Multimedia Tools and Applications (2022) 81:33205–33232
https://doi.org/10.1007/s11042-022-13057-x

Low complexity block tree coding for hyperspectral image sensors

Shrish Bajpai[1]

Received: 7 October 2021 / Revised: 24 December 2021 / Accepted: 4 April 2022 /
Published online: 18 April 2022
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2022

Abstract
Complexity of any onboard hyperspectral image sensor is a challenging issue. The existing hyperspectral image compression algorithm plays a great role in reducing the data transmission bandwidth, data processing time, processing power and coding memory. Many wavelet transform-based set partitioned hyperspectral image compression algorithms are proposed in the past which work with lossy and lossless compression. These compression algorithms use lists or state tables to keep track of significant and insignificant sets or coefficients. The 3D wavelet block tree coding (3D-WBTC) has superior coding performance due to the exploitation of the inter sub-band & intra sub-band redundancy. The 3D-Low-Complexity Block Tree Coding (3D-LCBTC) is a novel implementation of 3D-WBTC which uses two state tables and very small size link lists. The 3D-LCBTC uses depth-first search approach which reduces the complexity of the compression process significantly. Thus, the proposed compression algorithm is a suitable candidate for resources-constrained onboard hyperspectral image sensors.

Keywords Lossy hyperspectral image compression · Wavelet transform · Computational complexity · Zero block cube tree

1 Introduction

With rich spectral information contained in hundreds of continues spectral frames from visible to near-infrared (400 nm to 2500 nm) having spectral resolution of 10 nm [53]. HyperSpectral Image (HSI) has been applied in multiple application ranging from aerospace [22], cultivation [61], climate monitoring [24], document verification [45], earth observation [41], food quality control [40], forestry [46], military reconnaissance [27], pharmaceuticals [34], pollution detection [1], soil health observation [2] etc. Apart from the above applications, remote sensing

✉ Shrish Bajpai
shrishbajpai@gmail.com

[1] Electronics & Communication Engineering, Faculty of Engineering, Integral University, Lucknow, Uttar Pradesh, India

[55] is a one of the most active area of hyperspectral (HS) image processing, where researchers develop the algorithms related to the image compression for faster data transmission [21], feature extraction for target detection [16], image denoising [14], image segmentation [43], change detection [60], object classification for land-cover analysis [42], object recognition [37, 38] etc. The satellite based Hyperspectral image sensor acquire the HSI from their remote platform, process and transmit the same through the high frequency wireless channels [18]. There are hundreds of frequency frames in HSI, the pixel depth in HS image is about 12 bit/pixel to 16 bit/pixel [70]. Due to this a lot of memory has been required to save the few HSI in the onboard sensor memory [4, 5].

It has been known these electronics sensors generate a lot of data [62]. So, compression is a must step to save the memory and increase the performance of the sensor. Beside the storage of the HSI image, transmission bandwidth, sensor performance, power management and data transmission time are the major issues which can address by the HSI compression algorithm [54, 63].

On the basis of data loss the hyperspectral image compression algorithm (HSICA) are classified into the three category lossy, lossless and near lossless [44, 73], while on the basis of the coding process the HS image compression algorithms are classified into the six category prediction based algorithm [72], neural network (NN) based algorithm [39], vector quantization based algorithm [74], machine learning based algorithm [81], transform based algorithm [6] and hybrid compression algorithm [11].

The prediction based algorithm, the predicator (spatial, spectral and spatial-spectral) exploits the spectral correlation between the frames of the HS image and calculate the prediction error. The predictive error is encoded by the entropy coding methods such as huffman coding, or arithmetic coding etc. [80]. The prediction based algorithms are data dependable (compression ratio is depend upon the image) and these algorithms work with the lossless compression only. Adaptive Differential Pulse-Code Modulation (ADPCM), Variable-Length Coding (VLC), LookUp Tables (LUT), Cluster DPCM, Context-based Adaptive Lossless Image Coding (CALIC) are the promising prediction based HS image compression algorithms [26, 78].

The vector quantization (VQ) based compression algorithm constructed the code book (spectral libraries) and it is used when the spectral correlation is stronger than the spatial correlation. The VQ is optimal block coding strategy where the algorithm processes the spectral block compression. The VQ has three stages codebook generation process, encoding procedure and the decoding procedure. The codebook gets training from the input image having many codewords for multiple blocks. The searching of the best codeword for image block is completed in the encoding process. All blocks are encoded as codeword and codebook is also transmitted with these codeword. The image is constructed with the help of received codebook in the decoding process [10].

The transform based HS image compression algorithms works with both lossy and lossless compression. These algorithms use the mathematical transform (Fourier transform, Cosine transform, wavelet transform, Karhunen-Loeve Transform etc) to transform the image data into the domain where the data represented by the less correlated high energy coefficients [76]. These algorithms performance remarkable at the low bit rates during the lossy compression. Karami et al. [32] proposed the compression algorithm with the 3D-DCT and sparse tucker decomposition. Bilgin et al. [12] compressed the HS images through the 3D-EZW with the 3D-DWT. Penna et al. [51, 52] used the Karhunen-Loeve Transform (KTL) for the decorrlation of the spectral correlation in HS image. The KTL with the JPEG2000 (DCT

based) achieved the high coding gain by removing the correlation between the pixels of HS image [35]. Wang et al. gave 3D lapped transformation approach for HS image compression [78].

The NN-based algorithms have the neural network layer structure of multiple layer structure. The complexity in NN based compression algorithm increases rapidly but the coding gain also increases [31]. Before the start of the compression process, the NN is trained with help machine learning algorithms. The NN network with the predictive coding has outstanding prediction capability. Due to less predictive error, the coding gain increases. Autoencoder network, recurrent neural network, feed-forward neural network, radial basis function etc. are the major types of neural networks that the extensive use in the image compression process [56, 57]. Tensor decomposition has also been used along with deep learning technique in CNN-NTD [39, 64].

The machine learning-based HS image compression uses support vector machine, tensor decomposition, deep learning methodology for the HS image compression [50]. Although, it has a high coding gain as NN-based HS image compression algorithm but the complexity is also increased significantly. The wavelet transform is applied before the tensor decomposition. Zhang et al. [81] gave a tensor decomposition approach, which obtained a core tensor that is considered to be the representative of the original tensor. Das [20] presented a tensor-based compression algorithm that works both with HS images and video compression.

The hybrid compression algorithms are the combination of any two methods mention above. Hybrid compression algorithms have high coding efficiency than other types of compression algorithms but this coding gain is obtained at the cost of the high coding complexity. Li te al. [36] presented an algorithm that uses the predictor with the VQ. Báscones et al. [19] uses VQ and PCA for the spectral decorrelation and after that applied JPEG2000 to the obtain principle component for the spatial decorrelation.

Through 3D-LMBTC [7] and 3D-ZM-SPECK [9] reduces the demand of coding memory significantly but they increases the complexity of the compression scheme with respect to the other state of art HS image compression Scheme 3D-LSK [49] and 3D-NLS [65]. The proposed HS image compression scheme is a low memory solution with high coding efficiency and low complexity.

The remaining part of this paper is structured as follows. In Section 2 related work is discussed for the 3D-LCBTC which includes dyadic wavelet transform, set partitioned HS image compression algorithm and 3D-WBTC [6]. Details of the proposed HS image compression algorithm 3D-LCBTC are described in Section 3. The experimental results and discussion are provided in Section 4 while the conclusion is discussed in Section 5.

2 Related work

2.1 Dyadic wavelet transform

The wavelet transform is widely used mathematical transform for the image compression. It is powerful tool to convert the image to other domain having few high energy component (low frequency component). The transform based image compression algorithms work with both lossy (till bit budget available) and lossless compression [9, 71]. So, these algorithms are widely used in compression process. The wavelet transform is applied on the HS image in three ways. Either applies 3D wavelet transform on the whole HS image or apply 2D wavelet

transform frame by frame in spatial domain or apply 2D wavelet transform in spatial domain and then 1D transform in the spectral domain [78].

The performance of wavelet transform is better than other mathematical transform. The single level wavelet transform transforms the HS image into the eight non-overlapping coefficient sub-bands. The scanning order is LLL, LHL, HLL, LLH, HHL, HLH, LHH, and HHH. The H is high while L is low. The transform coefficients present in the LLL sub-band consider as the coarsest coefficients. The n level 3D-DWT can be obtained by repeating the process n time on the HS image cube. The inverse transform is obtained by the sampling and filtering sub-bands to construct the original HS image [3, 79].

2.2 3D set partitioned wavelet transform based HS image

The 3D set partitioned wavelet transform based HS image compression have superiority than other type of transform compression algorithms such that embeddedness, low computational complexity, low coding memory requirement and high coding efficiency [5]. These algorithms further sub divide into three type according to the set partitioned rule.

1. Zero Block Cube Compression Algorithms: This type of algorithms partitioned the HS image cube into the contiguous block cubes. The zero block cube is a block cube in which no significant coefficients with reference to the current threshold. The 3D-SPECK [67], 3D-LSK [49], 3D-SPEZBC [28] and 3D-ZM-SPECK [9] are the well known HSICA under this category.
2. Zero Tree Compression Algorithms: This type of algorithms partitioned the HS image by means of grouping the wavelet coefficients corresponding to the same location and form the spatial orientation tree (SOT) [6, 23]. The SOT is a zero tree having the no significant coefficients with reference to the current threshold. The 3D-SPIHT [68], DDWT-BISK [13] and 3D-NLS [65] are the well known HSICA under this category.
3. Zero Block Cube Tree Compression Algorithms: This type of algorithms combine the useful feature of both, zero block cube and zero tree compression algorithms. These algorithms partitioned the HS image into the contiguous block cubes and after that block cube trees are formed with the roots in the topmost sub-band in a zero tree fashion. The 3D-WBTC [6] and 3D-LMBTC [7] are the well known HSICA under this category. The 3D-WBTC [6] and 3D-LMBTC [7] have remarkable performance at the low bits rates because it can accumulate more insignificant coefficients than 3D-SPIHT [68] and 3D-NLS [65].

Tang et al. proposed the 3D-SPECK [67] which is the extension of the SPECK for the gray scale images. The 3D-LSK [49] is a listless version of 3D-SPECK which uses a coding small memory for the tracking the significance of the partitioned set or coefficient. The 3D-SPIHT [68] and it's listless version 3D-NLS [65] are also state of art petitioned HS image compression algorithms.

