive sensing-based HSICA shifts the computational complexity from the encoder to the decoder. It partitioned the HS image data into small blocks (pieces), compresses it and sends it to the decoder end. These compression algorithms have low coding memory requirements during the process and require less complex hardware than traditional software [11].

The Neural Network (NN) based HSICAs use multiple layer structures to compress HS images. It has a very high coding gain with a very complex neural network structure, increasing the compression algorithm's complexity. The neural network-based compression algorithms work with predictive coding or transform coding algorithms [17]. The machine learning-based HSICAs use the learning methods (deep learning, etc) to compress HS image data. In the same way, as NN based HSICA, learning based HSICA has very high complexity. It is useful for huge HS data compression because the learning process is only completed once. These algorithms change from sensor to sensor depending on the type of sensor [12].

In recent times, FPGA based HSICAs have been proposed. These compression algorithms use the spectral un-mixing of the pixels of HS images to achieve compression. These algorithms have two steps in which endmembers are extracted from the HS image followed by the abundance of each endmember is determined for each pixel vector of HS images [13].

Transform coding compression algorithms have low coding memory requirements have low complexity, and can work with lossy and lossless compression processes. Wavelet, Karhunen-Lobve, and cosine transform are the most used mathematical transform in transform-based HSICA [7]. The wavelet transform has many advantages, including simultaneous localization in the time and frequency domain [14]. The curvelet transform, on the other hand, contains singularities comparable to curves, and it provides information about the scale, location, and orientation of the image, which positions it as an excellent option for image compression. Apart from HS image compression, curvelet transform is also used for image fusion [15], feature classification [16] and image denoising [17].

The 3D set partitioned HSICAs are a special type of compression algorithm that uses the set structure of the transform HS image to track the significance of insignificance of the sets or coefficients. The 3D set partitioned HSICAs have lower coding complexity, high coding gain, embeddedness, and fewer coding memory requirements than other transform HS image compression algorithms. The 3D-Set Partitioning in Hierarchical Trees (3D-SPIHT), 3D-Set Partitioned Embedded bloCK (3D-SPECK), 3D-No List SPIHT (3D-NLS) are state of the art transform based HSICA. The complex compression algorithm requires a complex sensor architecture with much energy to execute the process [18, 19].

This manuscript identifies the coding memory, coding complexity, and coding efficiency issues that occur in the hyperspectral image compression process. The contributions of this study include the following.

- High coding memory poses a great threat to the performance of the HS image sensors. The proposed HSICA minimize the requirement for coding memory by the use of less number of markers.
- The coding complexity is at par with state of the art 3D-NLS but is less than multi-fold to the 3D-SPIHT.

The present compression algorithm follows the same partition rule and mathematical transform (curvelet transform) as 3D-Listless Embedded Zerotree Set Partitioning Coding (3D-LEZSPC) [23]. It uses a smaller number of state markers than 3D-LEZSPC. Through this, the demand for coding memory reduces by 20%. The coding efficiency of the proposed compression algorithm and 3D-LEZSPC is almost the same, but the coding efficiency is slightly higher.

The remaining sections are organized as follows. First, the works related to transform based HSICA are covered in Section 2, which includes the details about the curvelet transform and set partitioned-based transform HSICA. Section 3 proposes the details of our proposed 3D-Low Memory Zerotree Coding (3D-LMZC). The detailed experimental evaluations of the proposed 3D-LMZC with the other HSICA are reported in Section 4. Section 5 concludes the article with a discussion of future trends.

2. Related Work

Last couple of decades, many compression algorithms have been proposed for hyperspectral images. Initially, the compression algorithm considers HS images as 3D data arranged as a continuous stack of 2D data. The compression algorithm compressed the HS image frame by frame. Through this, the compression algorithm failed to eliminate the correlation (spectral) between the different HS image frames. The 3D mathematical transform removes the spectral and spectral correlation. The wavelet transform has many advantages to the Fourier transform, such as extracting local spectral and temporal information. But, the new family of mathematical transforms, such as the curvelet transform, showed better coding efficiency than the wavelet transform. Generating image data creates the issue with the wireless channel and affects the performance of the devices associated with it. The compression algorithms also address this issue.

2.1. Curvelet Transform

Curvelet Transform (CT) has multiple advantages over the wavelet transform, such as curve similar singularities, which provide adequate information about the scale, location, and orientation [20]. Candes and Donoho invented the first version of the curvelet transform in 1999. However, it suffered from excessive coding complexity and output image redundancy as it was the first generation. The second generation curvelet transform (fast discrete curvelet transform) is developed to solve the above mention issues. Two distinct types of curvelet transform families are utilized for the fast curvelet transform. They are the Unequispaced Fast Fourier Transform (USFFT) based curvelet transform and the frequency wrapping based curvelet transform. The section of the above methods depends on the application type [21, 22]. The curvelet transform-based HSICA 3D-LEZSPC shows a higher coding efficiency than wavelet transform based HSICA [23].

2.2. Set Partitioned Hyperspectral Image Compression Algorithms

The set partitioned HSICAs achieve compression by aggregating many unimportant coefficients in either spatial block cubes, spatial trees, or spatial block cube trees. These algorithms have low coding complexity and can work with lossy and lossless compression processes according to the availability of the remaining bits. The significance of the coefficients, block cubes, or block cube tree is either tracked by the linked lists or markers (state table). The set partitioned HSICAs future divided into two types named list-based set partitioned HSICA and listless set partitioned HSICA [24-33]. The list-based set partitioned HSICA uses data-dependent link lists to track the significance/insignificance of the coefficients/sets. The 3D-SPECK and 3D-SPIHT are the most famous HSICA as they have low computational complexity and high coding efficiency than other wavelet transform-based HSICAs [24, 25]. The listless set partitioned HSICA uses markers to track the significance/insignificance of the coefficients/sets. The 3D-LSK and 3D-NLS are state of art compression algorithms in this category [29, 30].

3. 3D- Low Memory Zerotree Coding (3D-LMZC)

The proposed algorithm uses the same set structure and partition rule and generates similar output embedded bit stream types as 3D-SPIHT [17] and 3D-NLS [12]. The functionalities of the lists LIP, LIS and LSP in the 3D-SPIHT are accomplished by the fixed size of the markers. The average coding memory required by the markers in 3D-NLS is 8 bits per coefficient, while the proposed algorithm needs an average of 1.25 bits per coefficient average coding memory. The demand for coding memory is minimized by using only two different markers: the State Marker for Coefficients (SMC) and the State Marker for Zerotrees (SMZ).

