SSRG International Journal of Electronics and Communication Engineering
ISSN: 2348-8549/ https://doi.org/10.14445/23488549/IJECE-V12I4P125

Volume 12 Issue 4, 265-281, April 2025
© 2025 Seventh Sense Research Group®

*Original Article*

# 3D Single List Set Partitioning in Hierarchical Trees For Onboard Hyperspectral Image Sensors

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: 18 February 2025 Revised: 20 March 2025 Accepted: 19 April 2025 Published: 29 April 2025

***Abstract*** *- The 3D-Set Partitioning in Hierarchical Trees (3D-SPIHT) has a moderate coding complexity and average coding memory requirement. It provides an embedded compressed bit stream that can be easily decoded at different data rates. These characteristics contribute to the algorithm's overall attractiveness. Unfortunately, due to the fact that it uses three linked lists to record the significance status of the coefficients, it requires an extremely large amount of computation memory. The random access to these lists also results in the 3D-SPIHT having a memory management system that is somewhat complicated. This manuscript presents a new compression algorithm called 3D-Single List SPIHT (3D-SLS). The memory requirements for the proposed 3D-SLS HSICA are extremely low since it requires around several folds lower memory requirement than the original 3D-SPIHT required. This is accomplished by using a single list in conjunction with two state mark bitmaps, as opposed to the three lists that 3D-SPIHT have. In addition, the proposed 3D-SLS offers a simpler memory management system because coefficients are never deleted from the list after they have been added to the list. In addition, the list size can be predetermined, which allows one to circumvent the issue of dynamic memory allocation. Because of these memory savings and management simplifications, the 3D-SLS compression algorithm is an excellent candidate for implementation in hardware for the onboard hyperspectral image sensors.*

***Keywords*** *- Lossy hyperspectral image compression, Wavelet transform coding, Zero tree, Set Partitioned Compression Algorithm.*

## 1. Introduction

Since the late 1990s, researchers have advocated using spectroscopy, spectral analysis, and colour photometry to characterise the composition of space objects [1, 2]. In recent years, hyperspectral (HS) remote sensing has developed into a reliable device for Earth studies [3]. Due to the fact that HS images are capable of collecting a vast amount of data in the spectral and spatial domains, these instruments have found use in a wide variety of sectors such as agriculture [4], biomedical [5], climate [6], corrosion (nuclear, steel structure) [7, 8], mineralogy [9], surgery [10] etc. In the last few decades, HS image compression has been one of the most popular research fields. It has attracted multitudinous scholars to devote great efforts to improving the performance of the HyperSpectral Image Compression Algorithm (HSICA) [11-15]. HSICA is an important factor used to improve HS image sensor performance by reducing coding complexity, lowering power consumption and minimising data transmission time [16].

HSICAs are broadly divided into different categories based on two parameters: HS image data loss and coding process. The HSICAs are divided into three sub-categories named lossless, lossy and near-lossless based on the HS image data loss [17]. In the same way, based on the coding process is divided into seven sub-categories named Predictive Coding (PC) [18], Vector Quantization (VQ) [19], Compressive Sensing (CS) [20], Tensor Decomposition (TD) [21], learning based coding (LC) [22], Transform Coding (TC) [23] and hybrid HSICA [24].

The PC-based HSICAs use the predictor to determine the spectral correlation that exists between the continuous spectral frames of HS images. Entropy coding methods, such as Huffman coding or arithmetic coding, are utilized to encode the prediction inaccuracy. The techniques that rely on prediction are reliant on data (specifically, the compression ratio is contingent upon the image). Furthermore, these algorithms exclusively operate with lossless compression [18].

The VQ-based HSICA accepts a 3D HS data cube as its input and produces a compressed HS image as its output. The process of Vector Quantization (VQ) involves two crucial stages: training, which entails the production of a codebook, and coding, which involves matching code vectors. Due to the fact that it employs a training method in order to produce an ideal codebook, quantization is typically employed in conjunction with transform-based or learning-based techniques.

VQ-based HSICAs have high complexity, so the principal objective of the method is to develop an efficient algorithm with fast execution [19].

The CS-based HSICA is well-known for its on-board compression algorithms because it moves the computational complexity from the encoder to the decoder. Real-time compression takes advantage of it because it can detect a relatively tiny amount of data, compress that data, send the compressed data to the receiver, and then take yet another bit of data [20].

TD-based HSICA is one of the most recent techniques for image compression, and in comparison to more conventional approaches, it offers significantly improved performance. A tensor could be considered an n-dimensional matrix, a straightforward structure to decompose. One of the TD approaches is used in conjunction with this method to store the HSI in a three-dimensional tensor (Y) [21].

Because it utilizes both machine learning and deep learning in the compression process, LC-based HSICA is currently one of the most widely used approaches. As another methodology that can anticipate pixel values, this method has always been investigated in conjunction with the prediction-based strategy. On the other hand, it possesses features that are well known to the public that automatically learn and update parameters [22].

In recent years, the most widely used method for compressing images in two dimensions, known as TC-based HSICA, has been expanded to three-dimensional formats, known as HSI compression.

This method is referred to as a transform-based technique because it converts the values of the pixels in the image into the frequency domain by applying a transformation function to each of the image's three dimensions [25, 26].

The hybrid compression algorithms are created by combining any two of the approaches described in the previous paragraphs. The coding efficiency of hybrid compression algorithms is significantly higher than that of other types of compression algorithms; however, this coding gain comes at the expense of an increased level of coding complexity [24].

A brief overview of the different HSICAs based on the coding process is covered in Table 1.

### *1.1. Contribution of this Article*

The major contributions of the present article are summarised as follows:

- It lowers down the requirement of the coding memory (through the use of a single list instead of three in state of art HSICA 3D-SPIHT) used by the compression algorithm during the tracking (insignificance or significance) of the sets or zerotree or coefficient.
- The present compression algorithm has higher coding efficiency than 3D-SPIHT and 3D-NLS. It has been clear from the result of the coding efficiency in the result section.
- It improves the embedding capacity of the HSICA, which is required to apply recent HS images.

The rest of this manuscript is structured as follows: Section 2 gives an overview of related works that were associated with the compression of hyperspectral images. Section 3 offers a detailed description of the proposed HSICA 3D-SLS employed in this study. Section 4 presents an extensive investigation through simulation studies and empirical analyses of four publicly available HS images. Finally, the conclusions are given in Section 5.

## 2. Related Work

The memory of the HS image sensors (AVIRIS and HYDICE) is limited [33]. A single HS image can be 100 MB or larger in size [34]. Therefore, between 448 and 640 high-speed photos can be saved in the onboard HS sensor that has a memory capacity of 64 gigabytes (GB) [17].

HS image compression has become a vital step as a means of reducing drains on memory storage, sensor power, and processing time [35]. The fact that TC-based HS image compression algorithms are versatile and can be used for any kind of compression [36].

### *2.1. Transform Coding Based HS Image Compression Algorithm (TC-HSICA)*

The TC-based HS image compression methods use the mathematical transform to transfer the HS image to the frequency domain [37].

The wavelet transform is a well-known mathematical transform utilized to convert the HS image from the time domain to the frequency domain [38]. Curvelet Transform [39], Karhunen-Loeve Transform [40], and Shearlet Transform [41] also give the promise of performance for the compression of HS images.

Removing spectral and spatial correlation is contingent upon the specific domain in which it is implemented [42]. Compression using the transform-based method encompasses a series of processes exhibiting potential variations across distinct compression algorithms [43].

**Table 1. Short comparative analysis between the types of HSICA based on the coding process**

| Type | Ref | Advantage | Limitation | Example |
|---|---|---|---|---|
| PC based HSICA | [18] | Easy to implement on hardware platforms | HS image will distort if all HS image data is not received | C-DPCM-RNN [27] |
| VQ based HSICA | [19] | Execution time is less for the lossless compression process | Codebook change for each HS image, which increases the cost of computation | VQPCA [28] |
| TC based HSICA | [23] | It can be implemented for lossy and lossless compression processes and has an error-tolerance mechanism | A lot of computational calculation (logical and arithmetical) for the encoding and decoding process | 3D-ZM-BCP-SPECK [17] |
| CS-based HSICA | [20] | Coding complexity shifts from encoder to decoder end | Very expensive decompression process and identification of sensing matrix | SHSIR [29] |
| TD based HSICA | [21] | Best compression performance with a low run time | High dependency of the compression performance depends on HS image data | PSO-NTD [30] |
| LC-based HSICA | [22] | Very high coding efficiency than other type of HSICAs | The complexity of these HSICAs is very high, with complex implementation | ANN [31] |
| Hybrid HSICA | [24] | Best performance at the lossless HS image compression | Implementation of the hardware is very complicated | CNN-LZMA [32] |

**Table 2. A short comparative analysis between different set partitioned HSICAs**

| HSICA | Ref | Year | Partition Type | List | CM | Embeddedness |
|---|---|---|---|---|---|---|
| 3D-SPECK | [48] | 2006 | Zero Block Cube | List (2) | Variable | Y |
| 3D-SPEZBC | [49] | 2007 | | List (2) | Variable | Y |
| 3D-LSK | [50] | 2010 | | Listless | Fixed | Y |
| 3D-ZM-SPECK | [51] | 2022 | | Listless | Zero | Y |
| 3D-BCP-ZM-SPECK | [17] | 2023 | | Listless | Fixed | Y |
| 3D-M-ZM-SPECK | [52] | 2023 | | Listless | Zero | Y |
| FrWF based ZMSPECK | [53] | 2023 | | Listless | Zero | Y |
| 3D-SPIHT | [54] | 2004 | Zero Tree | List (3) | Variable | Y |
| 3D-NLS | [55] | 2013 | | Listless | Fixed | Y |
| 3D-LEZSPC | [39] | 2023 | | Listless | Fixed | Y |
| 3D-BPEC | [56] | 2023 | | Array (6) | Variable | Y |
| 3D-MELS | [57] | 2023 | | Listless | Fixed | Y |
| 3D-WBTC | [58] | 2019 | Zero Block Cube Tree | List (3) | Variable | Y |
| 3D-LMBTC | [59] | 2019 | | Listless | Fixed | Y |
| 3D-M-WBTC | [60] | 2019 | | List (3) | Variable | Y |
| 3D-LCBTC | [61] | 2022 | | List (2) | Fixed | Y |
| 3D-LBCTC | [62] | 2022 | | Listless | Fixed | Y |

The forward transform involves the application of a transformation function to either the spatial or spectral domain or both. This is followed by a decorrelation process, resulting in the generation of coefficients [44-47].