The 3D-WBTC [6] utilizes the block cube tree structure to achieve high coding efficiency at low bit rate but at the high bit rate it's performance degrade due to the complex structure and multiple read/ write operation on the linked lists. The 3D-LMBTC [7] is a listless version of 3D-WBTC [6] which reduces the complexity and coding memory. The 3D-ZM-SPECK [9] is the special case of 3D-SPECK [67] which follows the same partitioned rule but it does not uses any link lists or state tables. The demand of memory reduces in 3D-ZM-SPECK [9]

significantly but the complexity was increased compare to the 3D-LSK [65] due to the set or coefficient testing for each bit plane.

The listless HSICA such as 3D-LSK, 3D-NLS, 3D-LMBTC & 3D-ZM-SPECK does not use the linked list to track the significance of the sets or coefficients but they use state tables or marker to track the sets or coefficients. The associated lists the data depended and multiple memory write or read operation increase the complexity of compression process [9]. The coding memory of listless HSICA is fixed and depends on the size & pixel depth of HS image.

Table 1 gives the comparative analysis of some state of art 3D wavelet based set partitioned HS image compression scheme.

2.3 3D wavelet block tree coding (3D-WBTC)

The 3D-WBTC [6] is a block cube tree based set partitioned HS image compression algorithm which exploits the redundancies within sub-band. The 3D-WBTC [6] combines the useful features of 3D-SPECK [67] and 3D-SPIHT [68]. The 3D-WBTC [6] is based on spatial orientation trees (SoT) in which a node is a block cube of 'm x n x p' coefficients rather than to a single coefficient in 3D-SPIHT [68]. Each SoT has a root node that is present in the LLL band of the transform HS image. By creating the block cube tree, eight 3D-SPIHT's SoTs are combined into a single SoT of 3D-WBTC [6]. A set of descendent block cubes are referred as type 'A' block cube tree and the set of grand descendent block cubes are referred as type 'B' block tree.

The 3D-WBTC [6] uses three link lists to store the significance information about the partitioned sets or coefficients: list of insignificant block (LIB), list of insignificant block set

Table 1 Wavelet transform based set partitioned HS image compression algorithm

Compression Algorithm	Reference	Set Partitioned Type	Coding List	Coding Table/ Marker	Coding Memory Requirement	Computation Load
3D-SPECK	[67]	Block Cube	Yes (2)	No	Variable	High
3D-SPIHT	[68]	Tree	Yes (3)	No	Variable	High
3D-WBTC	[6]	Block Cube Tree	Yes (3)	No	Variable	High
3D-SPEZBC	[28]	Block Cube	Yes (2)	No	Variable	Moderate
Adapting SPIHT	[17]	Tree	Yes (3)	No	Variable	Moderate
3-D Wavelet Fractal	[75]	Tree	Yes (3)	No	Variable	High
DDWT-BISK	[13]	Tree	Yes (3)	No	Variable	High
3D-BEZW	[15]	Block	No	Yes	Fixed	Low
AT-3D-SPIHT	[30]	Tree	Yes	No	Variable	High
AT-3D-SPECK	[77]	Block Cube	Yes	No	Variable	High
3D-EZBC	[29]	Block Cube	Yes	No	Variable	High
3D-LSK	[49]	Block Cube	No	Yes	Fixed	Low
3D-NLS	[65]	Tree	No	Yes	Fixed	Low
3D-LMBTC	[7]	Block Cube Tree	No	Yes	Low	Low
3D-WDR	[48]	Tree	Yes (3)	Yes	Variable	Moderate
3D-ASWDR	[48]	Tree	Yes (3)	Yes	Variable	High
3D-ZM-SPECK	[9]	Block Cube	No	No	Zero	Low

(LISB) and list of significant pixel (LSP). The 3D-WBTC [6] initialized with the block cubes in topmost LLL band (left) are added to the LIB while their descendants are added in LIBS. The LSP starts as empty list. Each block cube present in LIB has eight offspring block cubes at the same spatial orientation in the higher frequency sub band. Each bit plane starts with the sorting pass followed by the refinement pass.

In the sorting pass, the coefficients are encoded from top most bit plane and move towards the least significant bit-plane. If a block cube is found insignificant to the current threshold, then '0' is generated for the whole block cube and block cube is remains in LIB and it will again tested for the succeeding threshold. If a block cube is found significant to the current threshold,, then '1' is generated. The block cube is partitioned in to the eight equal small block cubes through the octa tree partitioning. This octa tree partitioning will continue till it reaches to the coefficient level (block cube size of one). The parent block cube is remove from the list. If the coefficient is significant to the current threshold, then '1' will be generated (significance) with the sign bit. The coefficient moves to the LSP. If the coefficient is not significant to the current threshold then it moves to LIB. After octa partitioned and significance testing to the current threshold, the block cube is deleted from the LIB. The insignificant block cube and its descendants are remain in LIBS. A significant type 'A' block cube set is partitioned into type 'B' block cube set with eight offspring block cubes. The type 'B' cube sets are added to the end of LIBS while eight offspring block cubes are tested for current threshold. A significant type 'B' block set is partitioned into type 'A' block cube set and added into the end of LIBS. This process continues till all block cube sets are encoded. After sorting pass, the refinement pass initiated for current threshold and one refinement bit for each coefficient generated. The threshold is reduce by half and process runs till the bit budget exhausts.

3 3D-low complexity block tree coding (3D-LCBTC)

The 3D-LCBTC is a low-weight version of 3D-WBTC [6] which has high coding gain, low coding memory requirement and low complexity. The 3D-LCBTC is a zero block cube tree-based set partitioned hyperspectral image compression algorithm which has the same partitioned rule as 3D-WBTC [6]. Like 3D-WBTC [6], 3D-LCBTC considers a block cube as node in the spatial oriented tree. Through this, a large number of insignificant descendent can be represented by a single coefficient. So, the zero block cube tree HS image compression scheme has higher gain in low bit rates. The node block cube present in the top LLL band has seven offspring nodes in the highest decomposition level sub-bands. The 3D-LCBTC uses the fixed block cube size ($2 \times 2 \times 2$) rather than changing the size of block cube as in 3D-WBTC [6]. The fix size of the block cube reduces the complexity of the proposed compression scheme significantly. Unlike other set partitioned compression schemes, the 3D-LCBTC executes refinement pass before sorting pass.

The 3D-LCBTC uses two state tables and two linked list for the tracking of the partitioned block cube or coefficients. Detail of the state table and linked list is given in Table 2.

The BCSM and DSM are used to store the significance information about each node. The 3D-LCBTC consists of two passes, initialization pass and bit plane pass. Each bit plane pass consist of sorting pass (SP) and refinement pass (RP). Unlike 3D-SPIHT [68] or 3D-WBTC [6], 3D-.

Table 2 Associated state table and linked list used in 3D-LMBTC

State Table	
BCSM	Block Cube Significance Memory
DSM	Descendant Significance Memory
Linked List	
LCBC	List of Child Block Cube
LPBC	List of Parent Block Cube

LCBTC executes refinement pass before sorting pass. The transform coefficients which are significant in last bit plane, are encoded and a refinement bit is generated for each coefficient in the refinement pass. The transform coefficients or block cubes, which have not become significant at previous threshold, are encoded in the sorting pass. The BCSM gives the information about the significance of each block cube in the HS image while DSM gives the information about the associated descendants. A block cube 'a' is significant if BCSM(a) = 1, while the information related to the significance of descendants of all nodes are store in DSM. When all descendants under the node 'a' are significant to the current bit plane then whole the whole block cube tree originated from the node 'a' is significant and it is represented as DSM(a) = '1'. This block cube tree is encode in refinement pass instead of sorting pass. Two associated list LCBC and LPBC are used in the sorting pass.

Initialization: The encoding process starts from the top most bit plane (most significant) 'n' and proceeds towards the lower bit planes till the bit budget is available. The 3D transform HS image is converted to the 1D array C_i through Morton mapping [9]. The highest threshold (n) is calculated by the Eq. 1 and magnitude of the threshold (T) is calculated by Eq. 2

$$n = \lfloor \log_2[\max\{|C_i|\}] \rfloor \tag{1}$$

$$T = 2^{\mathrm{n}} \tag{2}$$

The block cubes present in LLL band of transform HS image is assigned '1' in BCSM while other block cubes are assigned as '0'. All block cubes present in the DSM are assigned as '0'. The associated list LCBC and LPBC are initialized as empty.

Refinement Pass: Refinement pass is executed for those block cubes which are significant to the pervious bit plane. The significant block cubes are present in the BCSM having the tag '1'. Each coefficient of the significant block cube is encoded in RP and a single refinement bit is sent for coefficients which are previously significant. If the coefficients is significant first time (block cube is significant initially), than the sign bit is also sent with the significant bit. The block cubes are encoded in the breadth first approach in which the more significant coefficients are encoded first. The 3D wavelet transform HS image has most of the information present in the higher band than the lower band.

Sorting Pass: Those block cubes which are left in refinement pass are encoded in the sorting pass. Each block cube tree has roots in the LLL band. The sorting pass starts in block cube tree wise. The second block tree will be encoded when the previous block tree is completely encoded. Through this sequence follow, the requirement of the list memory has been reduce significantly and also less number of read/write operation which further reduce the complexity of the proposed compression algorithm.

The significance of the block cube 'B' and block cube tree 'BT' (starting from node 'a') is calculated with the Eq. 3 and Eq. 4

$$S_n(B)\begin{cases}1 & if\ T \le \max\limits_{i\in B}[|C_i|] \le 2T\\ 0 & if\ \max\limits_{i\in B}[|C_i|] < T\\ \phi & if\ \max\limits_{i\in B}[|C_i|] \ge 2T\end{cases} \tag{3}$$

$$S_n(BT) = \begin{cases}1 & if\ T \le \max\limits_{i\in B}[|C_i|] \le 2T\\ 0 & if\ \max\limits_{i\in B}[|C_i|] < T\end{cases} \tag{4}$$

The sorting pass uses the two linked lists LCBC and LPBC. The LCBC is used to encode the coefficients present in the block cube tree. The LPBC will execute after the execution of all entries in LCBC. The LPBC is use to update the state table.

When a block cube at node 'a' with its associated block cube tree is insignificant to the current threshold, only '0' bit is sent to the output. If block cube is significant then '1' is sent to the output and encoding process of performed for all eight coefficients. If all descendant block cubes are insignificant to the current threshold, then '0' is sent to the output otherwise, for the significant descendant block cubes, '1' '0' sent to the output with offsprings are added to the last of LCBC.

The significant offspring block cube is significant to the current threshold, then '1' is sent to the output and encoding process of performed for all eight coefficients.

If a block cube at node 'a' and it's offspring block cubes are significant to the current threshold, then if descendant block cubes are insignificant to the current threshold, then only '0' is sent to the output. The output '1' is generated and the block cubes are added to LCBC and the parent block cube moves to the LPBC.