The SMC markers [δ] are used to define the state (significance) of the coefficients. It works like a scanner. The current coefficient is tested for significance, insignificance, or refinement against the current threshold. Thus, a coefficient must be assigned to any two values for the SMC marker depending on the situation against the current threshold. These two values of SMC for each coefficient need only one bit per coefficient. All coefficients in the LLL sub-band are initialized by the SMC = ‘0’ marker, while the rest are initialized by the SMC = ‘1’ marker. The already significant coefficients having the SMC value '1' will generate the numeric value outputting the nth significant bit of the corresponding coefficient. The SMC markers are defined in Table 1. The top bit plane 'n' is calculated in Equation 1 as

$$n = \lfloor \log_2 \max[\Gamma_i] \rfloor \quad (1)$$

Table 1. Explanation of SMC markers

Marker	Explanation
δ[0]	Coefficient is not under consideration for significant tests against ongoing bit plane.
δ[1]	The coefficient will be checked for significance/insignificance/refinement for significant tests against the ongoing bit plane.

In addition to the SMC markers, one more marker is used to define the state (significance or insignificance) of the zerotrees. These markers are known as State Markers for Zerotrees (SMZ). The SMZ markers have the same functionalities as LIS in 3D-SPIHT. From the property of wavelet transform, seven-eighth of total transform coefficients are present in the seven-eighth lowest sub-bands of the wavelet pyramid and have no descendants (end leaf nodes of zerotree). Hence, these coefficients need not be assigned any SMZ value. The detail descriptions of the SMZ markers have been covered in Table 2.

Table 2. Explanation of SMZ markers

Marker	Explanation
ω[0]	No consideration is given to the tree node in the significant test that is performed against the contemporaneous threshold.
ω[1]	To type 'A' zerotree node: A node whose offspring and descendants are examined to see whether they significantly contribute to the ongoing bit plane.
ω[2]	Kind of 'B' zerotree node: A node whose offspring and descendants are examined to see whether they significantly contribute to the ongoing bit plane.
ω[3]	The tree node will not tested for significance in the upcoming bit plane.

In the significance testing of the coefficients, the zerotree node has any one possible state mentioned in Table 2. The two-bit marker defines this state. Simple arithmetic suggests that, on average, for any coefficient, only 0.25 bit per zerotree node is required for the SMZ marker. So, the total memory requirement for the HS image is given as 1.25 bits per coefficient.

In the proposed algorithm, all zerotree nodes in the LLL sub-band are initialized with the SMZ = '1' while the remaining zerotree nodes are initialized with the SMZ = '0'. During the significance testing of the zerotree nodes, if Type A zerotree is found significant, the output is coded with the '1' and partitioned into eight children and associated grand descendent set. First, eight immediate descendants are tested for significance against the current bit plane, and the output is encoded by the magnitude bit and sign bit. The status of the SMC marker is updated accordingly. Afterwards, the grand descendants are tested for significance against the current threshold. If any one of them is found significant, then eight new Type, A zerotree with the offspring as zerotree node, will be created with SMZ = '1', and the original zerotree root node is assigned the new marker value as SMZ = '3'. If the Type A zerotree is found insignificant, the associated marker is assigned the new numeric value as SMZ = '2'. The rest of the zerotree nodes are tested sequentially for their significance.

The proposed algorithm has no skip markers as 3D-NLS is used for the first coefficient of each zerotree node. If Type B zerotree is found significant, then eight new type A created from its parents' node and grandparent node are assigned the new marker value SMZ = '3'. In contrast, if it is found insignificant, the encoder will move to the next node without changing any values of the markers.

Table 3. 3D-LMZC encoding process

3D-Low Memory Zerotree Coding Algorithm

Encoding Process

Input: Curvelet Transform (5 levels) is applied to the HS image with the dimension M x N x P. Using the linear indexing method, it is transformed into the one-dimensional array Γ.

Output: Generation of Embedded bitstream

Initialization Pass

Fix: All coefficients of the LLL sub-band are assigned SMC [δ] = '1' while others are set as SMC = '0'

Fix: Tree nodes of the LLL band with SMZ [ω]= '1' and the rest of the associated tree nodes are mentioned as SMZ = '0'

Bit plane (all) $n = \lfloor \log_2 \max[\Gamma_i] \rfloor$

Required bits $\eta = [Bit\ rate * M * N * P]$

Bit Plane Pass

while (n≥0)

Insignificant Coefficient Pass

for i = 1:η
  if {δ (i) = '1' && $2^n \le \Gamma(i) \le 2^{n+1}$ }
    Output = '1' and sign bit of Γ(i)
  end
end

Insignificant Set Pass

for j = 1:η/8
  if {ω(j) = '1'}
    if {Max_SoT_A ≥ 2^n}
      Output = '1'
      for μ = {8j-7 : 8j}
        if {$2^n \le \Gamma(\mu) \le 2^{n+1}$ }
          Output = '1' and sign bit of Γ(μ)
          Set δ (μ) = '1'
        end
      end
    end
    if {Max_SoT_B ≥ 2^n}
      Output = '1'
      Fix ω (8j-7) = '1' ; Fix ω (8j-6) = '1'
      Fix ω (8j-5) = '1' ; Fix ω (8j-4) = '1'
      Fix ω (8j-3) = '1' ; Fix ω (8j-2) = '1'
      Fix ω (8j-1) = '1' ; Fix ω (8j) = '1'
      Fix ω (j) = '3'
    else
      Fix ω (j) = '2'
      elseif { ω (j) = '3'}
        if {Max_SoT_B ≥ 2^n}
          Output = '1'
          Fix ω (8j-7) = '1' ; Fix ω (8j-6) = '1'
          Fix ω (8j-5) = '1' ; Fix ω (8j-4) = '1'
          Fix ω (8j-3) = '1' ; Fix ω (8j-2) = '1'
          Fix ω (8j-1) = '1' ; Fix ω (8j) = '1'
          Fix ω (j) = '3'

```
                end
            end
        end
    end
end
```

Refinement Pass

```
for γ = 1: η
    if {δ(γ) = '1' && Γ(γ) ≥ 2^(n+1) }
        Output = n^th bit of the Γ(γ)
    end
end
n = n-1
end
```