### *2.2. Set Partitioned-Based Hyperspectral Image Compression Algorithms*

Set partitioned-based HSICAs are the special type of TC HSICAs that use the set structure of the transform HS image to compress the HS image. It has several advantages over other TC HSICA, such as embeddedness, low coding complexity, ease of implementation on hardware, moderate coding efficiency, and the ability to work with another type of HSICA to create hybrid compression algorithms. Further, set partition-based HSICAs can be sub-categories based on linked lists (list-based, listless and array) or partition methods (zerotree, zeroblock cube, zeroblock cube tree).

3D-SPIHT [54] uses three linked lists to track sets and coefficients and has higher coding efficiency than any HSICA.

The linked lists increase the coding complexity and coding memory demand. 3D-SPECK [48] uses a different partition rule (zeroblock cube) but uses two linked lists for the same task. To achieve high coding efficiency at low bit rates, 3D-WBTC [58] is proposed. 3D-WBTC [58] uses the zeroblock cube tree partition rule to achieve high coding efficiency, as it is known that the high number of coefficients is insignificant for the top bit planes. 3D-WBTC [58] can represent eight times insignificant coefficients than 3D-SPIHT [54] through a single bit for the top bit planes.

The issue of the coding complexity and coding memory is solved by the using of state markers. But this comes with the cost of coding efficiency. 3D-LSK [50], 3D-ZM-SPECK [51], 3D-BCP-ZM-SPECK [17] and 3D-M-ZM-SPECK [52] are the listless HSICA which follow the same partition rule as 3D-SPECK (zeroblock cube) while 3D-NLS [55], 3D-MELS and 3D-LEZSPC [39] follow the same rule as 3D-SPIHT [48] (zerotree). 3D-BPEC [56] uses six arrays to track the coefficient status and follows the same partition rule as 3D-SPIHT [48]. In the same way, 3D-LMBTC [59], 3D-LCBTC [61] and 3D-LBCTC [62] use the same partition rule as 3D-WBTC [58].

The short description between state-of-the-art set partition HSICAs is covered in Table 2. The present work is motivated by the 2D version of the single list SPIHT [63], which reduces the demand for coding memory for calculating the output bit steam and has high coding efficiency. The extension of the 2D version to the 3D for the HS image is performed in the present study.

## 3. 3D-Single List Set Partitioning in Hierarchical

The 3D Single List Set Partitioning in Hierarchical Trees, also known as 3D-Single List SPIHT (3D-SLS), used a single list instead of three in 3D-SPIHT [54]. Through the use of a single list, it reduces the demand for coding memory and significantly achieves a higher coding efficiency. The reduction of coding memory is due to using the single list as a multiple list (three in 3D-SPIHT). The LIP and LSP lists comprise the eight children of a root that belongs to a SIG SoT, and this fact forms the foundation of the 3D-SLS. Therefore, the offspring can be recalculated in each bit plane pass rather than being stored in these lists. Obviously, this will make the compression algorithms more difficult with increased coding memory. However, the 3D-SLS approach significantly reduces the complexity of the memory management issue. As a result, this increase in complexity is compensated for by a lower amount of overhead associated with memory management.

An individual list referred to as the List of Root Sets (LRS) is utilised in the proposed HSICA. The HS image transforms with ‘L’ level wavelet transform, and ‘n’ is denoted as the number of bit planes present in the transform HS image. The (i, j, k) coordinates of the roots of the SOTs are stored in LRS, just as they are in LIS for 3D-SPIHT [54]. Therefore, the dimensions of LRS are identical to those of LIS. On the other hand, LRS is distinct in the following two respects:

- When a set is added to LRS, it will not be withdrawn under any circumstances. Therefore, LRS can be implemented as a straightforward 1D array that is accessed in a sequential fashion using the First In First Out (FIFO) method. In contrast, the LIS needs to be built as a linked list that can only be accessed in a haphazard fashion due to the ongoing process of elements being added to and removed from it.
- The type field (zerotree type) is utilised in various ways. A set is considered to be of type A if it has not been tested or does not have any SIG SOT, while a set is considered of type B if it has one or more SIG SOT. To put it another way, a set in LRS begins its life as type A, but it transitions to type B when one of its SOTs is converted into an SIG.

In addition, 3D-SLS substitutes state markers for the LIP and LSP lists, which are responsible for the majority of the total memory use. To be more explicit, each coefficient in the $LLL_L$ sub-band is provided by a single status bit denoted by the number ‘$\lambda$’. This bit is set to 0 when the pixel is first created and is changed to 1 when it just became SIG. The type of coefficients can be determined based on the values of two status bits referred to as '$\omega$'. These values are used to generate the coefficients for all other sub-bands. After that, it sets the LRS to the coordinates (i, j, k) of the pixels in the $LLL_L$ sub-band with offspring that are type A sets and initializes it.

- $\omega = 0$ New Insignificant Coefficient (NIC): The NIC is the coefficient not tested against the current threshold. It may be a member of the progeny of type A sets in the LRS that have recently become significant in the last bit-plane pass.
- $\omega = 1$ Visited Insignificant Coefficient (VIC): The VIC is the coefficient that is tested as an insignificant coefficient against the previous bit-plane. It is possible that it is related to the sets of type B that were found in LRS. It is important to note that the (i, j, k) coordinates of these VICs are stored by the 3D-SPIHT [54] in the LIP list, which is then coded during the sorting phase.
- $\omega = 2$ New Significant Coefficient (NSC): The NSC is the significant coefficient during the current bit-plane pass.
- $\omega = 3$ Visited Significant Coefficient (VSC): The VSC is the coefficient that nis tested significantly in the previous bit-plane pass. It's possible that it's related to the type B sets that LRS spawned. Note that the 3D-SPIHT [54] saves the (i, j, k) coordinates of these VSC coefficients in LSP and that these coordinates are coded in the refinement pass.

Initialization is the first step of the 3D-SLS, which is followed by numerous runs of bit-plane coding till the bit budget is available. When 3D-SLS is being initialised, the first thing it does is compute and output the top bit-plane followed by the lower bit-plane accordingly. Following this, it initialises the LRS by assigning it the coordinates (i, j, k) of the coefficients in the $LLL_L$ sub-band that have offspring that are type A sets, and then it sets the LRS to those values.

The coding of a coefficient $\Gamma_{i,j,k}$ in $LLL_L$ involves determining whether or not it is significant in relation to the bit plane 'n'. If $\Gamma_{i,j,k}$ is still insignificant, then the '0' bit is outputted to the bit stream, while if it becomes significant against the current bit, then the '1' bit is outputted to the bit stream and the status of the $\lambda_{i,j,k}$ is updated to '1'. Following this, the sets of LRS, which are sets of type A, are successively processed in the following manner: the SOT of the set is evaluated to see whether or not it is significant in relation to 'n'. If SOT is insignificant to the current threshold, then 0' bit is outputted. On the other hand, if the SOT is significant, then the value '1' is outputted, the set's type is changed to a type B set, and each of the set's eight children (which is an NIC) is examined for significance with regard to n. This happens only if the SOT is significant. If it is still insignificant, then the bit stream will be identified as a VIC and the value '0' will be outputted to the stream.On the other hand, if the coefficient becomes significant, and '1' and its sign bit are outputted to the bit-stream, it is marked as an NSC. Finally, if the set has grandkids, those four descendants are appended to the end of LRS as type A sets to be coded in the current bit-plane pass.

If the set does not have grandchildren, the process ends here. The coefficients are encoded in $LLL_L$ format at the beginning of each of the following bit-plane coding passes. The values of these coefficients' status bits determine how they are encoded according to their sort of coefficient. If $\lambda_{i,j,k}$ = '0', the coefficient $\Gamma_{i,j,k}$ is insignificant, it is coded in the same way as was done in the first pass, and if $\lambda_{i,j,k}$ = '1' (significant coefficient) it is refined by outputting its $n^{th}$ bit to the bit-stream. It should be noted that the sorting and refining passes have been combined for the pixels in the $LLL_L$ sub-band, which may result in a decrease in the Peak Signal to Noise Ratio (PSNR) performance of the 3D-SLS because the information ordering has not been preserved. On the other hand, because the size of $LLL_L$ is relatively small compared to the size of the HS image as a whole, the amount of reduction is only very slight. The LRS list is then scanned twice after this step. Only the sets of type B are processed in the first scan by computing its four children; this only applies to those sets. There is a possibility that a type B collection will contain the pixels and VICs that were discovered to be significant in earlier rounds but which are currently labelled as NSC. In order to refine an NSC more in the second run through the LRS, it is identified as VSC and given that designation. On the other hand, if the pixel is a VIC, it is coded exactly in the same way as coding a NIC. It is important to take note that this phase is comparable to coding the pixels in the list LIP during the sorting process in the original 3D-SPIHT [54]. This indicates that both encoding algorithms use the same order when coding the VICs. As a result, the embedding capabilities of 3D-SLS are identical to those of 3D-SPIHT [54].

In the second run of the LRS scan, each and every set in the LRS is subjected to processing. If the set is of type A, then it is processed in precisely the same manner as it was in the initial pass of the procedure. If the set is of type B, on the other hand, only the VSCs of the set's eight offspring are refined by sending their $n^{th}$ bit to the bit-stream. This is the case because type B sets are less common. At this point, 'n' is decreased by one in preparation for the beginning of a new bit-plane coding pass. It is important to note that the process of coding the VSCs of type B sets in the second LRS scan pass is analogous to the process of refining the pixels in the list LSP during the refinement pass in the initial 3D-SPIHT [54]. This indicates that the 3D-SLS algorithm under consideration combines both the NICs and the VSCs. It has been demonstrated that the VICs have a greater influence on the reduction of distortion compared to both the NICs and the VSCs. Therefore, merging the VICs and VSCs as done may result in a decrease in PSNR, but combining the NICs and the VSCs results in a minor improvement in the PSNR of the 3D-SLS algorithm. The flow of the 3D-SLS is given by the Figure 1.