After the completion of the bit plane the threshold is reduce by half and next pass (bit plane) executed. This process is repeated till the bit budget is available or desire bit rate is achieved. The decoder follows the same procedure as encoder except for the additional step of significance testing of those sets which contains the refinement bit.

The pseudo code for the 3D-LCBTC is given in Table 3. The *Encode* function is used to encode the block cube at node 'a' of the transform HS image. The 'p' is the address of the coefficient while I(p) is the value of the coefficient.

Let consider $B_{s,\,t,\,u}(x)$ is a root block cube present in the LLL band of the transform HS image where 's,t,u' is the address of the coefficient present in the top left of the block cube while 'y'is define as a dimension of the block cube. The $O(x)$ *or* $O_{s,\,t,\,u}(x)$ is the set of offspring block cubes of the root block cube $B_{s,\,t,\,u}(x)$. The offspring block cubes are defined as (Fig. 1).

$B_{s,\,t,\,u}(x)$A root block cube of size 'y x y x y' that have wavelet transform coefficients $[\,C_{j,\,k,\,l}\,|\,s \le j \le (s+y);\ t \le k \le (t+y);\ u \le l \le (u+y)]$

$B_{s,\,t,\,u}(x)$A root block cube of size 'y x y x y' that have wavelet transform coefficients

$$\left[\,C_{j,k,l}\,|\,s \le j \le (s+y);\ t \le k \le (t+y);\ u \le l \le (u+y)\right]$$

$O_{s,\,t,\,u}(x) = [\,B_{2s,\,2t,\,2u}(x),\ B_{2s+y,\,2t,\,2u}(x),\ B_{2s,\,2t+y,\,2u}(x),\ B_{2s,\,2t,\,2u+y}(x)$
$B_{2s+y,\,2t+y,\,2u}(x),\ B_{2s,\,2t+y,\,2u+y}(x),\ B_{2s+y,\,2t,\,2u+y}(x),\ B_{2s+y,\,2t+y,\,2u+y}(x)]$

Table 3. Pseudo code of proposed 3D-LCBTC encoding algorithm.

Algorithm : 3D-LCBTC Encoding Process
M and N are the spatial dimension & P is the spectral dimension of the HS image
Input : The transform HS image (3D) is converted to the 1D array through the Morton mapping
The transform HS image as 1D array (C_i).
Output : Encoded Embedded Bit Stream

INITIALIZATION PASS
Set : $n = \lfloor \log_2 [\max\{|C_i|\}] \rfloor$
Magnitude of the threshold : T = 2^n
Set : $LCBC = \Phi$ & $LPBC = \Phi$
Set : $BCSM\left(1: \frac{MNP}{8}\right) = 0$
Set : $BCSM\left(1: \frac{MNP}{2^{L+1}}\right) = 1$
Set : $DSM\left(1: \frac{MNP}{2^{L+2}}\right) = 0$
Set : Starting index (i) of linear array T as zero
d(x) : Descendants of the block cube $'x'$
O(x) : Offspring of the block cube $'x'$

BIT PLANE PASS
REFINEMENT PASS
$foreach\ block\ cube\ a$ do
$if\ BCSM\ (a) = 1$ then
$Encode\ (a)$
end
end
SORTING PASS
$foreach\ block\ cube\ a$ in LLL band do
$if\ DSM\ (a) = 1$ then
Add a to LCBC
end
$foreach\ block\ cube\ a$ in LCBC do
$if\ BCSM\ (a) = 0$ then
$if\ S_n\ (a) = 0$ & $S_n[d(a)] = 0$ then
$Output \Leftarrow 0$
Remove a from LCBC
else
$Output \Leftarrow 1$
$Encode\ (a)$
Set BCSM(a) = 1
$if\ S_n[d(a)] = 1$ then
$Output \Leftarrow 1$
Move a to LPBC
Add each O(a) to LCBC
else
$Output \Leftarrow 0$
Remove a from LCBC
end
end
else
$foreach\ 'b' \in O(a)$ do
$if\ BSCM(b) = 0$ then
$if\ S_n[d(a)] = 1$ then
$Output \Leftarrow 1$
Move a to LPBC

```
                                    Add b to LCBC
                                else
                                    Output ⇐ 0
                                end
                            else
                                if DSM(b) = 0 then
                                    Add b to LCBC
                                end
                            end
                            Remove a from LCBC
                        end
                    end
                end
                foreach a in LPBC do
                    if DSM [O(a)] = 1 then
                        DSM(a) = 1
                    end
                    Remove a from LPBC
                end
            end

Encode (a)
{
    foreach p ∈ a do
        if I(p) ≥ T then
            if I(p) ≥ 2 * T then
                Output ⇐ 1
            else
                Output ⇐ 1
                sgn
            end
            Output ⇐ 0
            end
    end
}
```

$d_{s,\,t,\,u}(x)$Set of all descendent block cubes of the root block cube $B_{s,\,t,\,u}(x)$

The coding memory used by the 3D-LCBTC is calculated by the memory MEM_{LCBTC} used by the associated lists (LCBC and LPBC) & state tables (BCSM and DSM).

The BCSM is used to store the status of block cube. Since, the size of block cube in 3D-LCBTC is fixed (2 × 2 × 2). Hence, the size of BSM is MNP/8 for the HS image of 'M x N x P'. The DSM is used to store the status of the descendant block cube. For the lowest decomposition level of wavelet transform (L = 1) when no descendant is present. Thus, no entry in DSM is required. Hence, the size of DSM is MNP/64. The size of state tables is fixed throughout the coding process and it depends on the size of the test HS image. So, the requirement of memory used by the state table is fixed. The total memory (bit) MEM_{ST} required by the state tables is given by

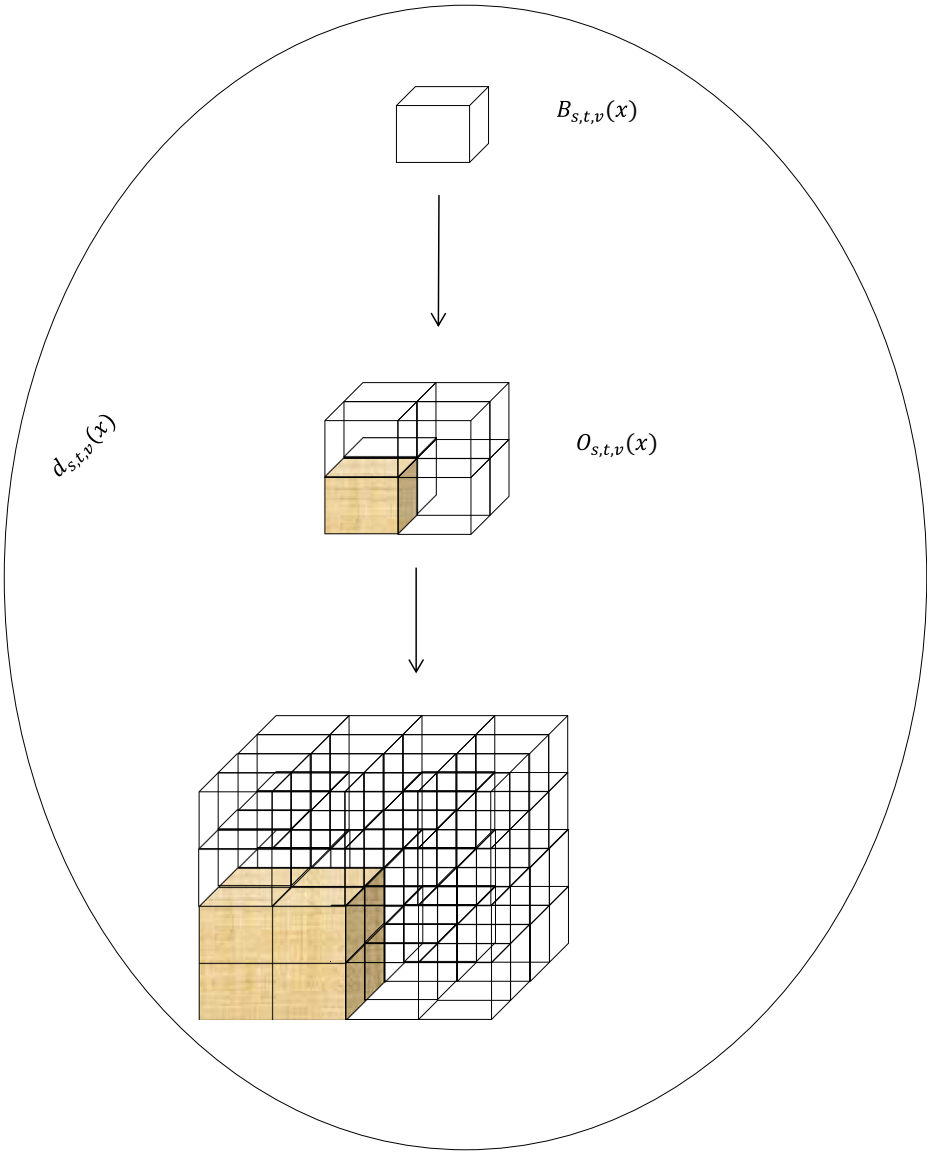

Fig. 1 A block cube tree structure

$$MEM_{ST} = \left[\frac{MNP}{8} + \frac{MNP}{64}\right] = \left[\frac{9*MNP}{64}\right] \tag{5}$$

The 3D-LCBTC used two linked lists LCBC and LPBC. The 3D-LCBTC encodes one block cube tree at a time. Common nodes are present in both lists at a time. The number of nodes depends on the level of the dyadic wavelet transform. The entries in the linked lists changes every time and the linked list memory is referred as dynamic memory. Experimentally, the maximum number of entries in the LCBC list MEM_{LCBC} and LPBC list MEM_{LPBC} calculated as

$$MEM_{LCBC} = [8 + 7(L{-}2)] \tag{6}$$

$$MEM_{LPBC} = 1 + \sum_{n=1}^{L-2} 8^n \tag{7}$$

Each entry in the linked lists is completed by the address bits, which is calculated by the $\log_2 MNP$.

So, the total memory MEM_{LL} required by the linked lists is

$$MEM_{LL} = \log_2(MNP) * \left[9 + 7(L{-}2) + \sum_{n=1}^{L-2} 8^n \right] \tag{8}$$

The coding memory MEM_{LCBTC} required by the 3D-LCBTC encode is given as

$$MEM_{LCBTC} = \left[\frac{9}{64} MNP + \log_2(MNP) * \left\{ 9 + 7(L{-}2) + \sum_{n=1}^{L-2} 8^n \right\} \right] \tag{9}$$

4 Result and analysis

The proposed HSICA is compared with the existing state of art HSICA 3D-SPECK [67], 3D-SPIHT [68], 3D-WBTC [6], 3D-LSK [49], 3D-NLS [65], 3D-LMBTC [7] and 3D-ZM-SPECK [9]. The performance of the proposed compression algorithm 3D-LCBTC is measured on the basis of coding efficiency, coding memory and computational complexity & tested on four publically available HS images which are Washington DC Mall, Urban, Jasper Ridge and Cuprite. The detail description about the HS images is presented in Table 4.