4. Results and Discussion

In order to show the effectiveness of the 3D-LMZC, its performance is compared with the other HSICA 3D-SPECK (Compression Algorithm 1) [25], 3D-SPIHT (Compression Algorithm 2) [25], 3D-SDB-SPIHT (Compression Algorithm 3) [26], 3D-FSPIHT (Compression Algorithm 4) [27], 3D-WBTC (Compression Algorithm 5) [28], 3D-LSK (Compression Algorithm 6) [29], 3D-NLS (Compression Algorithm 7) [30], 3D-LMBTC (Compression Algorithm 8) [31], 3D-LCBTC (Compression Algorithm 9) [32] and 3D-ZM-SPECK (Compression Algorithm 10) [33] on four publicly available HS images named as Washington DC Mall (HSI I), Yellowstone Scene 0 (HSI II), Yellowstone Scene 3 (HSI III), and Yellowstone Scene 18 (HSI IV). The Yellowstone dataset has natural HS images of trees, forests, water, etc., while the Washington DC Mall HS image has man-made structures. Washington DC Mall HS has a spatial dimension of 1280 and 307 with 191 frequency frames, while Yellowstone dataset images have a spatial dimension of 512 and 680 with 224 frequency frames. For the experiments, the HS images are cropped '256 x 256 x 256' from the topmost left corner of the HS image. The five-level curvelet transform (unequispaced fast fourier transform) converts the HS image to the frequency domain. After the transform, the HS image is quantized to the nearest integer. The linear indexing approach is utilized to transform from the 3D matrix to the 1D array. The coding efficiency, coding memory and coding complexity are the performance metrics used to evaluate the proposed HS image compression algorithm with other compression algorithms. The Peak Signal to Noise Ratio (PSNR) calculated in decibels (dB) as a unitless parameter is used to measure the coding efficiency of the compression algorithms. Kilobytes (KB) are used to calculate the amount of coding memory, while seconds are used to measure the complexity of the coding.

4.1. Coding Efficiency

The PSNR measures the coding efficiency. The PSNR is used to measure the difference between the pixel values of the HS images (original and reconstructed) [34]. The HS image before the compression process is defined as G (α,β,γ) and the HS image after the compression process is defined as H (α,β,γ). The dimension of the HS image cube is defined as 'λ'. The G_{max} is the maximum numeric value of the original HS image. Mathematically, it is defined in Equation 2 as

$$PSNR = 20 \log_{10}\left[\frac{G_{max}}{\frac{1}{(\lambda \times \lambda \times \lambda)} \sum_x \sum_y \sum_z [G(\alpha,\beta,\gamma) - H(\alpha,\beta,\gamma)]^2}\right] \quad (2)$$

The compression ratio (CR) is defined in Equation 3 as

$$Compression\ Ratio\ (CR) = \frac{\sum_x \sum_y \sum_z [G(\alpha,\beta,\gamma)]}{\sum_x \sum_y \sum_z [H(\alpha,\beta,\gamma)]} \quad (3)$$

PSNR (coding efficiency) versus bit rate for different hyperspectral compression algorithms with 3D-LMZC is tabulated in Table 4. From the table, it is clear that 3D-LMZC has a higher coding gain than other HSICAs. Due to the curvelet transform, the proposed compression algorithm has a higher coding gain than other compression algorithms. Simulation results show that 3D-LMZC surpasses the 3D-SPIHT and 3D-NLS at each bit rate. It is observed that the variation of PSNR between 3D-LMZC and 3D-SPIHT exists in the range of 0.19 dB to 0.38 dB for HSI I, 0.25 dB to 0.53 dB for HSI II and 0.14 dB to 0.35 dB for HSI III. The variation of PSNR between 3D-LMZC and 3D-SDB-SPIHT exists in the range of 0.31 dB to 0.56 dB for HSI I, 0.45 dB to 0.74 dB for HSI II and 0.29 dB to 0.63 dB for HSI III. The variation of PSNR between 3D-LMZC and 3D-FSPIHT exists in the range of 0.25 dB to 0.63 dB for HSI I, 0.45 dB to 0.68 dB for HSI II and 0.3 dB to 0.6 dB for HSI III. In the same way, variation of PSNR between 3D-LMZC & 3D-NLS exists in the range of 0.33 dB to 0.76 dB for HSI I, 0.34 dB to 0.74 dB for HSI II and - 0.4 dB to 0.51 dB for HSI III. The 3D-LMZC is outperformed in all bit rates except for the 3D-NLS, as it has more significant coefficients than the proposed HSICA 3D-LMZC.

Table 5 details the different compression algorithms based on significant bits (Newly Significant Coefficients as NSC) and refinement bits (Refinement Coefficients as RC) for the different bit rates. It has been clear that the proposed 3D-LMZC generates a more significant bit than other state of art HSICA for almost all bit rates with more refinement bits. The results presented in Table 5 make it abundantly evident that 3D-LMZC outperforms any other state-of-the-art HSICA. The Bjontegaard metric calculation calculated in Table 6 is a savings bit calculation using PSNR as the choice of objective quality metric across all bit rates under test [35].

4.2. Coding Memory

The listless HSICAs require fixed coding memory, which depends on the size of the HS image under test while list-based HSICAs require dynamic memory depending on

the bit rate (bpppb). For extremely low bit rates (0.001 to 0.1), 3D-SPIHT, 3D-SDB-SPIHT, and 3D-FSPIHT have low coding memory requirements because less coefficients are present for the processing. However, for the high bit rate, the demand for coding memory of these HSICAs increases rapidly, making them slow. Listless HSICAs perform well with constant coding memory requirements for low to high bit rates. The 3D-LMZC uses only two markers, requiring only 1.125 bits per coefficient marker, while state of the art 3D-NLS requires 8 bits per coefficient marker. The SMC marker requires a coding memory of one bit per coefficient, while the SMZ marker requires an average of 0.125 bits per coefficient coding memory. Less coding memory requirement reduces the multiple reading/writing operations, saving sensor power. Table 7 gives the cumulative coding memory requirement for the different HSICAs.