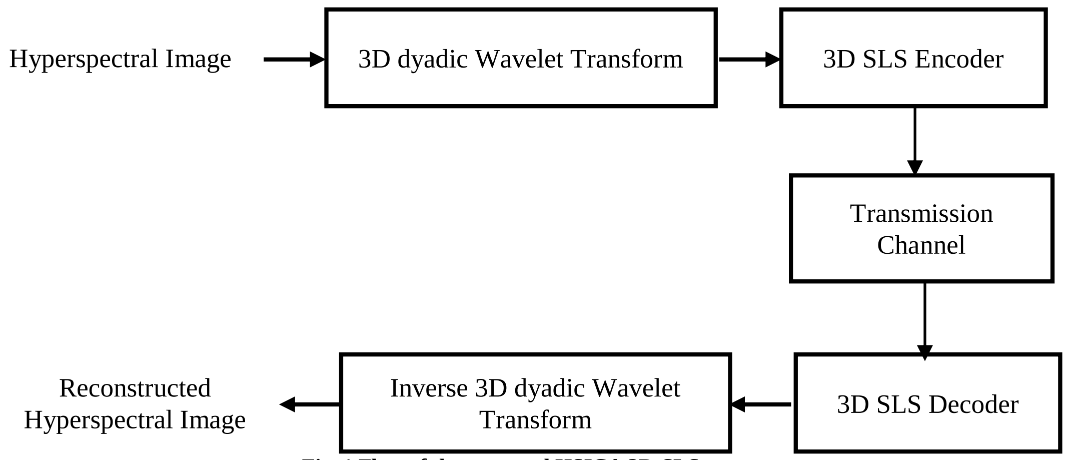
Hyperspectral Image
3D dyadic Wavelet Transform
3D SLS Encoder
Transmission Channel
3D SLS Decoder
Inverse 3D dyadic Wavelet Transform
Reconstructed Hyperspectral Image

**Fig. 1 Flow of the proposed HSICA 3D-SLS**

## 4. Simulation Result

In this section, We evaluate the compression performance (coding efficiency, coding memory and coding complexity) of our presented 3D-SLS on four commonly used hyperspectral datasets (hyperspectral images), including the Washington DC Mall HS image dataset (HS Image I), the Botswana HS image dataset (HS Image II), the Pavia Centre HS image dataset (HS Image III) and the Jasper Ridge HS image dataset (HS Image IV). The description about the HS images has been covered in Table 3.Our simulation experiments have been conducted on a hardware environment composed of an Intel(R) Core (TM) i3-4010U CPU with 8 GB RAM with a serial speed of 1.7 GHz. On the other hand, the software environment is composed of Windows 10 as an operating system, and simulation was performed using Matlab simulation software. The above configuration has been used for all the simulation experiments conducted with different HSICAs.

**Table 3. Short description of the HS images used for the simulations**

| HS Image | HS Image Sensor | HS Image Dimension | Pixel Depth | Wavelength | Pixel Resolution |
|---|---|---|---|---|---|
| HS Image I | HYDICE | 1280 x 307 x 191 | 14 | 400 nm to 2400 nm | 3 mt to 4 mt |
| HS Image II | Hyperion | 1476 x 256 x 242 | 16 | 400 nm to 2500 nm | 30 mt |
| HS Image III | ROSIS | 1096 x 1096 x 102 | 13 | 430 nm to 860 nm | 1.3 mt |
| HS Image IV | AVIRIS | 100 x 100 x 224 | 13 | 380 nm to 2500 nm | 4 mt to 20 mt |

For calculation of the coding efficiency, four different performance metrics are used, which are Peak Signal Noise Ratio (PSNR), Structural similarity (SSIM) index, Feature similarity (FSIM) index and Bjøntegaard delta peak signal-to-noise rate (BD-PSNR) [52, 64-66].

The coding memory is calculated as in kilobytes (KB), and coding complexity is calculated as in time required by the HSICA for encoding of the transform coefficients (encoding time) and decoding of the received bit stream from the transmission channel (decoding time). The coding complexity of any HSICA is directly proportional to the speed of the HSICA [61]. The performance of 3D-SLS on different performance metrics is compared with the 3D-SPECK (CA 1) [48], 3-SPIHT (CA 2) [54], 3D-WBTC (CA 3) [58], 3D-LSK (CA 4) [50], 3D-NLS (CA 5) [55], 3D-LMBTC (CA 6) [59], 3D-ZM-SPECK (CA 7) [51] and 3D-M-ZM-SPECK (CA 8) [52].

Let the original HS image be defined as A(α,β,γ) while the reconstructed HS image after the compression process is defined as B(α,β,γ). The 'α', 'β' and 'γ' are defined as the location of the pixel/coefficient of the HS image. The PSNR of any HSICA is defined in Equation (1), while the associated Mean Square Error (MSE) is defined in Equation (2) [67]. The SSIM is defined mathematically as Equation (3) [68-72].

$$PSNR = 20\ \log_{10}\left[\frac{Max\{A(\alpha,\beta,\gamma)\}}{MSE}\right] \tag{1}$$

$$MSE = \frac{1}{N_{pix}}\sum_{x,y,z}[A(\alpha,\beta,\gamma) - B(\alpha,\beta,\gamma)]^2 \tag{2}$$

$$\text{SSIM (A, B)} = \left[\frac{(2\mu_A\mu_B + C_1)(2\sigma_{AB} + C_2)}{(\mu_A^2 + \mu_B^2 + C_1)(\sigma_A^2 + \sigma_B^2 + C_2)}\right] \tag{3}$$

The total number of coefficients of the HS image under test is defined as $N_{pix}$ for Equation (1), while $C_1$ and $C_2$ are the constants in Equation (3).

The mean of the original HS image and reconstructed HS image is represented as $\mu_A$ and $\mu_B$. In the same way, the variance between the original HS image and the reconstructed HS image is represented $\sigma_A^2\ and\ \sigma_B^2$ while covariance is defined as $\sigma_{AB}$.

### *4.1. Coding Efficiency*

From Table 4, it is clear that the proposed HSICA outperforms the other state of art HSICA. It has been clear that the variation of the coding efficiency between the 3D-SLS and 3D-SPIHT [54] lies between 0.22 dB to 0.6 dB for HS image I, 0.1 dB to 0.43 dB for HS image II, 0.26 dB to 0.45 dB for HS image III and 0.24 dB to 0.56 dB for HS image IV. In the same way, the variation of the coding efficiency between 3D-SLS and 3D-NLS [55] lies between 0.41 dB to 1.34 dB for HS image I, 0.18 dB to 0.68 dB for HS image II, 0.33 dB to 0.63 dB for HS image III and 0.15 dB to 0.65 dB for HS image IV. The coding efficiency improvement is due to the increase in the number of significant coefficients. 3D-SLS has a higher number of significant coefficients than the other TC-HSICA. In the same way, 3D-SLS has slightly higher numeric values for the SSIM and FSIM, which are covered in Table 5 and Table 6. The calculation of BD-PSNR savings utilizes PSNR as the selected objective quality indicator. It has been clear from Table 7 that 3D-SLS has better performance than other HSICAs under test.

### *4.2. Coding Memory*

The requirement of the memory by the TC-HSICA for the coding process is known as coding memory. For the list-based HSICA, such as 3D-SPECK [48], 3D-SPIHT [54], and 3D-WBTC [58] had high coding memory requirements, and the demand for coding memory increased exponentially with the bit rate. At the same time, the listless HSICA has a fixed coding memory requirement, which depends only on the dimension of the HS image and not on the bit rate. Thus, at the low bit rates, list-based HSICA performs better than listless HSICA. It has been noticed from Table 8 that 3D-SLS has a lower coding memory demand than 3D-SPIHT [54] because it has only one list for the tracking of the significance/insignificance of the transform coefficients or zerotree. It required almost one-fifth of coding memory compared to the 3D-SPIHT [54].

### *4.3. Coding Complexity*

The proposed HSICA 3D-SLS has higher coding complexity than other zerotree HSICA, such as 3D-SPIHT [54], 3D-NLS [55] and other listless HSICA, as shown in Tables 9 and 10. The higher coding complexity of 3D-SLS is due to the more read and write operation during the execution of the HSICA, which nis more than 3D-SPIHT [54] and 3D-NLS [55]. The listless HSICA has low coding complexity and have fast execution process. It is quite clear that the high complexity of 3D-SLS is due to the use of the list.

**Table 4. Comparison of coding efficiency (PSNR) for 3D-SLS and reported compression algorithms**

| Bit Rate | CA 1 [48] | CA 2 [54] | CA 3 [58] | CA 4 [50] | CA 5 [55] | CA 6 [59] | CA 7 [51] | CA 8 [52] | 3D-SLS |
|---|---|---|---|---|---|---|---|---|---|
| | **HS Image I** | | | | | | | | |
| 0.025 | 34.56 | 34.43 | 34.54 | 34.47 | 34.43 | 34.35 | 34.44 | 34.11 | 34.89 |
| 0.05 | 36.52 | 36.27 | 36.47 | 36.51 | 36.28 | 36.52 | 36.48 | 36.34 | 36.77 |
| 0.1 | 38.53 | 38.28 | 38.5 | 38.35 | 38.12 | 38.29 | 38.33 | 38.04 | 38.62 |
| 0.2 | 41.54 | 41.34 | 41.52 | 41.49 | 41.27 | 41.19 | 41.42 | 41.17 | 41.76 |
| 0.25 | 42.97 | 42.89 | 43.08 | 43.12 | 41.92 | 42.02 | 42.17 | 42.87 | 43.26 |
| 0.5 | 46.81 | 46.6 | 46.81 | 46.76 | 46.41 | 46.09 | 46.73 | 46.21 | 46.82 |
| 0.75 | 50.54 | 50.41 | 50.59 | 50.61 | 50.33 | 50.37 | 50.51 | 50.11 | 51.01 |
| 1 | 53.52 | 53.32 | 53.51 | 53.49 | 53.33 | 53.46 | 53.47 | 53.12 | 53.84 |
| 2 | 66.09 | 66.02 | 66.19 | 65.97 | 65.91 | 65.84 | 66.01 | 65.22 | 66.41 |
| | **HS Image II** | | | | | | | | |
| 0.025 | 27.53 | 27.41 | 27.52 | 27.32 | 27.37 | 27.31 | 27.32 | 26.98 | 27.84 |
| 0.05 | 30.22 | 29.99 | 30.21 | 30.2 | 29.97 | 30.19 | 30.2 | 29.94 | 30.35 |
| 0.1 | 32.57 | 32.48 | 32.57 | 32.48 | 32.18 | 32.04 | 32.07 | 31.89 | 32.74 |
| 0.2 | 34.78 | 34.63 | 34.75 | 34.66 | 34.51 | 34.64 | 34.66 | 34.21 | 34.96 |
| 0.25 | 35.62 | 35.5 | 35.62 | 35.63 | 35.5 | 35.55 | 35.56 | 35.25 | 35.87 |
| 0.5 | 39.09 | 39.03 | 39.18 | 38.76 | 38.63 | 38.46 | 38.47 | 38.02 | 39.31 |
| 0.75 | 42.09 | 41.95 | 42.08 | 41.48 | 41.92 | 41.28 | 41.3 | 40.97 | 42.28 |
| 1 | 44.19 | 44.05 | 44.18 | 44.2 | 44.05 | 43.89 | 43.91 | 43.55 | 44.39 |
| 2 | 51.48 | 51.33 | 51.47 | 51.22 | 51.25 | 50.85 | 50.86 | 50.66 | 51.43 |
| | **HS Image III** | | | | | | | | |
| 0.025 | 28.83 | 28.68 | 28.82 | 28.68 | 28.62 | 28.64 | 28.67 | 28.44 | 29.08 |
| 0.05 | 30.32 | 30.06 | 30.3 | 30.29 | 30.06 | 30.28 | 30.29 | 30.01 | 30.45 |
| 0.1 | 32.43 | 32.17 | 32.38 | 32.23 | 32.01 | 32.25 | 32.29 | 31.98 | 32.51 |
| 0.2 | 34.84 | 34.62 | 34.83 | 34.61 | 34.51 | 34.56 | 34.58 | 34.38 | 35.01 |
| 0.25 | 35.7 | 35.49 | 35.69 | 35.49 | 35.31 | 35.42 | 35.44 | 35.08 | 35.92 |
| 0.5 | 39.06 | 39.03 | 39.01 | 38.97 | 38.85 | 38.77 | 38.78 | 38.64 | 39.48 |
| 0.75 | 42.24 | 42.04 | 42.24 | 42.01 | 41.97 | 41.83 | 41.84 | 41.69 | 42.49 |
| 1 | 45.16 | 44.93 | 45.16 | 45.14 | 44.82 | 44.5 | 44.5 | 44.34 | 45.38 |
| 2 | 55.46 | 55.31 | 55.46 | 55.34 | 55.24 | 54.89 | 54.91 | 54.31 | 55.57 |
| | **HS Image IV** | | | | | | | | |
| 0.025 | 29.84 | 29.68 | 29.82 | 29.73 | 29.58 | 29.71 | 29.74 | 29.51 | 30.08 |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0.05 | 32.27 | 31.97 | 32.25 | 32.26 | 31.88 | 32.3 | 32.22 | 31.97 | 32.41 |
| 0.1 | 35.08 | 35.11 | 35.07 | 35.29 | 35.04 | 35.06 | 35.03 | 34.84 | 35.48 |
| 0.2 | 39.35 | 39.13 | 39.6 | 39.4 | 39.01 | 39.11 | 39.41 | 39.02 | 39.55 |
| 0.25 | 41.11 | 41.24 | 42.01 | 41.74 | 41.21 | 41.45 | 41.28 | 40.89 | 41.68 |
| 0.5 | 45.98 | 46.33 | 46.78 | 46.26 | 46.24 | 45.91 | 46.14 | 45.87 | 46.89 |
| 0.75 | 50.94 | 51.01 | 51.34 | 51.06 | 51.31 | 51.41 | 51.39 | 50.76 | 51.46 |
| 1 | 54.77 | 54.72 | 55.13 | 54.86 | 54.62 | 54.78 | 54.92 | 54.55 | 55.11 |
| 2 | 61.02 | 60.85 | 61.04 | 60.96 | 60.84 | 60.9 | 60.89 | 60.77 | 61.09 |