The five level dyadic wavelet transform is used to transform the HS image. The wavelet transform coefficients are quantized to the nearest integer. The morton mapping (linear indexing) is used to convert the transform coefficients from 3D matrix to 1D array [8]. After that, the coefficients are encoding with the help of state of art set partitioned HS image compression algorithms 3D-SPECK [67], 3D-SPIHT [68], 3D-WBTC [6], 3D-LSK [49], 3D-NLS [65], 3D-LMBTC [7] and 3D-ZM-SPECK [9] with proposed 3D-LCBTC. Whole work is performed by using Intel core i3 central processing unit @ 1.6 GHz, RAM of 8 GB, 64 bit operating system and Windows 8.1 operating system. The HS images are cropped from the left top corner to the size of a cube and zero padding is done if it is required. The coding efficiency is calculated in decibel (dB) for Peak Signal to Noise Ratio (PSNR) and Bjøntegaard delta

Table 4 Detail of HS images used for analysis

HS Image Dataset	HS Image Sensor	Wavelength	Spectral Resolution	Spatial Resolution	Spatial Dimension	Spectral Dimension	Pixel Depth
Washington DC Mall	HYDICE	400 nm–2400 nm	10 nm	3 mt - 4 mt	1280×307	191	14
Urban					307×307	210	10
Jasper Ridge	AVIRIS	380 nm–2500 nm	10 nm	4mt - 20 mt	100×100	224	13
Cuprite					250×190	224	16

peak signal-to-noise rate (BD-PSNR) while coding memory and coding complexity is calculated in kilobyte (KB) and second (sec).

4.1 Coding efficiency

The metric used to calculate the coding efficiency are Peak Signal to Noise Ratio (PSNR), Bjøntegaard delta peak signal-to-noise rate (BD-PSNR), Structural Similarity Index (SSIM) and Feature Similarity Index (FSIM) [9].

The Peak Signal to Noise Ratio is the measurement of the rate-distortion (RD) performance. Mathematically, the PSNR is calculated as in Eq. 10 [58].

$$PSNR = 10 \log_{10}\left(\frac{MAX_a * MAX_a}{MSE}\right) \tag{10}$$

The MAX_a is the maximum value of the signal. The MSE (Mean Square Error) is calculated with the Eq. 11

$$MSE = \frac{1}{N_{pix}} \sum_i \sum_j \sum_k [f(i,j,k) - g(i,j,k)]^2 \tag{11}$$

The N_{pix} is the sum of all pixels present in the all frames of HS image. The reconstructed HS image is defines as g(i,j,k) while the original HS image is defined as f(i,j,k).

The compression ratio (CR) is the ratio between the bits used in the original HS image to the bits used in the reconstructed HS image [59]. Mathematically, it is formulated as follows in Eq. 12.

$$\text{Compression Ratio (CR)} = \left[\frac{\text{Original HS image size in bits}}{\text{Decoded bitstream size in bits}}\right] \tag{12}$$

The mathematical relationship between the bit rate (bpppb) and compression ratio for HS image is defined in Eq. 13 [9].

$$\text{Bit Rate (bpppb)} = \left[\frac{\text{Pixel depth} \times \text{Row} \times \text{Column} \times \text{Frequency Frames}}{\text{Compression Ratio}}\right] \tag{13}$$

The high value of compression ratio with the good PSNR shows the robustness of compression algorithm.

The 3D-LCBTC is a zerotree based set partitioned HS image compression scheme and has same set partition rules as 3D-WBTC [6]. The Table 5 gives the comparative analysis of the coding efficiency (PSNR) for 3D-LCBTC with seven state of art set partitioned HS image compression algorithms. From the Table 5, it is clear that the 3D-LCBTC outperform in the high bit rates (from 0.2 to 1) with other compression algorithms. It has been observed from the Table 5 that the variation between the PNSR of proposed 3D-LCBTC and 3D-WBTC [6] exists in the range of −0.49 dB ~ 0.45 dB for Washington DC Mall HS image, −0.27 dB ~ 0.08 dB for Cuprite HS image, −0.03 dB ~ 0.34 dB for Urban HS image and − 0.09 dB ~ 0.22 dB for Jasper Ridge HS image. Similarly, the variation between the PNSR of proposed 3D-LCBTC and 3D-LMBTC [7] exists in the range of −0.36 dB ~ 0.89 dB for Washington DC Mall HS image, −0.11 dB ~ 0.68 dB for Cuprite HS image, −0.01 dB ~ 1.24 dB for Urban HS image and − 0.11 dB ~ 0.98 dB for Jasper Ridge HS image. The variation between the PNSR of proposed 3D-LCBTC and 3D-ZM-SPECK [9] exists in the range of exists in the

Table 5 Comparison of Coding Efficiency (PSNR) between 3D-LCBTC and other compression algorithms for four HS images at fifteen different bit rates

Bit Rate	3D-SPECK [67]	3D-SPIHT [68]	3D-WBTC [6]	3D-LSK [49]	3D-NLS [65]	3D-LMBTC [7]	3D-ZM-SPECK [9]	3D-LCBTC [Proposed Algorithm]
	Washington DC Mall							
0.00625	32.13	32.95	32.12	32.19	32.10	32.93	32.09	32.57
0.0125	33.30	33.08	33.30	33.26	33.05	33.24	33.25	33.11
0.0250	34.56	34.43	34.54	34.47	34.43	34.35	34.44	34.29
0.0375	35.62	35.34	35.52	35.50	35.08	35.16	35.49	35.03
0.05	36.52	36.27	36.47	36.51	36.28	36.52	36.48	36.44
0.1	38.53	38.28	38.50	38.35	38.12	38.29	38.33	38.31
0.2	41.54	41.34	41.52	41.49	41.27	41.19	41.42	41.59
0.3	43.51	43.30	43.49	43.55	43.30	43.48	43.57	43.58
0.4	45.26	45.11	45.25	45.09	45.09	44.59	45.24	45.28
0.5	46.81	46.60	46.81	46.76	46.41	46.09	46.73	46.83
0.6	48.45	48.24	48.43	48.42	48.21	48.38	48.39	48.49
0.7	49.76	49.53	49.74	49.73	49.50	49.17	49.69	49.78
0.8	51.12	50.84	51.09	51.07	50.76	50.28	50.97	51.17
0.9	52.24	52.06	52.22	52.24	52.06	51.67	52.12	52.26
1	53.52	53.32	53.51	53.49	53.33	53.46	53.47	53.59
	Cuprite							
0.00625	18.64	18.03	18.49	18.52	17.99	18.38	18.54	18.28
0.0125	19.52	19.19	19.46	19.45	19.14	19.23	19.44	19.19
0.0250	21.36	21.05	21.25	21.29	20.94	21.18	21.24	21.11
0.0375	22.12	21.60	21.99	22.12	21.54	22.13	22.30	22.07
0.05	23.01	22.63	22.92	22.96	22.61	22.86	22.96	22.76
0.1	25.64	24.67	25.77	25.65	24.61	25.60	25.79	25.49
0.2	30.92	29.44	31.03	30.88	29.33	30.77	30.87	30.84
0.3	34.55	33.36	34.58	34.55	33.27	34.42	34.59	34.61
0.4	38.05	37.04	38.15	38.05	36.97	37.50	38.16	38.18
0.5	41.27	40.51	41.37	41.32	40.45	41.17	41.26	41.39
0.6	43.46	42.58	43.57	43.47	42.50	43.36	43.43	43.52
0.7	45.55	45.00	45.81	45.78	44.89	45.60	45.68	45.83
0.8	47.12	46.43	47.26	47.07	46.38	47.03	47.11	47.16
0.9	48.74	47.95	48.85	48.75	47.91	48.66	48.78	48.91
1	49.83	49.24	49.98	49.86	49.22	49.68	49.71	50.01
	Urban							
0.00625	52.74	52.98	52.72	52.67	52.73	52.86	52.90	52.94
0.0125	53.31	53.22	53.31	53.25	53.22	53.20	53.21	53.28
0.0250	54.34	54.20	54.33	54.26	54.11	54.28	54.30	54.31
0.0375	55.11	55	55.10	55.09	54.99	55.03	55.05	55.12
0.05	55.66	55.55	55.62	55.57	55.54	55.48	55.51	55.61
0.1	57.04	56.94	57.12	57.04	56.92	57.08	57.09	57.14
0.2	58.95	58.8	58.99	58.76	58.65	58.7	58.72	59.02
0.3	60.43	60.29	60.49	60.43	60.29	60.54	60.55	60.53
0.4	61.77	61.67	61.86	61.64	61.65	61.46	61.57	61.91
0.5	62.95	62.79	63.01	62.89	62.64	62.56	62.82	62.99
0.6	64.16	64	64.21	64.05	63.94	64.02	64.02	64.28
0.7	65.37	65.27	65.45	65.35	65.27	65.32	65.32	65.47
0.8	66.33	66.21	66.47	66.29	66.17	65.92	66.22	66.53
0.9	67.36	67.25	67.48	67.23	67.25	66.73	67.14	67.59
1	68.4	68.23	68.59	68.31	68.12	67.69	68.29	68.93
	Jasper Ridge							
0.00625	26.86	26.74	26.84	26.82	26.69	26.79	26.77	26.87
0.0125	28.24	27.75	28.13	28.04	27.67	28.01	28.18	28.21
0.0250	29.84	29.68	29.82	29.73	29.58	29.71	29.74	29.91
0.0375	30.88	30.55	30.90	30.85	30.47	30.73	30.81	30.88

Table 5 (continued)

Bit Rate	3D-SPECK [67]	3D-SPIHT [68]	3D-WBTC [6]	3D-LSK [49]	3D-NLS [65]	3D-LMBTC [7]	3D-ZM-SPECK [9]	3D-LCBTC [Proposed Algorithm]
0.05	32.27	31.97	32.25	32.26	31.88	32.30	32.22	32.19
0.1	35.08	35.11	35.07	35.29	35.04	35.06	35.03	35.14
0.2	39.35	39.13	39.60	39.40	39.01	39.11	39.41	39.51
0.3	41.72	41.89	42.40	41.95	41.74	41.71	41.92	42.62
0.4	44.52	44.41	44.81	44.55	44.31	44.52	44.56	44.97
0.5	45.98	46.33	46.78	46.26	46.24	45.91	46.14	46.89
0.6	48.17	48.19	48.62	48.31	48.07	48.18	48.23	48.83
0.7	49.94	50.06	50.53	50.05	49.94	49.95	49.98	50.73
0.8	51.73	51.74	52.02	51.91	51.96	52.13	52.07	52.17
0.9	52.97	53.09	53.51	53.30	53.01	52.98	53.02	53.68
1	54.77	54.72	55.13	54.86	54.62	54.78	54.92	55.21