4.3. Coding Complexity

Due to the listless compression, it has a lower complexity than other list based compression algorithms but has a greater time requirement than 3D-NLS, as shown in Table 8. It is common knowledge that the time required for encoding will always be more than the time required for decoding. This is because comparison operations are skipped during the decoding process in favor of more efficient skipping of zerortrees and sub-bands [36, 37]. The proposed HSICA has the lowest decoding time in most of the bit rates, as shown in Table 9. The listless HSICAs have low coding complexity as they cannot perform multiple read or write operations (memory access). The proposed HSICA has higher coding complexity than 3D-NLS because it has more computations in searching for the coefficients and sets, which requires more time.

Figure 1 provides a visual representation of the HS image (original) as well as the reconstructed HS image that was produced following the compression procedure for HSI II (frames 50, 100, and 150).

5. Comparative Analysis

Table 10 presents a concise comparative analysis of the various state-of-the-art HSICAs compared to the suggested compression method known as 3D-LMZC. The findings make it abundantly evident that the 3D-LMZC method performs far better than the most advanced link list compression algorithm.

Table 4. Performance evaluation of different HSICAs with proposed 3D-LMZC on coding efficiency (PSNR)

Bit Rate	Compression Ratio	Compression Algorithm 1 [24]	Compression Algorithm 2 [25]	Compression Algorithm 3 [26]	Compression Algorithm 4 [27]	Compression Algorithm 5 [28]	Compression Algorithm 6 [29]	Compression Algorithm 7 [30]	Compression Algorithm 8 [31]	Compression Algorithm 9 [32]	Compression Algorithm 10 [33]	3D-LMZC [Proposed]
	HSI I											
0.001	14000	26.28	26.28	26.34	26.22	26.25	26.14	25.90	26.26	26.41	26.32	27.07
0.005	2800	28.95	28.95	29.01	28.84	28.93	28.71	28.71	28.70	28.66	28.73	29.84
0.01	1400	30.08	30.08	30.14	29.87	30.04	29.99	29.83	29.98	30.01	29.99	30.91
0.05	280	34.23	34.23	34.33	34.27	34.21	34.04	33.81	33.99	34.29	34.06	34.97
0.1	140	37.22	37.22	37.48	37.11	37.20	36.96	37.00	36.83	37.34	36.87	38.14
0.25	56	42.17	42.17	42.21	42.27	42.16	41.62	41.69	41.34	42.28	41.37	43.08
0.5	28	48.02	47.99	48.09	47.91	47.97	47.01	47.79	47.51	48.11	47.55	48.95
	HSI II											
0.001	16000	27.11	26.75	26.63	26.69	27.09	26.83	26.61	26.75	26.87	26.82	27.89
0.005	3200	29.45	29.31	29.11	29.17	29.43	29.27	29.25	29.24	29.41	29.25	30.18
0.01	1600	30.28	30.19	30.01	30	30.27	30.27	30.15	30.31	30.53	30.33	31.11
0.05	320	33.76	33.61	33.45	33.5	33.73	33.56	33.59	33.51	33.69	33.54	34.52
0.1	160	35.57	35.44	35.4	35.33	35.56	35.49	35.41	35.45	35.55	35.46	36.21
0.25	64	39.30	39.19	39.01	38.94	39.29	39.26	39.17	39.22	39.37	39.23	40.02
0.5	32	43.62	43.65	43.51	43.54	43.51	43.57	43.26	43.55	43.62	43.58	44.53
	HSI III											

0.001	16000	27.82	27.49	27.31	27.34	27.8	27.78	27.28	27.88	28.07	27.92	27.97
0.005	3200	30.24	30.09	29.84	29.91	30.22	30.03	30.03	30.01	30.44	30.02	31.17
0.01	1600	31.27	31.14	30.8	30.87	31.25	31.17	31.1	31.13	31.42	31.14	32.31
0.05	320	34.57	34.39	34.27	34.3	34.55	34.58	34.27	34.44	34.67	34.51	35.14
0.1	160	36.63	36.49	36.29	36.32	36.64	36.42	36.49	36.35	36.74	36.37	37.34
0.25	64	40.83	40.63	40.47	40.51	40.84	40.46	40.59	40.29	40.81	40.31	41.58
0.5	32	45.88	45.66	45.41	45.46	45.87	45.39	45.57	45.13	45.58	45.15	46.21
	HSI IV											
0.001	16000	28.11	27.94	27.77	27.81	28.06	28.08	27.88	28.07	28.14	28.16	28.87
0.005	3200	30.44	30.32	30.2	30.17	30.43	30.27	30.03	30.26	30.22	30.28	31.03
0.01	1600	31.41	31.29	31.12	31.09	31.39	31.32	31.1	31.29	31.57	31.43	31.95
0.05	320	34.46	34.3	34.09	34.04	34.45	34.41	34.27	34.25	34.62	34.28	34.09
0.1	160	36.43	36.29	36.01	36.04	36.43	36.25	36.49	36.19	36.51	36.2	37.24
0.25	64	40.08	39.93	39.68	39.64	40.07	39.92	40.59	39.8	40.19	39.84	40.79
0.5	32	44.51	44.47	44.32	44.29	44.5	44.31	44.46	44.22	44.63	44.22	45.02

Table 5. Performance evaluation of different HSICAs with proposed 3D-LMZC on various HS image datasets for different image quality parameter

	HSI II								
	3D-SPIHT			**3D-NLS**			**3D-LMZC**		
Bit Rate	**HS Image Quality (Coding Efficiency/PSNR)**	**NSC**	**RC**	**HS Image Quality (Coding Efficiency/PSNR)**	**NSC**	**RC**	**HS Image Quality (Coding Efficiency/PSNR)**	**NSC**	**RC**
0.001	26.75	3614	1498	26.61	3219	1488	27.89	3887	1607
0.005	29.31	17157	7091	29.25	16689	7002	30.18	17995	7257
0.01	30.19	28877	12287	30.15	28111	12347	31.11	29007	12928
0.05	33.61	168978	46897	33.59	168842	46888	34.52	170111	47025
0.1	35.44	311954	140905	35.41	311021	139852	36.21	324569	145958
	HSI III								
	3D-SPIHT			**3D-NLS**			**3D-LMZC**		
	HS Image Quality (Coding Efficiency/PSNR)	**NSC**	**RC**	**HS Image Quality (Coding Efficiency/PSNR)**	**NSC**	**RC**	**HS Image Quality (Coding Efficiency/PSNR)**	**NSC**	**RC**
0.001	27.49	4712	924	27.28	4517	908	27.97	4664	1011
0.005	30.09	8419	5111	30.03	8301	5074	31.17	8357	5419
0.01	31.14	14218	8491	31.1	14109	8389	32.31	15001	9715
0.05	34.39	89254	54151	34.27	87452	52814	35.14	89217	54191
0.1	36.49	112327	74259	36.49	112327	74259	37.34	119251	76021