**Table 5. Comparison of coding efficiency (SSIM) for 3D-SLS and reported compression algorithms**

| Bit Rate | CA 1 [48] | CA 2 [54] | CA 3 [58] | CA 4 [50] | CA 5 [55] | CA 6 [59] | CA 7 [51] | CA 8 [52] | 3D-SLS |
|---|---|---|---|---|---|---|---|---|---|
| | **HS Image I** | | | | | | | | |
| 0.025 | 0.385 | 0.386 | 0.384 | 0.385 | 0.386 | 0.386 | 0.384 | 0.377 | 0.39 |
| 0.05 | 0.48 | 0.479 | 0.48 | 0.48 | 0.479 | 0.48 | 0.48 | 0.471 | 0.475 |
| 0.1 | 0.587 | 0.587 | 0.585 | 0.584 | 0.587 | 0.589 | 0.59 | 0.581 | 0.589 |
| 0.2 | 0.677 | 0.675 | 0.678 | 0.677 | 0.678 | 0.677 | 0.677 | 0.664 | 0.681 |
| 0.25 | 0.701 | 0.702 | 0.702 | 0.699 | 0.703 | 0.702 | 0.702 | 0.697 | 0.709 |
| 0.5 | 0.79 | 0.788 | 0.787 | 0.789 | 0.789 | 0.788 | 0.787 | 0.774 | 0.798 |
| 0.75 | 0.847 | 0.846 | 0.849 | 0.846 | 0.846 | 0.846 | 0.846 | 0.845 | 0.85 |
| 1 | 0.886 | 0.886 | 0.886 | 0.888 | 0.887 | 0.886 | 0.886 | 0.884 | 0.89 |
| 2 | 0.914 | 0.912 | 0.915 | 0.914 | 0.912 | 0.915 | 0.915 | 0.911 | 0.918 |
| | **HS Image II** | | | | | | | | |
| 0.025 | 0.477 | 0.476 | 0.479 | 0.481 | 0.472 | 0.473 | 0.475 | 0.478 | 0.477 |
| 0.05 | 0.536 | 0.535 | 0.537 | 0.538 | 0.532 | 0.536 | 0.537 | 0.53 | 0.536 |
| 0.1 | 0.625 | 0.626 | 0.623 | 0.624 | 0.621 | 0.625 | 0.626 | 0.627 | 0.625 |
| 0.2 | 0.696 | 0.692 | 0.696 | 0.698 | 0.693 | 0.697 | 0.697 | 0.695 | 0.696 |
| 0.25 | 0.711 | 0.709 | 0.711 | 0.712 | 0.708 | 0.711 | 0.712 | 0.71 | 0.711 |
| 0.5 | 0.764 | 0.764 | 0.762 | 0.767 | 0.764 | 0.764 | 0.764 | 0.765 | 0.764 |
| 0.75 | 0.791 | 0.789 | 0.794 | 0.795 | 0.793 | 0.794 | 0.794 | 0.792 | 0.791 |
| 1 | 0.807 | 0.808 | 0.809 | 0.81 | 0.809 | 0.809 | 0.809 | 0.81 | 0.807 |
| 2 | 0.831 | 0.829 | 0.828 | 0.831 | 0.831 | 0.832 | 0.832 | 0.832 | 0.831 |
| | **HS Image III** | | | | | | | | |
| 0.025 | 0.421 | 0.419 | 0.421 | 0.418 | 0.423 | 0.412 | 0.413 | 0.411 | 0.421 |
| 0.05 | 0.566 | 0.568 | 0.567 | 0.568 | 0.563 | 0.565 | 0.565 | 0.566 | 0.566 |
| 0.1 | 0.654 | 0.652 | 0.655 | 0.657 | 0.653 | 0.655 | 0.655 | 0.654 | 0.654 |
| 0.2 | 0.758 | 0.754 | 0.755 | 0.755 | 0.751 | 0.756 | 0.757 | 0.756 | 0.758 |
| 0.25 | 0.771 | 0.772 | 0.771 | 0.77 | 0.766 | 0.772 | 0.772 | 0.771 | 0.771 |
| 0.5 | 0.86 | 0.859 | 0.861 | 0.859 | 0.858 | 0.86 | 0.86 | 0.861 | 0.86 |
| 0.75 | 0.909 | 0.911 | 0.911 | 0.912 | 0.903 | 0.91 | 0.91 | 0.911 | 0.909 |
| 1 | 0.929 | 0.931 | 0.929 | 0.931 | 0.93 | 0.928 | 0.928 | 0.931 | 0.929 |
| 2 | 0.979 | 0.982 | 0.982 | 0.981 | 0.978 | 0.977 | 0.977 | 0.972 | 0.979 |
| | **HS Image IV** | | | | | | | | |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0.025 | 0.297 | 0.288 | 0.299 | 0.299 | 0.285 | 0.3 | 0.301 | 0.299 | 0.302 |
| 0.05 | 0.346 | 0.338 | 0.345 | 0.346 | 0.341 | 0.349 | 0.349 | 0.248 | 0.339 |
| 0.1 | 0.437 | 0.431 | 0.437 | 0.436 | 0.43 | 0.437 | 0.437 | 0.437 | 0.432 |
| 0.2 | 0.518 | 0.513 | 0.518 | 0.518 | 0.514 | 0.518 | 0.518 | 0.517 | 0.511 |
| 0.25 | 0.545 | 0.543 | 0.545 | 0.552 | 0.545 | 0.553 | 0.553 | 0.55 | 0.547 |
| 0.5 | 0.603 | 0.601 | 0.603 | 0.608 | 0.606 | 0.611 | 0.611 | 0.604 | 0.601 |
| 0.75 | 0.63 | 0.629 | 0.63 | 0.636 | 0.631 | 0.636 | 0.636 | 0.633 | 0.632 |
| 1 | 0.645 | 0.645 | 0.645 | 0.648 | 0.646 | 0.647 | 0.647 | 0.644 | 0.649 |
| 2 | 0.666 | 0.666 | 0.666 | 0.666 | 0.666 | 0.668 | 0.668 | 0.663 | 0.666 |