Table 6 Comparison of Structural Similarity (SSIM) index between 3D-LCBTC and other compression algorithms for two HS images at fifteen different bit rates

Bit Rate	3D-SPECK [67]	3D-SPIHT [68]	3D-WBTC [6]	3D-LSK [49]	3D-NLS [65]	3D-LMBTC [7]	3D-ZM-SPECK [9]	3D-LCBTC [Proposed Algorithm]
	Washington DC Mall							
0.00625	0.230	0.232	0.227	0.236	0.215	0.230	0.229	0.228
0.0125	0.311	0.311	0.311	0.312	0.311	0.311	0.310	0.311
0.0250	0.385	0.386	0.384	0.385	0.386	0.386	0.384	0.385
0.0375	0.455	0.454	0.453	0.454	0.454	0.456	0.456	0.454
0.05	0.480	0.479	0.480	0.480	0.479	0.480	0.480	0.478
0.1	0.587	0.587	0.585	0.584	0.587	0.589	0.590	0.589
0.2	0.677	0.675	0.678	0.677	0.678	0.677	0.677	0.678
0.3	0.713	0.713	0.713	0.713	0.712	0.714	0.714	0.715
0.4	0.766	0.765	0.765	0.767	0.765	0.771	0.766	0.767
0.5	0.790	0.788	0.787	0.789	0.789	0.788	0.787	0.791
0.6	0.814	0.815	0.814	0.814	0.815	0.814	0.814	0.816
0.7	0.830	0.830	0.829	0.833	0.833	0.836	0.829	0.837
0.8	0.849	0.848	0.852	0.852	0.853	0.851	0.849	0.855
0.9	0.873	0.869	0.872	0.872	0.869	0.872	0.871	0.874
1	0.886	0.886	0.886	0.888	0.887	0.886	0.886	0.889
	Jasper Ridge							
0.00625	0.188	0.180	0.187	0.191	0.183	0.182	0.183	0.185
0.0125	0.241	0.232	0.239	0.236	0.229	0.237	0.238	0.230
0.0250	0.297	0.288	0.299	0.299	0.285	0.300	0.301	0.299
0.0375	0.328	0.319	0.328	0.328	0.323	0.331	0.331	0.328
0.05	0.346	0.338	0.345	0.346	0.341	0.349	0.349	0.345
0.1	0.437	0.431	0.437	0.436	0.430	0.437	0.437	0.437
0.2	0.518	0.513	0.518	0.518	0.514	0.518	0.518	0.518
0.3	0.561	0.558	0.560	0.563	0.560	0.565	0.565	0.560
0.4	0.585	0.582	0.585	0.588	0.582	0.588	0.588	0.587
0.5	0.603	0.601	0.603	0.608	0.606	0.611	0.611	0.603
0.6	0.617	0.616	0.617	0.619	0.615	0.618	0.618	0.619
0.7	0.626	0.626	0.626	0.632	0.628	0.633	0.632	0.627
0.8	0.634	0.633	0.634	0.637	0.635	0.639	0.639	0.64
0.9	0.641	0.641	0.641	0.643	0.642	0.642	0.642	0.643
1	0.645	0.645	0.645	0.648	0.646	0.647	0.647	0.647

Table 7 Comparison of Feature-Similarity (FSIM) index between 3 D-LCBTC and other c ompression algorithms for two HS images at fifteen different bit rates

Bit Rate	3D-SPECK [67]	3D-SPIHT [68]	3D-WBTC [6]	3D-LSK [49]	3D-NLS [65]	3D-LMBTC [7]	3D-ZM-SPECK [9]	3D-LCBTC [Proposed Algorithm]
	Washington DC Mall							
0.00625	0.532	0.529	0.532	0.552	0.539	0.550	0.550	0.547
0.0125	0.565	0.565	0.565	0.565	0.565	0.559	0.559	0.562
0.0250	0.626	0.626	0.625	0.624	0.626	0.616	0.616	0.621
0.0375	0.601	0.594	0.594	0.626	0.630	0.621	0.621	0.594
0.05	0.641	0.642	0.640	0.639	0.643	0.637	0.638	0.638
0.1	0.710	0.695	0.709	0.712	0.712	0.716	0.716	0.708
0.2	0.774	0.759	0.770	0.780	0.779	0.784	0.784	0.771
0.3	0.803	0.804	0.802	0.801	0.790	0.805	0.805	0.801
0.4	0.831	0.825	0.827	0.847	0.826	0.852	0.852	0.829
0.5	0.851	0.844	0.847	0.855	0.856	0.863	0.863	0.848
0.6	0.865	0.865	0.865	0.872	0.865	0.865	0.865	0.874
0.7	0.880	0.880	0.880	0.899	0.886	0.898	0.898	0.884
0.8	0.898	0.893	0.896	0.912	0.901	0.910	0.910	0.911
0.9	0.914	0.912	0.914	0.913	0.912	0.917	0.917	0.917
1	0.919	0.917	0.919	0.920	0.918	0.918	0.918	0.921
	Jasper Ridge							
0.00625	0.286	0.286	0.282	0.290	0.288	0.295	0.295	0.294
0.0125	0.291	0.291	0.294	0.291	0.291	0.297	0.297	0.296
0.0250	0.293	0.293	0.289	0.294	0.286	0.302	0.301	0.299
0.0375	0.294	0.301	0.294	0.293	0.292	0.304	0.303	0.302
0.05	0.305	0.307	0.305	0.300	0.302	0.317	0.316	0.315
0.1	0.321	0.319	0.320	0.321	0.320	0.338	0.338	0.334
0.2	0.443	0.434	0.443	0.452	0.439	0.488	0.488	0.479
0.3	0.554	0.548	0.548	0.572	0.558	0.599	0.599	0.587
0.4	0.650	0.627	0.650	0.659	0.628	0.665	0.658	0.651
0.5	0.694	0.692	0.694	0.699	0.696	0.742	0.750	0.744
0.6	0.715	0.708	0.715	0.732	0.712	0.749	0.759	0.756
0.7	0.801	0.794	0.799	0.805	0.804	0.787	0.787	0.792
0.8	0.820	0.820	0.819	0.819	0.822	0.840	0.840	0.824
0.9	0.833	0.827	0.833	0.839	0.834	0.849	0.849	0.847
1	0.871	0.867	0.872	0.869	0.867	0.873	0.873	0.874

range of - 0.46 dB ~ 0.48 dB for Washington DC Mall HS image, − 0.30 dB ~ 0.30 dB for Cuprite HS image, − 0.02 dB ~ 0.64 dB for Urban HS image and - 0.03 dB ~ 0.75 dB for Jasper Ridge HS image. For the perfect reconstruction of any HS image after the compression process, the numerical value of PSNR should be infinity [33], but the image degradation having PSNR numeric value of 40 dB or higher is invisible by human eyes [69].

Table 8 Bjøntegaard Delta PSNR gain of 3D-LCBTC with other HS image compression algorithms for 15 different bit rates

HS Image	3D-SPECK [67]	3D-SPIHT [68]	3D-WBTC [6]	3D-LSK [49]	3D-NLS [65]	3D-LMBTC [7]	3D-ZM-SPECK [9]
Washington DC Mall	−0.1010	0.0386	−0.0714	−0.0401	0.1703	0.1020	−0.0178
Cuprite	−0.1111	0.5990	−0.1107	−0.0883	0.6660	0.0437	−0.1196
Urban	0.0582	0.1523	0.0312	0.1264	0.2109	0.1753	0.1225
Jasper Ridge	0.2246	0.3598	0.0618	0.1878	0.4463	0.2982	0.2248

Table 9 Comparison of coding memory between 3D-LCBTC and other compression algorithm s for four HS images at fifteen different bit rates

Bit Rate	3D-SPECK [67]	3D - SPIHT [68]	3D-WBTC [6]	3D-LSK [49]	3D-NLS [65]	3D-LMBTC [7]	3D-ZM-SPECK [9]	3D-LCBTC [Proposed Algorithm]
	Washington DC Mall							
0.00625	26.09	27.84	17.88	512	1024	12	0	300.59
0.0125	33.39	37.62	34.84	512	1024	12	0	300.59
0.0250	54.87	54.42	55.21	512	1024	12	0	300.59
0.0375	96.22	103.4	99.35	512	1024	12	0	300.59
0.05	136.1	145.3	137.9	512	1024	12	0	300.59
0.1	243.8	263.3	250.1	512	1024	12	0	300.59
0.2	416.3	438	416	512	1024	12	0	300.59
0.3	701.1	628.6	704	512	1024	12	0	300.59
0.4	733.8	723.6	733	512	1024	12	0	300.59
0.5	1048.8	1060.5	1049	512	1024	12	0	300.59
0.6	1191.1	1222.6	1195.4	512	1024	12	0	300.59
0.7	1277.3	1302.1	1281.7	512	1024	12	0	300.59
0.8	1407.7	1415.3	1404.4	512	1024	12	0	300.59
0.9	1702.5	1725.5	1704.6	512	1024	12	0	300.59
1	1802.5	1826.7	1724.6	512	1024	12	0	300.59
	Cuprite							
0.00625	18.47	19.67	15.03	512	1024	12	0	300.59
0.0125	42.63	43.33	32.07	512	1024	12	0	300.59
0.0250	66.20	68.79	52.77	512	1024	12	0	300.59
0.0375	120.6	129.7	102.1	512	1024	12	0	300.59
0.05	124.7	156.7	111.7	512	1024	12	0	300.59
0.1	277.7	277.6	282.8	512	1024	12	0	300.59
0.2	414.5	434.3	417.2	512	1024	12	0	300.59
0.3	544.3	514.7	546.3	512	1024	12	0	300.59
0.4	601.9	576.5	594.5	512	1024	12	0	300.59
0.5	671.2	701.9	674.5	512	1024	12	0	300.59
0.6	854	783.7	857.6	512	1024	12	0	300.59
0.7	947.5	865.5	971.7	512	1024	12	0	300.59
0.8	1065.3	964.6	1057.3	512	1024	12	0	300.59
0.9	1158.5	1182.4	1159.5	512	1024	12	0	300.59
1	1286.1	1308.2	1292.1	512	1024	12	0	300.59
	Urban							
0.00625	18.40	21.63	19.09	512	1024	12	0	300.59
0.0125	27.83	30.59	28.23	512	1024	12	0	300.59
0.0250	70.15	79.18	72.53	512	1024	12	0	300.59
0.0375	95.16	105.20	97.38	512	1024	12	0	300.59
0.05	109.4	112.7	110.3	512	1024	12	0	300.59
0.1	293.2	299.4	294.3	512	1024	12	0	300.59
0.2	478.1	529.8	483.5	512	1024	12	0	300.59
0.3	841.2	864	842.7	512	1024	12	0	300.59
0.4	841.2	867	842.7	512	1024	12	0	300.59
0.5	1076.6	1110.3	1077	512	1024	12	0	300.59
0.6	1419.4	1444.9	1424.8	512	1024	12	0	300.59
0.7	1491.8	1499.2	1493.5	512	1024	12	0	300.59
0.8	1563.4	1586.4	1564.1	512	1024	12	0	300.59
0.9	1590.3	1714.6	1589.8	512	1024	12	0	300.59
1	1889.5	1906.4	1888.9	512	1024	12	0	300.59
	Jasper Ridge							
0.00625	13.02	15.46	14.25	512	1024	12	0	300.59
0.0125	32.95	37.73	34.66	512	1024	12	0	300.59
0.0250	55.23	55.52	55.61	512	1024	12	0	300.59
0.0375	99.37	109.9	102.8	512	1024	12	0	300.59