Table 6. Performance evaluation of different HSICAs with proposed 3D-LMZC on various HS image -datasets for Bjøntegaard Delta PSNR

HS Image	Compression Algorithm 1 [24]	Compression Algorithm 2 [25]	Compression Algorithm 3 [26]	Compression Algorithm 4 [27]	Compression Algorithm 5 [28]	Compression Algorithm 6 [29]	Compression Algorithm 7 [30]	Compression Algorithm 8 [31]	Compression Algorithm 9 [32]	Compression Algorithm 10 [33]
HSI I	0.8496	0.8515	0.7550	0.9286	0.8758	1.1216	1.1578	1.1365	0.8654	1.0985
HSI II	0.7591	0.8976	1.0492	1.0393	0.7820	0.8804	0.9675	0.9060	0.7474	0.8830
HSI III	0.7630	0.9405	1.1634	1.1146	0.7750	0.9273	1.0136	0.9976	0.6526	0.9709
HSI IV	0.4672	0.6001	0.7894	0.8080	0.4820	0.5872	0.6173	0.6509	0.4168	0.5973

Table 7. Performance evaluation of different HSICAs with proposed 3D-LMZC on various HS image datasets for coding memory

Bit Rate	Compression Algorithm 1 [24]	Compression Algorithm 2 [25]	Compression Algorithm 3 [26]	Compression Algorithm 4 [27]	Compression Algorithm 5 [28]	Compression Algorithm 6 [29]	Compression Algorithm 7 [30]	Compression Algorithm 8 [31]	Compression Algorithm 9 [32]	Compression Algorithm 10 [33]	3D-LMZC [Proposed]
	HSI I										
0.001	26.67	37.33	41.07	43.54	28.08	4096	8192	96	2318	0	2176
0.005	102.3	99.21	109.2	121.4	89.33	4096	8192	96	2318	0	2176
0.01	232.2	222.7	249.4	257.1	202.4	4096	8192	96	2318	0	2176
0.05	1084	1041	1108	1142	991.7	4096	8192	96	2318	0	2176
0.1	1846	1931	2007	2048	1756	4096	8192	96	2318	0	2176
0.25	4571	4463	4519	4548	4289	4096	8192	96	2318	0	2176
0.5	8644	8555	8694	8705	8514	4096	8192	96	2318	0	2176
	HSI II										
0.001	22.58	21.51	23.17	27.54	22.69	4096	8192	96	2318	0	2176
0.005	91.12	98.91	111.9	145.7	91.29	4096	8192	96	2318	0	2176
0.01	265.9	267.8	302.8	322.1	266.4	4096	8192	96	2318	0	2176
0.05	982.4	1036	1254	1351	985.4	4096	8192	96	2318	0	2176
0.1	2219	2326	2521	2532	2229	4096	8192	96	2318	0	2176
0.25	5450	5611	5854	6001	5464	4096	8192	96	2318	0	2176
0.5	10005	9981	10841	10925	9832	4096	8192	96	2318	0	2176
	HSI III										
0.001	25.28	24.94	26.31	27.02	25.06	4096	8192	96	2318	0	2176
0.005	101.2	105.8	112.7	117.1	101.5	4096	8192	96	2318	0	2176
0.01	205.1	218.9	237.1	255.1	208.6	4096	8192	96	2318	0	2176
0.05	1108	1149	1224	1307	1136	4096	8192	96	2318	0	2176
0.1	1855	1808	1984	2078	1854	4096	8192	96	2318	0	2176
0.25	4401	4449	4617	4891	4412	4096	8192	96	2318	0	2176
0.5	7918	7805	8007	8191	7935	4096	8192	96	2318	0	2176
	HSI IV										
0.001	24.67	22.41	27.04	30.21	24.55	4096	8192	96	2318	0	2176
0.005	100.8	105.5	127.9	137.1	101.1	4096	8192	96	2318	0	2176
0.01	210.9	229.9	264.1	294.1	214.4	4096	8192	96	2318	0	2176
0.05	1088	1212	1328	1408	1106	4096	8192	96	2318	0	2176
0.1	1970	2083	2254	2401	1980	4096	8192	96	2318	0	2176
0.25	4867	5047	5287	5408	4878	4096	8192	96	2318	0	2176
0.5	9078	8488	8741	8911	9093	4096	8192	96	2318	0	2176

Table 8. Performance evaluation of different HSICAs with proposed 3D-LMZC on various HS image datasets for encoding time