**Table 6. Comparison of coding efficiency (FSIM) for 3D-SLS and reported compression algorithms**

| Bit Rate | CA 1 [48] | CA 2 [54] | CA 3 [58] | CA 4 [50] | CA 5 [55] | CA 6 [59] | CA 7 [51] | CA 8 [52] | 3D-SLS |
|---|---|---|---|---|---|---|---|---|---|
| | **HS Image I** | | | | | | | | |
| 0.025 | 0.641 | 0.642 | 0.712 | 0.639 | 0.643 | 0.637 | 0.638 | 0.633 | 0.648 |
| 0.05 | 0.71 | 0.695 | 0.787 | 0.712 | 0.712 | 0.716 | 0.716 | 0.711 | 0.709 |
| 0.1 | 0.774 | 0.759 | 0.83 | 0.78 | 0.779 | 0.784 | 0.784 | 0.777 | 0.789 |
| 0.2 | 0.784 | 0.776 | 0.843 | 0.783 | 0.79 | 0.795 | 0.795 | 0.791 | 0.81 |
| 0.25 | 0.851 | 0.844 | 0.909 | 0.855 | 0.856 | 0.863 | 0.863 | 0.849 | 0.864 |
| 0.5 | 0.891 | 0.888 | 0.947 | 0.91 | 0.889 | 0.903 | 0.903 | 0.896 | 0.901 |
| 0.75 | 0.919 | 0.917 | 0.97 | 0.92 | 0.918 | 0.918 | 0.918 | 0.912 | 0.931 |
| 1 | 0.978 | 0.978 | 0.996 | 0.98 | 0.98 | 0.981 | 0.981 | 0.978 | 0.988 |
| 2 | 66.09 | 66.02 | 66.19 | 65.97 | 65.91 | 65.84 | 66.01 | 65.22 | 66.41 |
| | **HS Image II** | | | | | | | | |
| 0.025 | 0.588 | 0.59 | 0.586 | 0.591 | 0.588 | 0.579 | 0.579 | 0.58 | 0.588 |
| 0.05 | 0.67 | 0.672 | 0.674 | 0.676 | 0.675 | 0.674 | 0.673 | 0.671 | 0.67 |
| 0.1 | 0.712 | 0.714 | 0.711 | 0.717 | 0.713 | 0.697 | 0.698 | 0.699 | 0.712 |
| 0.2 | 0.761 | 0.774 | 0.76 | 0.779 | 0.747 | 0.775 | 0.775 | 0.776 | 0.761 |
| 0.25 | 0.807 | 0.806 | 0.805 | 0.808 | 0.793 | 0.802 | 0.805 | 0.809 | 0.807 |
| 0.5 | 0.92 | 0.922 | 0.921 | 0.923 | 0.914 | 0.91 | 0.91 | 0.911 | 0.92 |
| 0.75 | 0.961 | 0.963 | 0.962 | 0.963 | 0.962 | 0.959 | 0.959 | 0.961 | 0.961 |
| 1 | 0.975 | 0.971 | 0.973 | 0.976 | 0.978 | 0.977 | 0.977 | 0.974 | 0.975 |
| 2 | 0.997 | 0.997 | 0.996 | 0.997 | 0.998 | 0.998 | 0.998 | 0.998 | 0.997 |
| | **HS Image III** | | | | | | | | |
| 0.025 | 0.71 | 0.711 | 0.702 | 0.71 | 0.666 | 0.699 | 0.701 | 0.7 | 0.71 |
| 0.05 | 0.713 | 0.715 | 0.715 | 0.714 | 0.715 | 0.713 | 0.714 | 0.715 | 0.713 |
| 0.1 | 0.787 | 0.781 | 0.786 | 0.788 | 0.788 | 0.785 | 0.785 | 0.786 | 0.787 |
| 0.2 | 0.832 | 0.83 | 0.831 | 0.836 | 0.834 | 0.835 | 0.835 | 0.836 | 0.832 |
| 0.25 | 0.851 | 0.849 | 0.851 | 0.852 | 0.837 | 0.838 | 0.838 | 0.84 | 0.851 |
| 0.5 | 0.917 | 0.915 | 0.911 | 0.915 | 0.915 | 0.914 | 0.914 | 0.912 | 0.917 |
| 0.75 | 0.955 | 0.956 | 0.954 | 0.957 | 0.947 | 0.95 | 0.95 | 0.951 | 0.955 |
| 1 | 0.97 | 0.968 | 0.964 | 0.97 | 0.969 | 0.97 | 0.97 | 0.972 | 0.97 |
| 2 | 0.994 | 0.992 | 0.993 | 0.995 | 0.996 | 0.996 | 0.996 | 0.998 | 0.994 |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | **HS Image IV** | | | | | | | | |
| 0.025 | 0.286 | 0.286 | 0.282 | 0.29 | 0.288 | 0.295 | 0.303 | 0.288 | 0.29 |
| 0.05 | 0.293 | 0.288 | 0.289 | 0.294 | 0.286 | 0.302 | 0.302 | 0.294 | 0.294 |
| 0.1 | 0.321 | 0.319 | 0.32 | 0.321 | 0.32 | 0.338 | 0.338 | 0.337 | 0.329 |
| 0.2 | 0.443 | 0.434 | 0.443 | 0.452 | 0.439 | 0.488 | 0.488 | 0.482 | 0.441 |
| 0.25 | 0.518 | 0.522 | 0.518 | 0.53 | 0.529 | 0.537 | 0.537 | 0.531 | 0.526 |
| 0.5 | 0.694 | 0.692 | 0.694 | 0.699 | 0.696 | 0.75 | 0.75 | 0.745 | 0.698 |
| 0.75 | 0.807 | 0.812 | 0.808 | 0.817 | 0.813 | 0.835 | 0.834 | 0.827 | 0.828 |
| 1 | 0.871 | 0.867 | 0.872 | 0.869 | 0.867 | 0.873 | 0.873 | 0.866 | 0.889 |
| 2 | 0.979 | 0.978 | 0.979 | 0.979 | 0.978 | 0.979 | 0.979 | 0.965 | 0.985 |

**Table 7. BD-PSNR calculation for different HSICA with reference to 3D-SLS**

| Image | CA 1 [48] | CA 2 [54] | CA 3 [58] | CA 4 [50] | CA 5 [55] | CA 6 [59] | CA 7 [51] | CA 8 [52] |
|---|---|---|---|---|---|---|---|---|
| HS Image I | 0.2404 | 0.4171 | 0.2296 | 0.2753 | 0.6108 | 0.5815 | 0.4283 | 0.6486 |
| HS Image II | 0.1783 | 0.3129 | 0.1779 | 0.3377 | 0.4315 | 0.5261 | 0.51 | 0.8263 |
| HS Image III | 0.1961 | 0.3940 | 0.2151 | 0.3435 | 0.5016 | 0.4904 | 0.4716 | 0.7426 |
| HS Image IV | 0.3761 | 0.4154 | 0.08 | 0.2240 | 0.4583 | 0.3488 | 0.3098 | 0.6218 |

**Table 8. Comparison of coding memory for 3D-SLS and reported compression algorithms**

| Bit Rate | CA 1 [48] | CA 2 [54] | CA 3 [58] | CA 4 [50] | CA 5 [55] | CA 6 [59] | CA 7 [51] | CA 8 [52] | 3D-SLS |
|---|---|---|---|---|---|---|---|---|---|
| | **HS Image I** | | | | | | | | |
| 0.025 | 54.87 | 54.42 | 55.21 | 512 | 1024 | 12 | 0 | 32 | 8.95 |
| 0.05 | 136.1 | 145.3 | 137.9 | 512 | 1024 | 12 | 0 | 32 | 23.78 |
| 0.1 | 243.8 | 263.3 | 250.1 | 512 | 1024 | 12 | 0 | 32 | 41.08 |
| 0.2 | 416.3 | 438 | 416 | 512 | 1024 | 12 | 0 | 32 | 68.33 |
| 0.25 | 586.67 | 605.78 | 630.49 | 512 | 1024 | 12 | 0 | 32 | 96.92 |
| 0.5 | 1048.8 | 1060.5 | 1049 | 512 | 1024 | 12 | 0 | 32 | 167.54 |
| 0.75 | 1287.31 | 1333.19 | 1329.77 | 512 | 1024 | 12 | 0 | 32 | 296.26 |
| 1 | 1802.5 | 1826.7 | 1724.6 | 512 | 1024 | 12 | 0 | 32 | 411.42 |
| 2 | 2865.17 | 2897 | 3052.09 | 512 | 1024 | 12 | 0 | 32 | 652.48 |
| | **HS Image II** | | | | | | | | |
| 0.025 | 60.22 | 63.41 | 60.69 | 512 | 1024 | 12 | 0 | 32 | 10.54 |
| 0.05 | 114.86 | 120.09 | 115.57 | 512 | 1024 | 12 | 0 | 32 | 20.39 |
| 0.1 | 245.34 | 247.27 | 246.04 | 512 | 1024 | 12 | 0 | 32 | 46.74 |
| 0.2 | 463.49 | 495.61 | 473.86 | 512 | 1024 | 12 | 0 | 32 | 105.7 |
| 0.25 | 628.84 | 651.89 | 630.49 | 512 | 1024 | 12 | 0 | 32 | 123.4 |
| 0.5 | 1148.6 | 1164.8 | 1149.5 | 512 | 1024 | 12 | 0 | 32 | 267.8 |
| 0.75 | 1305.8 | 1299.3 | 1306 | 512 | 1024 | 12 | 0 | 32 | 288.7 |
| 1 | 2056 | 2090.4 | 2057.4 | 512 | 1024 | 12 | 0 | 32 | 470.8 |
| 2 | 3051 | 3081.8 | 3052.1 | 512 | 1024 | 12 | 0 | 32 | 708.3 |
| | **HS Image III** | | | | | | | | |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0.025 | 54.84 | 61.36 | 55.03 | 512 | 1024 | 12 | 0 | 32 | 10.09 |
| 0.05 | 136.4 | 150.95 | 138.5 | 512 | 1024 | 12 | 0 | 32 | 24.83 |
| 0.1 | 246.14 | 260.59 | 251.71 | 512 | 1024 | 12 | 0 | 32 | 41.3 |
| 0.2 | 418.18 | 462.06 | 418.28 | 512 | 1024 | 12 | 0 | 32 | 79.07 |
| 0.25 | 602.43 | 630.18 | 610.53 | 512 | 1024 | 12 | 0 | 32 | 117.9 |
| 0.5 | 1042.3 | 1086.6 | 1041.9 | 512 | 1024 | 12 | 0 | 32 | 196.2 |
| 0.75 | 1453.5 | 1488 | 1453.2 | 512 | 1024 | 12 | 0 | 32 | 335.2 |
| 1 | 1937.9 | 1919.7 | 1936.8 | 512 | 1024 | 12 | 0 | 32 | 426.6 |
| 2 | 2713.4 | 2665.7 | 2714.3 | 512 | 1024 | 12 | 0 | 32 | 633.1 |
| | **HS Image IV** | | | | | | | | |
| 0.025 | 55.23 | 55.52 | 55.61 | 512 | 1024 | 12 | 0 | 32 | 8.8 |
| 0.05 | 143.02 | 145.9 | 143.3 | 512 | 1024 | 12 | 0 | 32 | 26.11 |
| 0.1 | 241.4 | 245.9 | 245.8 | 512 | 1024 | 12 | 0 | 32 | 46.31 |
| 0.2 | 440 | 445.7 | 443.7 | 512 | 1024 | 12 | 0 | 32 | 87.22 |
| 0.25 | 480.11 | 462.3 | 489.73 | 512 | 1024 | 12 | 0 | 32 | 90.29 |
| 0.5 | 821.6 | 808.9 | 827.9 | 512 | 1024 | 12 | 0 | 32 | 168.5 |
| 0.75 | 1150.17 | 1152.97 | 1155.23 | 512 | 1024 | 12 | 0 | 32 | 271.3 |
| 1 | 1492.71 | 1503.8 | 1532.6 | 512 | 1024 | 12 | 0 | 32 | 346.5 |
| 2 | 2592.52 | 2626.3 | 2503.1 | 512 | 1024 | 12 | 0 | 32 | 643.7 |