Table 9 (continued)

Bit Rate	3D-SPECK [67]	3D - SPIHT [68]	3D-WBTC [6]	3D-LSK [49]	3D-NLS [65]	3D-LMBTC [7]	3D-ZM-SPECK [9]	3D-LCBTC [Proposed Algorithm]
0.05	143.02	145.9	143.3	512	1024	12	0	300.59
0.1	241.4	245.9	245.8	512	1024	12	0	300.59
0.2	440	445.7	443.7	512	1024	12	0	300.59
0.3	541.3	555.3	549.3	512	1024	12	0	300.59
0.4	729.6	759.5	741.6	512	1024	12	0	300.59
0.5	821.6	808.9	827.9	512	1024	12	0	300.59
0.6	1099.9	1123.2	1106.7	512	1024	12	0	300.59
0.7	1123.8	1145.9	1131.2	512	1024	12	0	300.59
0.8	1178.9	1192.7	1189.1	512	1024	12	0	300.59
0.9	1443.2	1468	1450.3	512	1024	12	0	300.59
1	1492.7	1503.8	1532.6	512	1024	12	0	300.59

The Structural Similarity Index Measure (SSIM) is an image quality metric that calculates the similarity between two images (original HS image and reconstructed HS image). It is has been.

noticed from the Table 6 that SSIM index is similar to all HS image compression algorithms. Mathematically, SSIM [47] is calculated with the Eq. 14

$$\mathrm{SSIM}(x,y) = \frac{\left(2\mu_x\mu_y + C_1\right)\left(2\sigma_{xy} + C_2\right)}{\left(\mu_x^2 + \mu_y^2 + C_1\right)\left(\sigma_x^2 + \sigma_y^2 + C_2\right)} \tag{14}$$

The input HS image is represented as 'x' while reconstructed image after compression process is represented as 'y'. The mean average of the input HS image 'x' and reconstructed HS image 'y' is μ_x *and* μ_y. The variance of the input HS image 'x' and reconstructed HS image 'y' is σ_x^2 *and* σ_y^2. The covariance between these HS images is given as σ_{xy}. The correction factors are represented by the C_1 & C_2.

The Feature-Similarity (FSIM) index maps the features and measures the similarities between the input HS image and reconstructed HS image [66]. It has been observed from Table 7 that FSIM index is similar to all HS image compression algorithms.

Bjontegaard metric calculation which is known as BD-PSNR [25] is calculated for all four HS images under test. It had been noticed from the Table 8 that 3D-LCBTC outperformed 3D-SPIHT [68], 3D-NLS [49] and 3D-LMBTC [7]. The 3D-LCBTC has a superior performance with reference to other compression algorithms for the two HS images Urban and Jasper Ridge.

4.2 Coding memory

The 3D-LCBTC uses both state table and linked lists for the tracking of the significant block cubes or coefficients. The 3D-LCBTC utilized the best features of the listless algorithm and list- based algorithm. From Eq. 9, the coding memory is calculated and it is fixed throughout the compression process. The calculation of coding memory is calculated when the demand of memory is highest. The state table requires 288 KB memory and linked lists need 12.59 KB of coding memory. The memory required for the coding process is the sum of these two and it is

Table 10 Comparison of encoding time (coding complexity) between 3D-LCBTC and other compression algo rithms fo r four HS images at fifteen different bit rates

Bit Rate	3D-SPECK [67]	3D-SPIHT [68]	3D-WBTC [6]	3D-LSK [49]	3D-NLS [65]	3D-LMBTC [7]	3D-ZM-SPECK [9]	3D-LCBTC [Proposed Algorithm]
	Washington DC Mall							
0.00625	0.30	0.92	0.85	0.29	0.32	0.91	0.40	0.31
0.0125	1.48	1.08	1.23	0.37	0.44	1.09	0.63	0.39
0.0250	3.09	1.44	1.71	0.43	0.51	1.33	0.83	0.47
0.0375	5.30	2.06	2.16	0.46	0.57	1.58	1.01	0.51
0.05	6.23	2.71	2.78	0.55	0.65	1.90	1.11	0.68
0.1	25	7.5	6.50	0.80	0.91	3.90	1.78	0.76
0.2	57.9	25.8	24.8	1.10	1.21	5.10	2.81	1.04
0.3	92.1	37.5	32	1.50	1.65	7.70	3.68	1.41
0.4	269.7	117.9	195.5	2.00	2.12	9.70	5.69	1.93
0.5	414.8	140.1	211.2	2.50	2.64	11.30	7.41	2.44
0.6	576	166.4	247.9	2.90	3.02	13.30	7.99	2.82
0.7	887.5	405.7	625	3.20	3.37	18.10	9.66	3.13
0.8	1130.5	474.2	710.2	3.80	3.96	20	9.91	3.85
0.9	1334.6	555.7	746	4	4.14	20.60	12.53	4.04
1	1497.5	575	804	4.41	4.57	21.10	13.21	4.38
	Cuprite							
0.00625	1.06	0.39	0.92	0.18	0.22	0.58	0.50	0.21
0.0125	1.46	0.69	1.01	0.33	0.41	0.92	0.59	0.37
0.0250	3.01	1.17	1.45	0.43	0.53	1.40	0.80	0.47
0.0375	4.69	2.00	1.93	0.49	0.62	1.61	0.96	0.52
0.05	7.20	2.67	2.46	0.57	0.74	2.22	1.16	0.61
0.1	17.3	6.3	4.7	0.9	1.12	3.2	1.78	0.86
0.2	55.8	26	16.6	1.2	1.54	6.8	3.01	1.09
0.3	107.9	45.5	39.1	2	2.27	7.1	4.08	1.92
0.4	182.3	75.6	68.2	2.1	2.41	9.2	5.21	2.07
0.5	276.1	95.4	93.3	2.2	2.58	11.1	6.32	2.11
0.6	298.4	161.7	155.7	3.4	3.61	12.9	7.55	3.24
0.7	438.8	179.2	202.2	3.9	4.21	15	8.76	3.79
0.8	558.7	198.5	358.5	4.2	4.48	16.5	9.66	4.02
0.9	656.1	282.8	371	4.4	4.69	18.1	11.20	4.12
1	905.1	364	652.5	5	5.23	20.3	15.89	5.02
	Urban							
0.00625	0.59	0.85	0.72	0.34	0.39	0.93	0.45	0.34
0.0125	1.65	0.88	1.19	0.37	0.43	1.04	0.69	0.39
0.0250	3.01	1.47	1.49	0.40	0.52	1.28	0.85	0.43
0.0375	5.12	2.17	2.25	0.49	0.61	1.57	1.17	0.51
0.05	7.64	3.09	3.10	0.51	0.67	1.80	1.36	0.57
0.1	15.60	6.80	6.90	1.60	1.81	3.50	2.25	1.53
0.2	49.50	19.70	19.10	3.00	3.19	5.60	3.50	2.96
0.3	85.70	48.90	26.40	4.10	4.37	7.80	4.57	4.02
0.4	312.4	102.1	191.8	5.60	5.89	10.70	6.09	5.55
0.5	416.2	158.1	253.6	7.10	8.24	11.70	7.06	7.07
0.6	886.9	206.7	300.3	8.80	9.07	13.20	8.15	8.64
0.7	905.1	211.9	371.7	9.50	10.71	16.90	9.40	9.39
0.8	1125.2	541.8	788.7	11.10	11.34	18.30	10.60	11.11
0.9	1542.7	774.1	1067.1	12.70	13.02	19.50	12.16	12.67
1	1702.8	780.2	1184.5	14.40	14.53	20.90	12.96	14.39
	Jasper Ridge							
0.00625	0.73	0.85	0.88	0.37	0.42	0.74	0.41	0.41
0.0125	1.44	0.93	1.30	0.41	0.47	1.07	0.65	0.44
0.0250	3.01	1.34	1.66	0.49	0.53	1.38	0.86	0.51
0.0375	4.45	2.03	2.02	0.51	0.59	1.65	1.01	0.53

Table 10 (continued)

Bit Rate	3D-SPECK [67]	3D-SPIHT [68]	3D-WBTC [6]	3D-LSK [49]	3D-NLS [65]	3D-LMBTC [7]	3D-ZM-SPECK [9]	3D-LCBTC [Proposed Algorithm]
0.05	6.01	2.55	2.36	0.56	0.67	2.06	1.14	0.61
0.1	21.1	7.6	6.4	0.9	1.09	3	1.77	0.88
0.2	54.2	20.6	17.7	1.2	1.34	5.2	2.84	1.11
0.3	100.6	39.4	42.8	1.5	1.64	7.5	4.76	1.47
0.4	150.9	47.8	70.7	2.2	2.32	9.3	5.90	2.18
0.5	315.3	101.6	182.4	2.6	2.74	10.9	6.24	2.58
0.6	356	115.3	227.5	3.03	3.21	13.5	7.65	2.97
0.7	426.1	232.3	480.9	3.2	3.42	15.2	8.70	3.14
0.8	585.7	382.3	676.4	3.7	3.95	17.8	14.45	3.62
0.9	701.2	415	771.9	4	4.29	18.8	12.98	4.04
1	757.3	425.4	942.8	5.1	5.34	22.7	15.84	5.07

fixed. Table 5 gives the comparative analysis of the coding memory use by the different compression algorithms. Due to the two types of separate state tables (BCSM and DSM), the coding memory requirement is higher than the 3D-LMBTC [7] and 3D-ZM-SPECK [9], but superior to the 3D-NLS [65] and 3D-LSK [49].