Bit Rate	Compression Algorithm 1 [24]	Compression Algorithm 2 [25]	Compression Algorithm 3 [26]	Compression Algorithm 4 [27]	Compression Algorithm 5 [28]	Compression Algorithm 6 [29]	Compression Algorithm 7 [30]	Compression Algorithm 8 [31]	Compression Algorithm 9 [32]	Compression Algorithm 10 [33]	3D-LMZC [Proposed]
	HSI I										
0.001	3.99	4.06	1.71	2.09	5.94	2.67	14.18	5.91	3.17	3.24	26.17
0.005	9.85	9.73	3.08	5.14	8.2	2.78	61.33	8.35	3.35	4.83	99.47
0.01	20.45	29.93	9.19	11.25	10.99	3.25	73.64	9.26	4.41	5.97	114.2
0.05	222.2	303.4	91.3	108.9	94.36	5	90.57	19.45	5.49	12.18	149.2
0.1	1163	1297	256.9	311.5	762.6	7.31	102.5	34.74	7.94	19.55	164.7
0.25	6234	6871	6008	6521	4358	13.35	120.8	68.15	14.02	40.25	189.1
0.5	17995	18742	19124	19952	19551	24.12	151.3	122.5	26.03	74.87	209.3
	HSI II										
0.001	3.42	4.33	1.89	2.04	5.94	2.35	15.97	5.73	2.47	2.94	22.81
0.005	9.84	5.85	2.21	2.57	8.5	2.71	75.93	7.36	3.87	6.44	121.9
0.01	22.53	9.41	4.04	4.57	10.83	2.88	90.43	16.99	4.29	10.28	166.1
0.05	250.3	134.4	81.5	97.5	131.5	4.14	106.55	27.4	5.02	16.02	181.2
0.1	966.7	570.8	288.4	301.9	632.6	6.04	125.87	36.27	7.21	18.42	201.7
0.25	4973	3032	2988	3001	4100	10.24	134.4	96.34	12.21	56.67	217.8
0.5	12007	10112	9998	9845	12975	17.25	154.41	177.73	18.95	67.74	241.5
	HSI III										
0.001	4.08	4.03	3.51	3.84	5.85	2.07	15.97	5.68	2.76	3.19	21.81
0.005	9.12	5.96	4.78	5.09	7.87	2.89	75.93	7.78	3.28	4.74	124.1
0.01	20.18	9.7	8.15	8.84	11.64	3.34	90.43	8.55	4.01	7.52	147.5
0.05	204.3	125.2	101.2	111.2	89.77	4.57	106.55	19.48	5.31	22.88	179.2
0.1	1183	775.8	624.9	694.8	835.9	5.91	125.87	32.46	6.47	30.14	197.8
0.25	8499	5151	4517	4872	6309	10.41	134.14	70.4	11.91	43.49	211.4
0.5	29849	18383	15874	16247	23861	16.19	154.41	125.42	17.09	72.62	261.3
	HSI IV										
0.001	4.56	5.6	4.94	5.31	7.23	2.39	6.03	5.74	2.89	2.82	11.57
0.005	15.24	6.23	5.54	5.87	8.15	2.81	11.53	7.53	3.34	4.44	20.14
0.01	21.67	10.2	9.17	9.87	12.64	3.18	18.44	8.93	3.98	5.64	37.18
0.05	269.6	130.4	109.2	117.2	98.12	4.3	22.64	18.61	4.88	13.02	42.17
0.1	1336	893.4	767.2	808.1	882.3	6.11	25.53	32.45	6.41	18.18	49.81
0.25	8435	5133	4581	4851	5501	10.35	34.5	69.66	11.38	36.3	61.74
0.5	27917	17945	15547	16027	18818	17.43	65.13	125.19	19.01	66.91	127.2

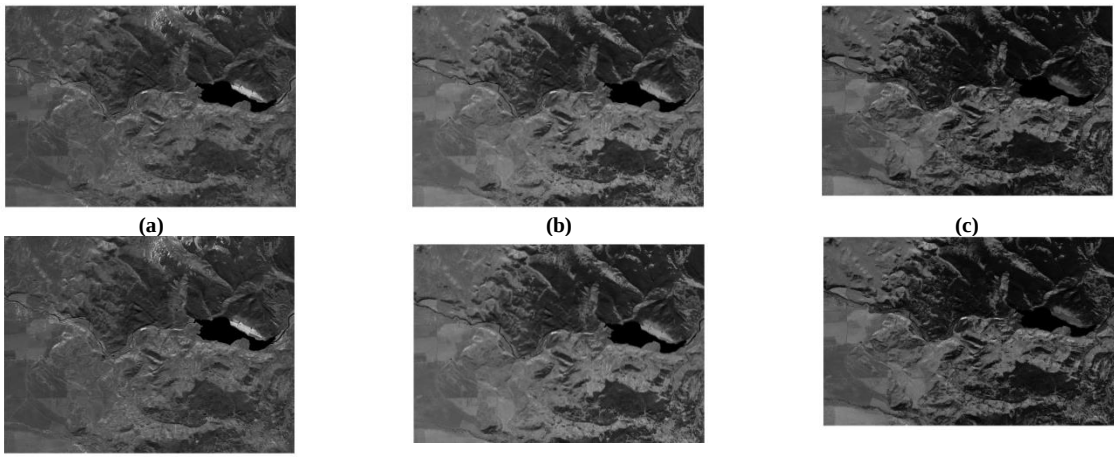
Table 9. Performance evaluation of different HSICAs with proposed 3D-LMZC on various HS image -datasets for decoding time

Bit Rate	Compression Algorithm 1 [24]	Compression Algorithm 2 [25]	Compression Algorithm 3 [26]	Compression Algorithm 4 [27]	Compression Algorithm 5 [28]	Compression Algorithm 6 [29]	Compression Algorithm 7 [30]	Compression Algorithm 8 [31]	Compression Algorithm 9 [32]	Compression Algorithm 10 [33]	3D-LMZC [Proposed]
	HSI I										
0.001	1.78	2.92	1.51	1.89	1.59	2.08	12.79	2.48	2.21	3.02	11.19
0.005	5.18	5.25	2.94	3.51	2.41	2.43	48.29	3.86	2.68	4.65	21.94
0.01	10.78	14.31	8.07	10.29	4.51	2.68	57.16	4.04	3.08	5.61	32.07
0.05	172.7	236.2	79.22	94.25	84.75	4.02	69.23	12.01	4.34	11.79	38.34
0.1	1081	1078	219.5	319.8	762.11	6.24	77.57	21.79	6.71	18.36	41.68
0.25	6012	6305	5129	5598	4703	11.68	90.45	50.91	12.02	37.86	44.59
0.5	17597	18534	17459	19049	15400	22.65	100.5	96.84	25.07	69.02	54.24
	HSI II										
0.001	1.87	1.52	1.71	1.93	1.46	1.4	12.18	2.18	1.61	2.79	3.17