**Table 9. Comparison of computational complexity (encoding time) for 3D-SLS and reported compression algorithms**

| Bit Rate | CA 1 [48] | CA 2 [54] | CA 3 [58] | CA 4 [50] | CA 5 [55] | CA 6 [59] | CA 7 [51] | CA 8 [52] | 3D-SLS |
|---|---|---|---|---|---|---|---|---|---|
| | **HS Image I** | | | | | | | | |
| 0.025 | 3.09 | 1.44 | 1.71 | 0.43 | 0.51 | 1.33 | 0.83 | 0.49 | 1.59 |
| 0.05 | 6.23 | 2.71 | 2.78 | 0.55 | 0.65 | 1.9 | 1.11 | 0.61 | 2.91 |
| 0.1 | 25.1 | 7.5 | 6.5 | 0.8 | 0.91 | 3.9 | 1.78 | 0.88 | 8.11 |
| 0.2 | 57.9 | 25.8 | 24.8 | 1.1 | 1.21 | 5.1 | 2.81 | 1.17 | 28.09 |
| 0.25 | 104.58 | 31.19 | 29.92 | 1.71 | 1.79 | 10.61 | 5.05 | 1.77 | 33.24 |
| 0.5 | 414.8 | 140.1 | 211.2 | 2.5 | 2.64 | 11.3 | 7.41 | 2.61 | 149.02 |
| 0.75 | 950.77 | 370.29 | 713.02 | 3.57 | 3.88 | 17.22 | 10.47 | 3.71 | 381.35 |
| 1 | 1497.5 | 575 | 804 | 4.41 | 4.57 | 21.12 | 13.21 | 4.49 | 594.2 |
| 2 | 3822.03 | 1426.61 | 4409.82 | 11.55 | 14.58 | 40.09 | 23.39 | 12.91 | 1484.21 |
| | **HS Image II** | | | | | | | | |
| 0.025 | 5.19 | 1.08 | 2.05 | 0.51 | 1.99 | 2.33 | 1.55 | 0.91 | 1.24 |
| 0.05 | 8.01 | 2.19 | 3.68 | 0.94 | 9.43 | 3.08 | 2.08 | 1.74 | 2.59 |
| 0.1 | 42.11 | 5.47 | 7.26 | 2.01 | 12.38 | 4.79 | 3.49 | 3.12 | 7.02 |
| 0.2 | 68.31 | 17.14 | 20.14 | 4.27 | 14.01 | 8.64 | 5.26 | 4.98 | 19.22 |
| 0.25 | 91.77 | 23.38 | 67.74 | 7.24 | 17.56 | 10.44 | 9.09 | 8.17 | 28.95 |
| 0.5 | 402.48 | 100.15 | 179.99 | 9.95 | 18.22 | 19.91 | 11.76 | 10.92 | 109.71 |
| 0.75 | 1066.62 | 581.78 | 682.4 | 12.17 | 20.19 | 29.03 | 17.1 | 16.68 | 602.37 |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1320.53 | 641.34 | 875.01 | 14.96 | 22.96 | 35.17 | 19.04 | 18.07 | 688.21 |
| 2 | | | | | | | | | |
| | **HS Image III** | | | | | | | | |
| 0.025 | 5.34 | 1.96 | 2.15 | 0.57 | 3.34 | 2.42 | 1.1 | 1.03 | 2.08 |
| 0.05 | 10.17 | 8.34 | 3.31 | 2.03 | 7.72 | 4.62 | 3.69 | 3.12 | 8.91 |
| 0.1 | 24.02 | 17.65 | 6.54 | 3.65 | 10.32 | 7.78 | 4.34 | 4.03 | 19.29 |
| 0.2 | 69.81 | 22.2 | 14.58 | 5.15 | 11.26 | 9.16 | 6.02 | 5.87 | 26.35 |
| 0.25 | 91.62 | 38.56 | 37.3 | 7.7 | 14.92 | 12.26 | 8.16 | 7.91 | 40.6 |
| 0.5 | 371.82 | 187.18 | 198.26 | 9.14 | 17.71 | 17.57 | 11.38 | 10.57 | 198.21 |
| 0.75 | 955.11 | 440.87 | 597.21 | 11.01 | 21.48 | 24.43 | 14.32 | 12.86 | 478.77 |
| 1 | 1553.24 | 715.1 | 1011.4 | 13.69 | 24.92 | 27.96 | 18.64 | 16.28 | 731.02 |
| 2 | 4770.03 | 2338.89 | 3882.91 | 27.52 | 39.89 | 55.15 | 36.57 | 32.14 | 2501.16 |
| | **HS Image IV** | | | | | | | | |
| 0.025 | 3.01 | 1.34 | 1.66 | 0.49 | 0.53 | 1.38 | 0.86 | 0.51 | 1.51 |
| 0.05 | 6.01 | 2.55 | 2.36 | 0.56 | 0.67 | 2.06 | 1.14 | 0.63 | 2.94 |
| 0.1 | 21.1 | 7.6 | 6.4 | 0.9 | 1.09 | 3 | 1.77 | 1.03 | 9.03 |
| 0.2 | 54.2 | 20.6 | 17.7 | 1.2 | 1.34 | 5.2 | 2.84 | 1.27 | 22.31 |
| 0.25 | 97.1 | 37.92 | 21.35 | 1.84 | 1.99 | 8.72 | 4.73 | 1.91 | 54.29 |
| 0.5 | 315.3 | 101.6 | 182.4 | 2.6 | 2.74 | 10.9 | 6.24 | 2.72 | 121.38 |
| 0.75 | 705.45 | 267.31 | 530.54 | 3.3 | 3.77 | 17.79 | 10.63 | 3.51 | 299.74 |
| 1 | 757.3 | 425.4 | 942.8 | 5.1 | 5.34 | 22.7 | 15.84 | 5.24 | 438.21 |
| 2 | 3255.3 | 1213.1 | 2380.4 | 9.55 | 14.2 | 59.26 | 56.42 | 13.67 | 1249.8 |

**Table 10. Comparison of computational complexity (decoding time) for 3D-SLS and reported compression algorithms**

| Bit Rate | CA 1 [48] | CA 2 [54] | CA 3 [58] | CA 4 [50] | CA 5 [55] | CA 6 [59] | CA 7 [51] | CA 8 [52] | 3D-SLS |
|---|---|---|---|---|---|---|---|---|---|
| | **HS Image I** | | | | | | | | |
| 0.025 | 3.09 | 1.44 | 1.71 | 0.43 | 0.51 | 1.33 | 0.83 | 0.49 | 1.59 |
| 0.05 | 6.23 | 2.71 | 2.78 | 0.55 | 0.65 | 1.9 | 1.11 | 0.61 | 2.91 |
| 0.1 | 25.1 | 7.5 | 6.5 | 0.8 | 0.91 | 3.9 | 1.78 | 0.88 | 8.11 |
| 0.2 | 57.9 | 25.8 | 24.8 | 1.1 | 1.21 | 5.1 | 2.81 | 1.17 | 28.09 |
| 0.25 | 104.58 | 31.19 | 29.92 | 1.71 | 1.79 | 10.61 | 5.05 | 1.77 | 33.24 |
| 0.5 | 414.8 | 140.1 | 211.2 | 2.5 | 2.64 | 11.3 | 7.41 | 2.61 | 149.02 |
| 0.75 | 950.77 | 370.29 | 713.02 | 3.57 | 3.88 | 17.22 | 10.47 | 3.71 | 381.35 |
| 1 | 1497.5 | 575 | 804 | 4.41 | 4.57 | 21.12 | 13.21 | 4.49 | 594.2 |
| 2 | 3822.03 | 1426.61 | 4409.82 | 11.55 | 14.58 | 40.09 | 23.39 | 12.91 | 1484.21 |
| | **HS Image II** | | | | | | | | |
| 0.025 | 5.19 | 1.08 | 2.05 | 0.51 | 1.99 | 2.33 | 1.55 | 0.91 | 1.24 |
| 0.05 | 8.01 | 2.19 | 3.68 | 0.94 | 9.43 | 3.08 | 2.08 | 1.74 | 2.59 |
| 0.1 | 42.11 | 5.47 | 7.26 | 2.01 | 12.38 | 4.79 | 3.49 | 3.12 | 7.02 |
| 0.2 | 68.31 | 17.14 | 20.14 | 4.27 | 14.01 | 8.64 | 5.26 | 4.98 | 19.22 |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0.25 | 91.77 | 23.38 | 67.74 | 7.24 | 17.56 | 10.44 | 9.09 | 8.17 | 28.95 |
| 0.5 | 402.48 | 100.15 | 179.99 | 9.95 | 18.22 | 19.91 | 11.76 | 10.92 | 109.71 |
| 0.75 | 1066.62 | 581.78 | 682.4 | 12.17 | 20.19 | 29.03 | 17.1 | 16.68 | 602.37 |
| 1 | 1320.53 | 641.34 | 875.01 | 14.96 | 22.96 | 35.17 | 19.04 | 18.07 | 688.21 |
| 2 | | | | | | | | | |
| | **HS Image III** | | | | | | | | |
| 0.025 | 5.34 | 1.96 | 2.15 | 0.57 | 3.34 | 2.42 | 1.1 | 1.03 | 2.08 |
| 0.05 | 10.17 | 8.34 | 3.31 | 2.03 | 7.72 | 4.62 | 3.69 | 3.12 | 8.91 |
| 0.1 | 24.02 | 17.65 | 6.54 | 3.65 | 10.32 | 7.78 | 4.34 | 4.03 | 19.29 |
| 0.2 | 69.81 | 22.2 | 14.58 | 5.15 | 11.26 | 9.16 | 6.02 | 5.87 | 26.35 |
| 0.25 | 91.62 | 38.56 | 37.3 | 7.7 | 14.92 | 12.26 | 8.16 | 7.91 | 40.6 |
| 0.5 | 371.82 | 187.18 | 198.26 | 9.14 | 17.71 | 17.57 | 11.38 | 10.57 | 198.21 |
| 0.75 | 955.11 | 440.87 | 597.21 | 11.01 | 21.48 | 24.43 | 14.32 | 12.86 | 478.77 |
| 1 | 1553.24 | 715.1 | 1011.4 | 13.69 | 24.92 | 27.96 | 18.64 | 16.28 | 731.02 |
| 2 | 4770.03 | 2338.89 | 3882.91 | 27.52 | 39.89 | 55.15 | 36.57 | 32.14 | 2501.16 |
| | **HS Image IV** | | | | | | | | |
| 0.025 | 3.01 | 1.34 | 1.66 | 0.49 | 0.53 | 1.38 | 0.86 | 0.51 | 1.51 |
| 0.05 | 6.01 | 2.55 | 2.36 | 0.56 | 0.67 | 2.06 | 1.14 | 0.63 | 2.94 |
| 0.1 | 21.1 | 7.6 | 6.4 | 0.9 | 1.09 | 3 | 1.77 | 1.03 | 9.03 |
| 0.2 | 54.2 | 20.6 | 17.7 | 1.2 | 1.34 | 5.2 | 2.84 | 1.27 | 22.31 |
| 0.25 | 97.1 | 37.92 | 21.35 | 1.84 | 1.99 | 8.72 | 4.73 | 1.91 | 54.29 |
| 0.5 | 315.3 | 101.6 | 182.4 | 2.6 | 2.74 | 10.9 | 6.24 | 2.72 | 121.38 |
| 0.75 | 705.45 | 267.31 | 530.54 | 3.3 | 3.77 | 17.79 | 10.63 | 3.51 | 299.74 |
| 1 | 757.3 | 425.4 | 942.8 | 5.1 | 5.34 | 22.7 | 15.84 | 5.24 | 438.21 |
| 2 | 3255.3 | 1213.1 | 2380.4 | 9.55 | 14.2 | 59.26 | 56.42 | 13.67 | 1249.8 |