The 3D-ZM-SPECK [9] is a novel implementation of 3D-SPECK [61], which does not require any coding memory as the refinement pass is merged with the sorting pass. The 3D-LMBTC [7] requires only four types of coefficients; hence the requirement of coding memory is also very less. The 3D-LSK [49] and 3D-NLS [65] uses 4 bits/coefficient marker and 8 bits/coefficient marker. The 3D-LCBTC outperforms the list based set petitioned compression algorithms at the high bit levels (higher bit rate at 0.1 bpppb) (Table 9).

4.3 Coding complexity

The coding complexity of any image compression algorithm is measured by the time required for the encoding process (compression) and decoding process (reconstruction). The encoding time is always greater than the decoding time because the encoder took more time to find the set size and testing of the sets for each threshold. Table 10 shows the encoding time and Table 11 shows the decoding time. It has been observed from the result that 3D-LCBTC out perform with all other compression algorithms for the high bit rate (bpppb = 0.1) except for cuprite HS image at bpppb 1. For the low bit rates 3D-LCBTC outperform with other compression algorithms except 3D-LSK [44]. The 3D-LCBTC performance is superior than 3D-LMBTC [7] and 3D-WBTC [6]. The low complexity is due to the fixed block cube size while for 3D-LMBTC [7] and 3D-WBTC [6], the block cube size is kept changing according to the significance of the block cube and block cube tree. The size of the state tables used in the 3D-LCBTC is small with respect to the 3D-NLS [56]. Accessing memory multiple time creates the load on the sensor and increases the complexity of the compression algorithm. The coding complexity of the listless compression algorithm (3D-LSK, 3D-NLS, 3D-LMBTC, 3D-ZM-SPECK) is always less than the list-based compression algorithm (3D-SPECK, 3D-SPIHT, 3D-WBTC). The 3D-LSK [44] uses 4 bits/coefficient marker and 3D-NLS [56] uses 8 bits/coefficient marker. The 3D-LMBTC [7] uses four types of marker (2 bits/coefficient marker). The coding complexity of 3D-ZM-SPECK [9] is higher because compression algorithm has to test the sets are coefficients for each threshold.

Table 11 Comparison of decoding time (coding complexity) between 3D-LCBTC and other c ompression algorithms for four HS images a t fifteen different bit rates

Bit Rate	3D-SPECK [67]	3D-SPIHT [68]	3D-WBTC [6]	3D-LSK [49]	3D-NLS [65]	3D-LMBTC [7]	3D-ZM-SPECK [9]	3D-LCBTC [Proposed Algorithm]
	Washington DC Mall							
0.00625	0.28	0.26	0.18	0.23	0.27	0.31	0.31	0.29
0.0125	0.62	0.77	0.36	0.33	0.31	0.44	0.62	0.35
0.0250	0.79	0.99	0.75	0.38	0.42	0.64	0.79	0.41
0.0375	0.94	1.93	1.08	0.42	0.49	0.83	0.94	0.46
0.05	1.09	2.58	1.36	0.47	0.57	1.00	1.01	0.51
0.1	17.40	6.10	5.00	0.70	0.79	2.30	1.71	0.64
0.2	48.80	24.80	22.50	1.07	1.07	3.30	2.71	1.01
0.3	75.40	34.80	28.50	1.45	1.43	4.90	3.59	1.33
0.4	264.2	106.3	180.4	1.70	1.94	8.10	5.41	1.62
0.5	339.1	135.4	191.7	2.20	2.31	7.70	6.82	2.07
0.6	532.4	149.6	244.6	2.60	2.79	9.80	7.95	2.48
0.7	807.6	327.1	558	2.70	3.04	11.60	8.80	2.53
0.8	1058.1	448.9	675.3	3.10	3.67	13.40	9.38	3.09
0.9	1142.3	486.2	725	3.20	3.93	13.60	11.82	3.33
1	1289.7	504	774	3.70	4.24	15.50	12.31	3.87
	Cuprite							
0.00625	0.81	0.25	0.45	0.12	0.19	0.40	0.42	0.15
0.0125	0.96	0.32	0.53	0.23	0.34	0.41	0.54	0.27
0.0250	1.77	0.61	0.60	0.36	0.47	0.69	0.79	0.39
0.0375	2.62	1.24	0.87	0.39	0.58	0.91	0.94	0.4
0.05	5.03	1.83	1.38	0.44	0.69	1.18	1.14	0.42
0.1	13.40	5.00	3.10	0.70	0.94	2.20	1.70	0.63
0.2	46.70	22.10	14.60	1.00	1.32	4.80	2.92	0.91
0.3	93.70	40.20	35.40	1.80	2.04	5.50	3.98	1.69
0.4	162.5	70.10	65.80	1.90	2.31	7.00	5.07	1.81
0.5	236.1	88.30	91.50	2.00	2.45	8.50	6.01	1.92
0.6	281.2	160.9	149	2.90	3.38	10.20	7.17	2.79
0.7	435	175.8	196.8	3.10	4.03	11.90	8.28	2.97
0.8	525.9	195.3	316	3.80	4.24	13.10	9.21	2.61
0.9	599.2	273.5	366.9	4.00	4.47	15.00	10.34	3.83
1	884.4	346.6	596	4.50	5.07	15.90	14.03	4.38
	Urban							
0.00625	0.20	0.63	0.28	0.29	0.32	0.24	0.27	0.30
0.0125	0.99	0.74	0.39	0.33	0.37	0.43	0.66	0.34
0.0250	1.70	1.07	0.61	0.37	0.41	0.61	0.79	0.41
0.0375	3.42	1.26	1.24	0.43	0.53	0.82	1.09	0.47
0.05	5.68	2.18	2.08	0.49	0.53	1.00	1.13	0.58
0.1	11.80	4.30	4.80	1.50	1.67	1.70	1.90	1.48
0.2	41.60	16.80	16.40	2.40	2.94	3.20	2.90	2.36
0.3	72.30	42.50	23.20	3.20	4.09	4.70	3.65	3.14
0.4	292.2	88.3	184.5	4.70	5.74	6.00	4.66	4.63
0.5	388.4	148.8	243.2	5.40	7.95	7.30	5.40	5.33
0.6	487.2	197.2	294.7	6.10	8.87	8.60	5.89	6.07
0.7	592.7	201.1	365.7	8.20	10.52	10.00	7.08	8.11
0.8	1066.7	507.4	771.7	8.80	11.08	11.40	7.96	8.72
0.9	1506.4	749.8	1052.3	9.40	12.84	12.60	8.76	9.33
1	1661.8	761.9	1173.4	10	14.22	13.90	9.50	10.04
	Jasper Ridge							
0.00625	0.67	0.78	0.80	0.31	0.31	0.34	0.38	0.33
0.0125	0.75	0.83	1.15	0.35	0.42	0.44	0.64	0.38
0.0250	1.89	1.22	1.19	0.38	0.47	0.65	0.83	0.41
0.0375	2.66	1.54	1.79	0.41	0.51	0.85	1.00	0.45

Table 11 (continued)

Bit Rate	3D-SPECK [67]	3D-SPIHT [68]	3D-WBTC [6]	3D-LSK [49]	3D-NLS [65]	3D-LMBTC [7]	3D-ZM-SPECK [9]	3D-LCBTC [Proposed Algorithm]
0.05	3.50	1.98	2.04	0.47	0.58	1.02	1.12	0.57
0.1	15.3	7.41	4.57	0.75	0.97	1.9	1.73	0.71
0.2	37	17.6	15.3	1.1	1.22	3.7	2.70	1.04
0.3	84.7	37.1	40	1.4	1.51	6.1	4.73	1.41
0.4	128.8	45	69	2.1	2.17	8.7	5.14	2.04
0.5	290.8	98.5	178.4	2.5	2.59	9.3	6.06	2.47
0.6	330.9	101.8	217.1	2.8	2.98	10.5	7.23	2.74
0.7	386.4	232.2	432.1	3	2.17	11.3	8.41	2.97
0.8	487.8	382.2	608.4	3.5	3.72	12.9	9.86	3.46
0.9	667.4	402.7	673.5	3.6	3.97	17.5	11.73	3.55
1	726.9	421.3	923.1	4.3	5.02	21.8	12.31	4.24

4.4 Effect of block cube size on the performance of 3D-LCBTC

In order to analyze the impact of the block cube size on the performance of proposed HS image compression algorithm, the coding efficiency (PSNR & SSIM), coding memory and coding complexity (encoding time & decoding time) is used for the analysis of effect of block cube size.

It had been noticed from the Table 12 that increase in the block cube size moderately improving the coding gain at the low bit rates but for the high bit rate, it almost same or degraded. The coding gain reduces because increase in the block cube size also increases the location of significance bits or significance block cubes. It also required the extra bits in searching for

Table 12 Effect of different block cube size for the proposed HS image compression algorithm on PSNR, SSIM, encoding time & decoding time for the Washington DC Mall HS image

Block Cube Size	Bit Rate (bpppb)				
	0.0125	0.05	0.1	0.5	1
	Peak Signal to Noise Ratio				
2×2×2	33.11	36.44	38.31	46.83	53.59
4×4×4	33.14	36.41	38.29	46.8	53.57
8×8×8	33.17	36.38	38.26	46.77	53.53
	Structural Similarity (SSIM) index				
2×2×2	0. 385	0.478	0.589	0.791	0.889
4×4×4	0.386	0.476	0.587	0.789	0.886
8×8×8	0.388	0.471	0.582	0.784	0.881
	Coding Memory				
2×2×2	300.59	300.59	300.59	300.59	300.59
4×4×4	37.57	37.57	37.57	37.57	37.57
8×8×8	4.7	4.7	4.7	4.7	4.7
	Encoding Time				
2×2 × 2	0.39	0.68	0.76	2.44	4.38
4×4×4	0.38	0.68	0.76	2.44	4.38
8×8×8	0.38	0.67	0.76	2.44	4.38
	Decoding Time				
2 × 2 × 2	0.35	0.51	0.64	2.07	3.87
4 × 4 × 4	0.35	0.52	0.65	2.09	3.88
8 × 8 × 8	0.35	0.55	0.71	2.18	3.97