0.005	5.4	2.45	1.89	2.21	2.77	2.49	66.24	3.21	3.01	6.05	28.4
0.01	10.01	4.92	3.74	4.02	3.86	2.71	81.48	6.23	3.27	10.04	32.8
0.05	207.2	127.8	74.9	81.2	130.1	3.38	94.49	14.94	3.94	11.35	37.1
0.1	887.6	717.5	249.7	274.7	614.3	5.98	106.8	23.01	6.64	17.81	39.7
0.25	4796	3129	2791	2548	4140	6.74	113.86	58.62	7.18	47.06	42.8
0.5	11898	9954	9549	9358	12299	14.7	125.56	120.33	15.34	60.13	48.2
	HSI III										
0.001	1.74	1.39	3.17	3.42	1.32	1.89	8.43	4.1	2.11	3.02	3.14
0.005	5.13	2.24	4.31	4.61	2.44	2.47	66.02	6.02	2.74	3.99	18.7
0.01	12.51	5.18	7.67	8.05	5.14	2.69	84.96	7.06	3.02	6.33	27.9
0.05	160.3	114.7	91.8	94.8	80.01	4.46	92.68	14.84	5.19	18.56	31.8
0.1	1474	760.5	588.2	601.8	827.8	5.59	104.98	21.49	6.37	27.82	35.2
0.25	8587	5832	4291	4481	6549	9.27	115.94	48.95	10.34	39.95	41.7
0.5	26948	15672	15047	15841	23161	14.97	141.97	114.52	16.68	67.23	48.5
	HSI IV										
0.001	2.41	1.64	4.47	4.78	1.73	2.02	5.27	2.1	2.24	2.74	2.07
0.005	9.57	2.33	4.98	5.11	2.55	2.34	8.26	2.88	2.47	4.28	2.67
0.01	12.68	5.23	8.07	8.34	6.11	2.89	14.44	3.91	3.23	5.41	3.87
0.05	226.5	120.5	91.9	100.1	89.08	3.74	19.5	11.48	4.29	11.36	5.88
0.1	1241	829.1	698.5	721.8	866.3	5.96	21.07	21.02	6.57	17.22	6.24
0.25	9067	4536	4009	4219	5494	6.62	29.65	48.91	7.08	33.79	8.14
0.5	25042	17677	14854	15274	18136	12.03	55.03	92.97	12.87	62.31	14.8

Table 10. Short comparative analysis between the different HSICAs

HSICA	Ref	Major Contribution
3D-SPIHT	[25]	HSICA initially generates the embedded output through the utilization of zerotree coding. There are three connected lists in it.
3D-SPECK	[24]	Being slightly more efficient in terms of coding and has a lower level of complexity than 3D-SPIHT, as well as having two linked lists.
3D-FSPIHT	[27]	It uses a fast search method to fetch the significant zerotree in the transformed HS image.
3D-SDB-SPIHT	[28]	Utilizes distinct descendant-based set partitioning across hierarchical trees in order to enhance the efficiency of the coding process
3D-WBTC	[28]	Uses the best features of zerotree and zero block cube HSICA to obtain high coding efficiency at low bit rates
3D-NLS	[30]	It makes use of the markers in order to simplify the coding process and necessitates the use of a fixed coding memory.
3D_ZM-SPECK	[33]	The coefficient tracking did not require any coding memory to be carried out.
3D-LBCSPC	[36]	It uses the wavelet transform's pyramid hierarchy property to achieve excellent coding efficiency to achieve the desired result.



(a) (b) (c)

(d) (e) (f)

Fig 1 : HS Image 'HSI II' Before compression process (a) Frame 50 (b) Frame 100 (c) Frame 150
Reconstructed HS Image 'HSI II' with CR=32 (Bit Rate = 0.5) (d) Frame 50 (e) Frame 100 (f) Frame 150

6. Conclusion

The 3D-LMZC is a non-list version of 3D-SPIHT and 3D-FSPIHT, with low coding memory requirements and high coding gain. The curvelet transform represents edges and other singularities along the curvelet much more efficiently than the wavelet transform. The coding memory consumption greatly decreases compared to other zerotree HSICAs, such as 3D-SPIHT and 3D-NLS. The intricacy of the coding just a little bit increased. The low coding memory needs of 3D-LMZC more than makeup for the algorithm's slightly higher level of complexity. Applying the bandlet, contourlet, and ripplet transforms can improve the coding gain. By utilizing fewer markers, the load on coding memory can be further decreased. In the future, we will explore an efficient compression algorithm using the abovementioned mathematical transform for high coding efficiency.

Acknowledgement

The authors thank Integral University for providing MCN "IU/R&D/2024-MCN0003222".

Declarations

Authors' Contributions

Tripathi and Bajpai developed the algorithm and algorithms simulation and prepared the manuscript.

Consent for Publication

All authors agreed on the final approval of the version to be published.

Availability of Data and Material

The data and material are available within the manuscript.

References

[1] D. Chutia et al., "Hyperspectral Remote Sensing Classifications: A Perspective Survey," *Transactions in GIS*, vol. 20, no. 4, pp. 463-490, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[2] Sneha, and Ajay Kaul, "Hyperspectral Imaging and Target Detection Algorithms: A Review," *Multimedia Tools and Applications*, vol. 81, pp. 44141-44206, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[3] Gemine Vivone, "Multispectral and Hyperspectral Image Fusion in Remote Sensing: A Survey," *Information Fusion*, vol. 89, pp. 405-417, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[4] Ajay Kaul, and Sneha Raina, "Support Vector Machine versus Convolutional Neural Network for Hyperspectral Image Classification: A Systematic Review," *Concurrency and Computation: Practice and Experience*, vol. 34, no. 15, pp. 1-35, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[5] Divya Sharma, Y.K. Prajapati, and R. Tripathi, "Success Journey of Coherent PM-QPSK Technique with its Variants: A Survey," *IETE Technical Review*, vol. 37, no. 1, pp. 36-55, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[6] R. Nagendran, and A. Vasuki, "Hyperspectral Image Compression Using Hybrid Transform with Different Wavelet-Based Transform Coding," *International Journal of Wavelets, Multiresolution and Information Processing*, vol. 18, no. 1, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[7] Sara Álvarez-Cortés, Naoufal Amrani, and Joan Serra-Sagristà, "Low Complexity Regression Wavelet Analysis Variants for Hyperspectral Data Lossless Compression," *International Journal of Remote Sensing*, vol. 39, no. 7, pp. 1971-2000, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[8] Rui Li, Zhibin Pan, and Yang Wang, "The Linear Prediction Vector Quantization for Hyperspectral Image Compression," *Multimedia Tools and Applications*, vol. 78, pp. 11701-11718, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[9] Daniel Báscones, Carlos González, and Daniel Mozos, "Hyperspectral Image Compression Using Vector Quantization, PCA and JPEG2000," *Remote Sensing*, vol. 10, no. 6, pp. 1-13, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[10] Daniel Báscones, Carlos González, and Daniel Mozos, "An FPGA Accelerator for Real-Time Lossy Compression of Hyperspectral Images," *Remote Sensing*, vol. 12, no. 16, pp. 1-20, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[11] K.S. Gunasheela, and H.S. Prasantha, "Compressive Sensing Approach to Satellite Hyperspectral Image Compression," *Information and Communication Technology for Intelligent Systems*, Smart Innovation, Systems and Technologies, vol. 106, pp. 495-503, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[12] Lefei Zhang et al., "Compression of Hyperspectral Remote Sensing Images by Tensor Approach," *Neurocomputing*, vol. 147, pp. 358-363, 2015. [CrossRef] [Google Scholar] [Publisher Link]