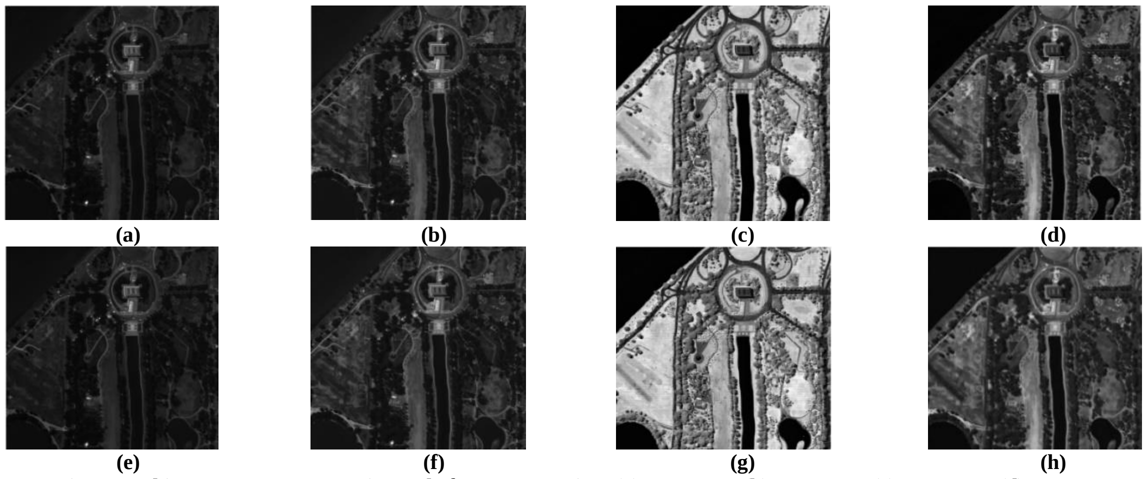

**Fig. 2 Washington DC MALL HS image before compression, (a) Frame 29, (b) Frame 59, (c) Frame 89, (d) Frame 149 Washington DC MALL HS image after compression with CR 16, (e) Frame 29, (f) Frame 59, (g) Frame 89, and (h) Frame 149.**

Due to the multiple access (read or write operation) of the HSICA. It has been clear that a lot of computations (logical, algebraic and arithmetic) during the coding process make the compression algorithm slow. In order to investigate the impact of the compression on the compressed HS image, we present Figure 2, which provides both quantitative and qualitative evidence for 4 frequency frames of the WDC Mall HS image during the compression process.

## 5. Comparison with state-of-the-art HSICA

Table 11 throws a short comparative analysis between the different state-of-the-art HSICAs with the proposed compression algorithm 3D-SLS. From the result, it is clear that 3D-SLS perform better than the state-of-the-art link list compression algorithm.

**Table 11. Comparison between the different compression algorithms**

| Reference | HSICA | Major Outcome |
|---|---|---|
| [54] | 3D-SPIHT | Uses the linked list and has improved coding efficiency compared to other transform coding |
| [48] | 3D-SPECK | Slightly higher performance than 3D-SPIHT but still has high complexity at a high bit rate. |
| [58] | 3D-WBTC | Perform better at low bit rates, but complexity increases rapidly with an increase in the bit rate. |
| [56] | 3D-BPEC | Moderate coding efficiency with the high coding memory requirement due to the use of six fixed-length array |
| [72] | 3D-LMZC | Coding efficiency is higher due to the use of curvelet transform |
| Proposed Research | 3D-SLS | High coding efficiency with low coding memory and coding efficiency |

## 6. Conclusion

In this manuscript, we present a low-memory variant of the 3D-SPIHT [54] method that we created. The 3D-SLS uses a single list to maintain track of the significance of the coefficients or sets. The theoretical analysis and experimental findings both revealed very clearly that the suggested 3D-SLS has a superior PSNR than the 3D-SPIHT and 3D-NLS, and it has almost the same coding time as the 3D-SPIHT.

This was demonstrated by the fact that the two sets of results were in complete agreement. This indicates that we could improve the performance of 3D-SPIHT without compromising its ease of use or incurring any additional overhead costs while simultaneously decreasing the amount of coding memory that was required by almost six times. Because of its low memory requirements and simplified management, the 3D-SLS is ideally suited for memory-limited portable HS image sensors.

## Acknowledgement

We are sincerely thankful to the anonymous reviewers for their critical comments and suggestions to improve the quality of this chapter. The authors want to express their gratitude to Integral University, Lucknow, Uttar Pradesh, India, for providing manuscript number IU/R&D/2025-MCN0003432 for the present research work.

## Authors' Contributions

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

## References

[1] Rajat Kumar Arya, Subhojit Paul, and Rajeev Srivastava, "An Efficient Hyperspectral Image Classification Method Using Retentive Network," *Advances in Space Research*, vol. 75, no. 2, pp. 1701-1718, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[2] Samiran Das, and Sandip Ghosal, "Unmixing Aware Compression of Hyperspectral Image by Rank Aware Orthogonal Parallel Factorization Decomposition," *Journal of Applied Remote Sensing*, vol. 17, no. 4, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[3] Shrish Bajpai, Harsh Vikram Singh, and Naimur Rahman Kidwai, "Feature Extraction & Classification of Hyperspectral Images Using Singular Spectrum Analysis & Multinomial Logistic Regression Classifiers," *2017 International Conference on Multimedia, Signal Processing and Communication Technologies (IMPACT)*, Aligarh, India, pp. 97-100, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[4] Rabi N. Sahoo et al., "Optimizing the Retrieval of Wheat Crop Traits from UAV-Borne Hyperspectral Image with Radiative Transfer Modelling Using Gaussian Process Regression," *Remote Sensing*, vol. 15, no. 23, pp. 1-18, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[5] Jost Stergar, Rok Hren, and Matija Milanic, "Design and Validation of a Custom-Made Hyperspectral Microscope Imaging System for Biomedical Applications," *Sensors*, vol. 23, no. 5, pp. 1-18, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[6] Chaitanya B. Pande, and Kanak N. Moharir, *Application of Hyperspectral Remote Sensing Role in Precision Farming and Sustainable Agriculture Under Climate Change: A Review*, Climate Change Impacts on Natural Resources, Ecosystems and Agricultural Systems, Springer Climate, pp. 503-520, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[7] Pengfei Ma, Liang Fan, and Genda Chen, "Hyperspectral Reflectance for Determination of Steel Rebar Corrosion and Cl− Concentration," *Construction and Building Materials*, vol. 368, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[8] Jaime Zabalza et al., "Hyperspectral Imaging Based Corrosion Detection in Nuclear Packages," *IEEE Sensors Journal*, vol. 23, no. 21, pp. 25607-25617, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[9] Amit Rotem et al., “Interpretation of Hyperspectral Shortwave Infrared Core Scanning Data Using SEM-Based Automated Mineralogy: A Machine Learning Approach,” *Geosciences*, vol. 13, no. 7, pp. 1-17, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[10] Peichao Li et al., “Spatial Gradient Consistency for Unsupervised Learning of Hyperspectral Demosaicking: Application to Surgical Imaging,” *International Journal of Computer Assisted Radiology and Surgery*, vol. 18, pp. 981-988, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[11] Yuanyuan Guo, Yanwen Chong, and Shaoming Pan, “Hyperspectral Image Compression via Cross-Channel Contrastive Learning,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 61, pp.1-18, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[12] Divya Mohan, J. Aravinth, and Sankaran Rajendran, “Reconstruction of Compressed Hyperspectral Image Using SqueezeNet Coupled Dense Attentional Net,” *Remote Sensing*, vol. 15, no. 11, pp. 1-25, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[13] Yaman Dua, Vinod Kumar, and Ravi Shankar Singh, “Comprehensive Review of Hhyperspectral Image Compression Algorithms,” *Optical Engineering*, vol. 59, no. 9, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[14] Yaman Dua, Ravi Shankar Singh, and Vinod Kumar, “Compression of Multi-Temporal Hyperspectral Images based on RLS Filter,” *The Visual Computer*, vol. 38, pp. 65-75, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[15] Tie Zheng et al., “Recursive Least Squares for Near-Lossless Hyperspectral Data Compression,” *Applied Sciences*, vol. 12, no. 14, pp. 1-11, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[16] Amal Altamimi, and Belgacem Ben Youssef, “A Systematic Review of Hardware-Accelerated Compression of Remotely Sensed Hyperspectral Images,” *Sensors*, vol. 22, no. 1, pp. 1-53, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[17] Shrish Bajpai, “Low Complexity Image Coding Technique for Hyperspectral Image Sensors,” *Multimedia Tools and Applications*, vol. 82, pp. 31233-31258, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[18] Ali Can Karaca, and Mehmet Kemal Güllü, “Superpixel based Recursive Least-squares Method for Lossless Compression of Hyperspectral Images,” *Multidimensional Systems and Signal Processing*, vol. 30, pp. 903-919, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[19] 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]

[20] Lei Liu et al., “Karhunen-Loève Transform for Compressive Sampling Hyperspectral Images,” *Optical Engineering*, vol. 54, no. 1, 2015. [CrossRef] [Google Scholar] [Publisher Link]

[21] Samiran Das, “Hyperspectral Image, Video Compression using Sparse Tucker Tensor Decomposition,” *IET Image Processing*, vol. 15, no. 4, pp. 964-973, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[22] Jyh Wen Chai, Jing Wang, and Chein-I Chang, “Mixed Principal-Component-Analysis/Independent-Component-Analysis Transform for Hyperspectral Image Analysis,” *Optical Engineering*, vol. 46, no. 7, 2007. [CrossRef] [Google Scholar] [Publisher Link]

[23] Rui Dusselaar, and Manoranjan Paul, “Hyperspectral Image Compression Approaches: Opportunities, Challenges, and Future Directions: Discussion,” *Journal of the Optical Society of America A*, vol. 34, no. 12, pp. 2170-2180, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[24] Rui Dusselaar, and Manoranjan Paul, “A Block-based Inter-band Predictor using Multilayer Propagation Neural Network for Hyperspectral Image Compression,” *arXiv*, pp. 1-11, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[25] Azam Karami, Mehran Yazdi, and Grégoire Mercier, “Compression of Hyperspectral Images Using Discerete Wavelet Transform and Tucker Decomposition,” *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 5, no. 2, pp. 444-450, 2012. [CrossRef] [Google Scholar] [Publisher Link]

[26] Qian Du, and James E. Fowler, “Hyperspectral Image Compression Using JPEG2000 and Principal Component Analysis,” *IEEE Geoscience and Remote Sensing Letters*, vol. 4, no. 2, pp. 201-205, 2007. [CrossRef] [Google Scholar] [Publisher Link]

[27] Jiqiang Luo et al., “Lossless Compression for Hyperspectral Image Using Deep Recurrent Neural Networks,” *International Journal of Machine Learning and Cybernetics*, vol. 10, pp. 2619-2629, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[28] 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]

[29] K.S. Gunasheela, and H.S. Prasantha, “Compressive Sensing Approach to Satellite Hyperspectral Image Compression,” *Proceedings of ICTIS 2018 Information and Communication Technology for Intelligent Systems*, vol. 1, pp. 495-503, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[30] Azam Karami, Rob Heylen, and Paul Scheunders, “Hyperspectral Image Compression Optimized for Spectral Unmixing,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, no. 10, pp. 5884-5894, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[31] M. Yahya Masalmah et al., “A Framework of Hyperspectral Image Compression Using Neural Networks,” *Latin American and Caribbean Conference for Engineering and Technology Proceedings*, Santo Domingo, Dominican Republic, vol. 13, pp. 1-6, 2015. [CrossRef] [Google Scholar] [Publisher Link]

[32] Ben Sujitha et al., "Optimal Deep Learning based Image Compression Technique for Data Transmission on Industrial Internet of Things Applications," *Transactions on Emerging Telecommunications Technologies*, vol. 32, no. 7, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[33] Yubal Barrios et al., "SHyLoC 2.0: A Versatile Hardware Solution for On-board Data and Hyperspectral Image Compression on Future Space Missions," *IEEE Access*, vol. 8, pp. 54269-54287, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[34] Jonathan K. Su, Su May Hsu, and Seth Orloff, "Assessment of Effects of Lossy Compression of Hyperspectral Image Data," *Proceedings Algorithms and Technologies for Multispectral, Hyperspectral, and Ultraspectral Imagery X*, Orlando, Florida, United States, vol. 5425, 2004. [CrossRef] [Google Scholar] [Publisher Link]

[35] S. Sanjith, and R. Ganesan, "A Review on Hyperspectral Image Compression," *2014 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT)*, Kanyakumari, India, pp. 1159-1163, 2014. [CrossRef] [Google Scholar] [Publisher Link]

[36] Yaman Dua et al., "Convolution Neural Network based Lossy Compression of Hyperspectral Images," *Signal Processing: Image Communication*, vol. 95, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[37] Emmanuel Christophe, Corinne Mailhes, and Pierre Duhamel, "Hyperspectral Image Compression: Adapting SPIHT and EZW to Anisotropic 3-D Wavelet Coding," *IEEE Transactions on Image Processing*, vol. 17, no. 12, pp. 2334-2346, 2008. [CrossRef] [Google Scholar] [Publisher Link]

[38] Anjaneya, and G.K Rajini, "Light Field Hyper Spectral Lossless Compression Employing Greedy Discrete Wavelet and Poincare Recurrence Network," *International Journal of Intelligent Engineering & Systems*, vol. 16, no. 5, pp. 386-394, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[39] Shrish Bajpai et al., "Curvelet Transform Based Compression Algorithm for Low Resource Hyperspectral Image Sensors," *Journal of Electrical and Computer Engineering*, vol. 2023, pp. 1-18, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[40] Barbara Penna et al., "Transform Coding Techniques for Lossy Hyperspectral Data Compression," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 45, no. 5, pp. 1408-1421, 2007. [CrossRef] [Google Scholar] [Publisher Link]

[41] A. Karami, "Lossy Compression of Hyperspectral Images Using Shearlet Transform and 3D SPECK," *Proceedings Image and Signal Processing for Remote Sensing XXI*, Toulouse, France, vol. 9643, 2015. [CrossRef] [Google Scholar] [Publisher Link]

[42] Cuiping Shi et al., "Remote Sensing Image Compression Based on Direction Lifting-Based Block Transform with Content-Driven Quadtree Coding Adaptively," *Remote Sensing*, vol. 10, no. 7, pp. 1-24, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[43] Lucana Santos et al., "Performance Evaluation of the H.264/AVC Video Coding Standard for Lossy Hyperspectral Image Compression," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 5, no. 2, pp. 451-461, 2012. [CrossRef] [Google Scholar] [Publisher Link]

[44] Mark R. Pickering, and Michael J. Ryan, *An Architecture for the Compression of Hyperspectral Imagery*, Hyperspectral Data Compression, Springer, pp. 1-34, 2006. [CrossRef] [Google Scholar] [Publisher Link]

[45] Ying Hou, and Ying Li, "Hyperspectral Image Lossy-to-Lossless Compression Using 3D SPEZBC Algorithm Based on KLT and Wavelet Transform," *Conference Proceedings Second Sino-foreign-interchange Workshop: International Conference on Intelligent Science and Intelligent Data Engineering*, Xi'an, China, pp. 713-721, 2012. [CrossRef] [Google Scholar] [Publisher Link]

[46] Ying Hou, "Lossy-to-Lossless Compression of Hyperspectral Image Using the 3D Set Partitioned Embedded ZeroBlock Coding Algorithm," *Journal of Software Engineering and Applications*, vol. 2, pp. 86-95, 2009. [Google Scholar] [Publisher Link]

[47] Jiming Fan et al., "Hyperspectral Image Data Compression based on DSP," *Proceedings Optoelectronic Imaging and Multimedia Technology*, vol. 7850, pp. 92-101, 2010. [CrossRef] [Google Scholar] [Publisher Link]

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

[49] Ying Hou, and Guizhong Liu, "3D Set Partitioned Embedded Zero Block Coding Algorithm for Hyperspectral Image Compression," *Proceedings MIPPR 2007: Remote Sensing and GIS Data Processing and Applications; and Innovative Multispectral Technology and Applications*, vol. 6790, 2007. [CrossRef] [Google Scholar] [Publisher Link]

[50] Ruzelita Ngadiran et al., "Efficient Implementation of 3D Listless SPECK," *International Conference on Computer and Communication Engineering (ICCCE'10)*, Kuala Lumpur, Malaysia, pp. 1-4, 2010. [CrossRef] [Google Scholar] [Publisher Link]

[51] 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]

[52] Shrish Bajpai, "Low Complexity and Low Memory Compression Algorithm for Hyperspectral Image Sensors," *Wireless Personal Communications*, vol. 131, pp. 805-833, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[53] Shrish Bajpai, and Naimur Rahman Kidwai, "Fractional Wavelet Filter Based Low Memory Coding for Hyperspectral Image Sensors," *Multimedia Tools and Applications*, vol. 83, pp. 26281-26306, 2024. [CrossRef] [Google Scholar] [Publisher Link]

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

[55] V.K. Sudha, and R. Sudhaka, "3D Listless Embedded Block Coding Algorithm for Compression of Volumetric Medical Images," *Journal of Scientific & Industrial Research*, vol. 72, pp. 735-738, 2013. [Google Scholar]

[56] Harshit Chandra, and Shrish Bajpai, "3D-Block Partitioning Embedded Coding for Hyperspectral Image Sensors," *2023 International Conference on Power, Instrumentation, Energy and Control (PIECON)*, Aligarh, India, pp. 1-5, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[57] Harshi Chandra et al., "3D-Memory Efficient Listless Set Partitioning in Hierarchical Trees for Hyperspectral Image Sensors" *Journal of Intelligent & Fuzzy Systems*, vol. 45, no. 6, pp. 1163-11187, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[58] Shrish Bajpai, Naimur Rahman Kidwai, and Harsh Vikram Singh, "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]

[59] 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]

[60] Shrish Bajpai, Harsh Vikram Singh, and Naimur Rahman Kidwai, "3D Modified Wavelet Block Tree Coding for Hyperspectral Images," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 15, no. 2, pp. 1001-1008, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[61] Shrish Bajpai, "Low Complexity Block Tree Coding for Hyperspectral Image Sensors, *Multimedia Tools and Applications*, vol. 81, pp. 33205-33323, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[62] Harshit Chandra, and Shrish Bajpai, "Listless Block Cube Tree Coding for Low Resource Hyperspectral Image Compression Sensors," *2022 5th International Conference on Multimedia, Signal Processing and Communication Technologies (IMPACT)*, Aligarh, India pp. 1-5, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[63] Ali Kadhim Al-Janabi, "Low Memory Set-Partitioning in Hierarchical Trees Image Compression Algorithm," *International Journal of Video & Image Processing and Network Security IJVIPNS-IJENS*, vol. 13, no. 2, pp. 12-18, 2013. [Google Scholar]

[64] 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]

[65] 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]

[66] Divya Sharma, "Image Quality Assessment Metrics for Hyperspectral Image Compression Algorithms," *2024 Second International Conference Computational and Characterization Techniques in Engineering & Sciences (IC3TES)*, Lucknow, India, pp. 1-5, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[67] Satvik Agrawal et al., "Hyperspectral Image Compression using Modified Convolutional Autoencoder," *International Journal of Computer Information Systems and Industrial Management Applications*, vol. 15, pp. 396-407, 2023. [Google Scholar] [Publisher Link]

[68] Nadia Zikiou, Mourad Lahdir, and David Helbert, "Support Vector Regression-based 3D-Wavelet Texture Learning for Hyperspectral Image Compression," *The Visual Computer*, vol. 36, pp. 1473-1490, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[69] Divya Sharma, Y.K. Prajapati, and Rajeev Tripathi, "0.55 Tb/s Heterogeneous Nyquist-WDM Superchannel using Different Polarization Multiplexed Subcarriers," *Photonic Network Communications*, vol. 39, pp. 120-128, 2020. [CrossRef] [Google Scholar] [Publisher Link]

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

[71] Pallavi Ranjan et al., "Revolutionizing Hyperspectral Image Classification for Limited Labeled Data: Unifying Autoencoder-Enhanced GANs with Convolutional Neural Networks and Zero-Shot Learning," *Earth Science Informatics*, vol. 18, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[72] Vinod Kumar Tripathi, and Shrish Bajpai, "Curvelet Transform Based Hyperspectral Image Compression with Listless Set Partitioned Compression Algorithm for Unmanned Aerial Vehicle Image Sensor," *SSRG International Journal of Electronics and Communication Engineering*, vol. 11, no. 12, pp. 71-82, 2024. [CrossRef] [Google Scholar] [Publisher Link]