[13] Raúl Guerra et al., "A Hardware-Friendly Hyperspectral Lossy Compressor for Next-Generation Space-Grade Field Programmable Gate Arrays," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 12, no. 12, pp. 4813-4828, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[14] V.K. Bairagi, A.M. Sapkal, and M.S. Gaikwad, "The Role of Transforms in Image Compression," *Journal of the Institution of Engineers (India): Series B*, vol. 94, pp. 135-140, 2013. [CrossRef] [Google Scholar] [Publisher Link]

[15] Jillela Malleswara Rao et al., "Hyperspectral and Multispectral Data Fusion Using Fast Discrete Curvelet Transform for Urban Surface Material Characterization, *Geocarto International*, vol. 37, no. 7, pp. 2018-2030, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[16] Tong Qiao et al., "Effective Denoising and Classification of Hyperspectral Images Using Curvelet Transform and Singular Spectrum Analysis," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 55, no. 1, pp. 119-133, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[17] Chuna Wu, Xiaoyanb Ma, and Wenboc Wang, "Hyperspectral Image Denoise Based on Curvelet Transform Combined with Weight Coefficient Method," *Journal of Intelligent & Fuzzy Systems*, vol. 37, no. 4, pp. 4425-4429, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[18] Amit Kumar Pandey et al., "Analysis of Noise Immunity for Wide OR Footless Domino Circuit Using Keeper Controlling Network," *Circuits, Systems, and Signal Processing*, vol. 37, pp. 4599-4616, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[19] T. Santosh Kumar, and Suman Lata Tripathi, "Low Power and Suppressed Noise 6T, 7T SRAM Cell Using 18 nm FinFET," *Wireless Personal Communications*, vol. 130, pp. 103-112, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[20] Lynda Inouri et al., "A Fast and Efficient Approach for Image Compression Using Curvelet Transform," *Sensing and Imaging*, vol. 19, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[21] Manas Saha, Mrinal Kanti Naskar, and B.N. Chatterji, "Advanced Wavelet Transform for Image Processing-A Survey," *Information, Photonics and Communication*, *Lecture Notes in Networks and Systems*, vol. 79, pp. 185-194, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[22] Muhammad Azhar Iqbal, Muhammad Younus Javed, and Usman Qayyum, "Curvelet-Based Image Compression with SPIHT," *International Conference on Convergence Information Technology*, Gwangju, Korea (South), pp. 961-965, 2007. [CrossRef] [Google Scholar] [Publisher Link]

[23] Shrish Bajpai et al., "Curvelet Transform Based Compression Algorithm for Low Resource Hyperspectral Image Sensors," *Journal of Electrical and Computer Engineering*, vol. 2023, no. 1, pp. 1-18, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[24] Xiaoli Tang, and William A. Pearlman, *Three-Dimensional Wavelet-Based Compression of Hyperspectral Images*, Hyperspectral Data Compression, pp. 273-308, 2006. [CrossRef] [Google Scholar] [Publisher Link]

[25] Xiaoli Tang, and W.A. Pearlman, "Lossy-to-Lossless Block-Based Compression of Hyperspectral Volumetric Data," *International Conference on Image Processing*, Singapore, vol. 5, pp. 3283-3286, 2004. [CrossRef] [Google Scholar] [Publisher Link]

[26] Xiaoying Song et al., "Three-Dimensional Separate Descendant-Based SPIHT Algorithm for Fast Compression of High-Resolution Medical Image Sequences," *IET Image Processing*, vol. 11, no. 1, pp. 80-87, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[27] Zhong Cuixiang, and Huaung Minghe, "An Effective Improvement on 3D SPIHT," *International Conference on Image Analysis and Signal Processing*, Huangzhou, China, pp. 1-14, 2012. [CrossRef] [Google Scholar] [Publisher Link]

[28] Shrish Bajpai et al., "3D Wavelet Block Tree Coding for Hyperspectral Images," *International Journal of Innovative Technology and Exploring Engineering*, vol. 8, no. 6c, pp. 64-68, 2019. [Google Scholar] [Publisher Link]

[29] Ruzelita Ngadiran et al., "Efficient Implementation of 3D Listless SPECK," *International Conference on Computer and Communication Engineering*, Kuala Lumpur, Malaysia, pp. 1-4, 2010. [CrossRef] [Google Scholar] [Publisher Link]

[30] V.K. Sudha, and R. Sudhakar, "3D Listless Embedded Block Coding Algorithm for Compression of Volumetric Medical Images," *Journal of Scientific and Industrial Research*, vol. 72, no. 12, pp. 735-738, 2013. [Google Scholar] [Publisher Link]

[31] Shrish Bajpai et al., "Low Memory Block Tree Coding for Hyperspectral Images," *Multimedia Tools and Applications*, vol. 78, pp. 27193-27209, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[32] Shrish Bajpai, "Low Complexity Block Tree Coding for Hyperspectral Image Sensors," *Multimedia Tools and Applications*, vol. 81, pp. 33205-33232, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[33] Shrish Bajpai et al., "A Low Complexity Hyperspectral Image Compression Through 3D Set Partitioned Embedded Zero Block Coding," *Multimedia Tools and Applications*, vol. 81, pp. 841-872, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[34] De Rosal Igantius Moses Setiadis, "PSNR vs SSIM: Imperceptibility Quality Assessment for Image Steganography," *Multimedia Tools and Applications*, vol. 80, pp. 8423-8444, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[35] Nabajeet Barman, Maria G. Martini, and Yuriy Reznik, "Revisiting Bjontegaard Delta Bitrate (BD-BR) Computation for Codec Compression Efficiency Comparison," *Proceedings of the 1st Mile-High Video Conference*, pp. 113-114, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[36] Shrish Bajpai, "3D-Listless Block Cube Set-Partitioning Coding for Resource Constraint Hyperspectral Image Sensors," *Signal, Image and Video Processing*, vol. 18, pp. 3163-3178, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[37] Yimy E. García-Vera et al., "Hyperspectral Image Analysis and Machine Learning Techniques for Crop Disease Detection and Identification: A Review," *Sustainability*, vol. 16, no. 4, pp. 1-31, 2024. [CrossRef] [Google Scholar] [Publisher Link]