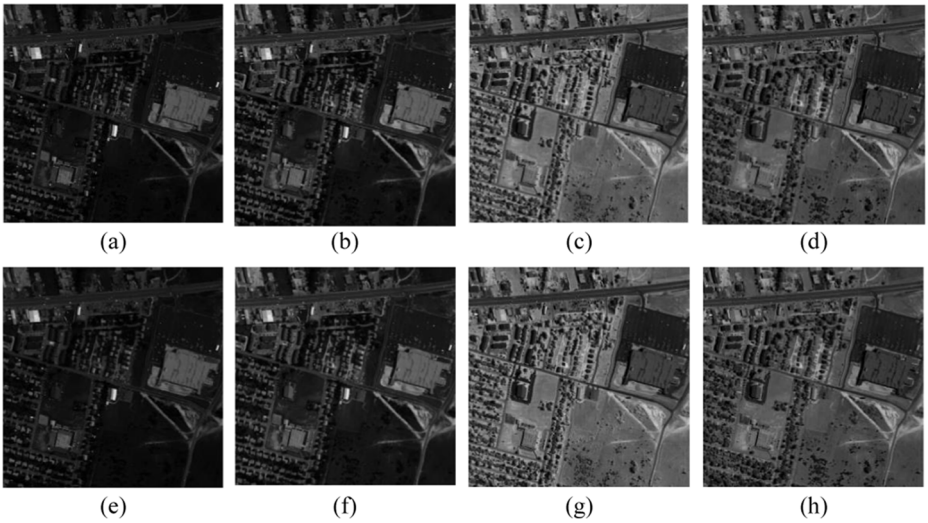

Fig. 2 Original Urban HS image (a) Frame 25 (b) Frame 50 (c) Frame 100 (d) Frame 175 Reconstructed Urban HS image with CR 16 (e) Frame 25 (f) Frame 50 (g) Frame 100 (h) Frame 175

significant coefficients which let slightly increase in the coding complexity (due to multiple memory read or write operation). However, the requirement of coding memory reduces significantly due to the use of large block cube which limited the number of bits in the output list.

The visual representation (original and reconstructed) of four sub-bands (Band 25, 50, 100 and 175) for Urban HS image and Washington DC MALL HS image at CR = 16 is shown in Figs. 2 and Fig. 3. The Fig. 2 shows the visual representation of four frames (25, 50, 100 and 175) for Urban HS image and Fig. 3 shows the visual representation of four frames (25, 50, 100 and 175) for Washington DC MALL HS image.

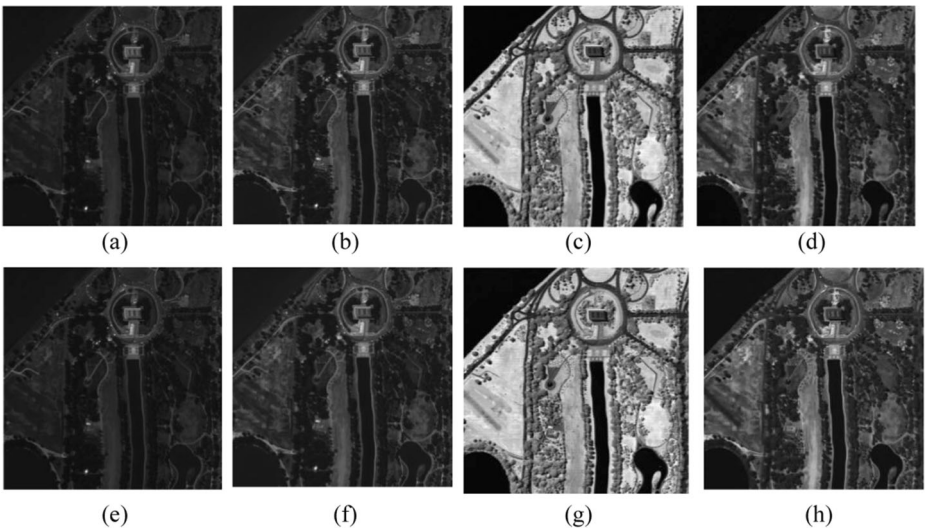

Fig. 3 Original Washington DC MALL HS image (a) Frame 25 (b) Frame 50 (c) Frame 100 (d) Frame 175 Reconstructed Washington DC MALL HS image with CR 16 (e) Frame 25 (f) Frame 50 (g) Frame 100 (h) Frame 175

5 Conclusion

The onboard HS image sensor has limited resources to process the HS images and transmit to the earth station. This paper proposes a novel algorithm that outperforms other state of art wavelet-based HS image compression algorithms on coding efficiency and coding complexity. The 3D-LCBTC is a compression algorithm which uses both state tables and linked list. The requirement of coding memory of 3D-LCBTC is higher than 3D-LMBTC which is due to the fixed size of the block cube. The fixed size of the block cube increases the demand of state table memory. The proposed HS image compression algorithm performs efficient compression without affecting the performance of succeeding applications.

Acknowledgements I am sincerely thankful to the anonymous reviewers for their critical comments and suggestions to improve the quality of the paper.

Funding The author received no financial support for the research, authorship, and/or publication of this article.

Declarations

Conflict of interest The author declares that there are no conflicts of interest.

References

1. Achard V, Foucher PY, Dubucq D (2021) Hydrocarbon pollution detection and mapping based on the combination of various hyperspectral imaging processing tools. Remote Sens 13(5):1020. https://doi.org/10.3390/rs13051020
2. Anand R, Veni S, Aravinth J (2017) Big data challenges in airborne hyperspectral image for urban landuse classification. In 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI): 1808–1814. https://doi.org/10.1109/ICACCI.2017.8126107
3. Bairagi VK, Sapkal AM, Gaikwad MS (2013) The role of transforms in image compression. Journal of The Institution of Engineers (India): Series B 94(2):135–140. https://doi.org/10.1007/s40031-013-0049-9
4. Bajpai S, Singh HV, Kidwai NR (2017) Feature extraction & classification of hyperspectral images using singular spectrum analysis & multinomial logistic regression classifiers. In IEEE International Conference on Multimedia, Signal Processing and Communication Technologies (IMPACT) Aligarh, India: 97-100. 10.1109/MSPCT.2017.8363982
5. Bajpai, Shrish, Harsh Vikram Singh, and Naimur Rahman Kidwai (2019) 3D modified wavelet block tree coding for hyperspectral images. Indonesian Journal of Electrical Engineering and Computer Science (IJEECS) 15 (2): 1001–1008. https://doi.org/10.11591/ijeecs.v15.i2.pp1001-1008
6. Bajpai S, Kidwai NR, Singh HV (2019) 3D wavelet block tree coding for hyperspectral images. International Journal of Innovative Technology and Exploring Engineering 8(6C):64–68
7. Bajpai S, Kidwai NR, Singh HV, Singh AK (2019) Low memory block tree coding for hyperspectral images. Multimed Tools Appl 78(19):27193–27209. https://doi.org/10.1007/s11042-019-07797-6
8. Bajpai, Shrish, Naimur Rahman Kidwai, Vishal Singh Chandel (2020) Low memory wavelet based hyperspectral image coding using 2D Dyadic Wavelet Transform, 11(6): 25–33. https://doi.org/10.34218/IJEET.11.6.2020.003
9. Bajpai S, Kidwai NR, Singh HV, Singh AK (2022) A low complexity hyperspectral image compression through 3D set partitioned embedded zero block coding. Multimed Tools Appl 81:841–872. https://doi.org/10.1007/s11042-021-11456-0
10. Báscones D, González C, Mozos D (2020) An FPGA accelerator for real-time lossy compression of hyperspectral images. Remote Sens 12(16):2563. https://doi.org/10.3390/rs12162563
11. Ben S, Parvathy VS, Laxmi Lydia E, Rani P, Polkowski Z, Shankar K (2020) Optimal deep learning based image compression technique for data transmission on industrial Internet of things applications Transactions on Emerging Telecommunications Technologies, e3976. https://doi.org/10.1002/ett.3976

12. Bilgin A, Zweig G, Marcellin MW (2000) Three-dimensional image compression with integer wavelet transforms. Appl Opt 39(11):1799–1814. https://doi.org/10.1364/AO.39.001799
13. Boettcher JB, Du Q, Fowler JE (2007) Hyperspectral image compression with the 3D dual-tree wavelet transform. IEEE International Geoscience and Remote Sensing Symposium: 1033-1036. https://doi.org/10.1109/IGARSS.2007.4422977
14. Chen Y, Huang TZ, He W, Zhao XL, Zhang H, Zeng J (2021). Hyperspectral image Denoising using factor group sparsity-regularized nonconvex low-rank approximation. IEEE Trans Geosci Remote Sens https://doi.org/10.1109/TGRS.2021.3110769.
15. Cheng KJ, Dill J (2014) Lossless to lossy dual-tree BEZW compression for hyperspectral images. IEEE Trans Geosci Remote Sens 52(9):5765–5770. https://doi.org/10.1109/TGRS.2013.2292366
16. Cheng T, Wang B (2021) Decomposition model with background dictionary learning for hyperspectral target detection. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing 14: 1872–1884. https://doi.org/10.1109/JSTARS.2021.3049843
17. Christophe E, Mailhes C, Duhamel P (2008) Hyperspectral image compression: adapting SPIHT and EZW to anisotropic 3-D wavelet coding. IEEE Trans Image Process 17(12):2334–2346. https://doi.org/10.1109/TIP.2008.2005824
18. Chutia D, Bhattacharyya DK, Sarma KK, Kalita R, Sudhakar S (2016) Hyperspectral remote sensing classifications: a perspective survey. Trans GIS 20(4):463–490. https://doi.org/10.1111/tgis.12164
19. Daniel B, González C, Mozos D (2018) Hyperspectral image compression using vector quantization, PCA and JPEG2000. Remote Sens 10(6):907. https://doi.org/10.3390/rs10060907
20. Das S (2021) Hyperspectral image, video compression using sparse tucker tensor decomposition. IET Image Process 15(4):964–973. https://doi.org/10.1049/ipr2.12077
21. Datta A, Ghosh S, Ghosh A (2017) Supervised feature extraction of hyperspectral images using partitioned maximum margin criterion. IEEE Geosci Remote Sens Lett 14(1):82–86. https://doi.org/10.1109/LGRS.2016.2628078
22. Dmitriev EV, Kozoderov VV, Dementyev AO, Safonova AN (2018) Combining classifiers in the problem of thematic processing of hyperspectral aerospace images. Optoelectronics, Instrumentation and Data Processing 54(3):213–221. https://doi.org/10.3103/S8756699018030019
23. Dragotti PL, Poggi G, Ragozini ARP (2000) Compression of multispectral images by three-dimensional SPIHT algorithm. IEEE Trans Geosci Remote Sens 38(1):416–428. https://doi.org/10.1109/36.823937
24. Dussarrat P, Theodore B, Coppens D, Standfuss C, Tournier B (2021) Introduction to the ringing effect in satellite hyperspectral atmospheric spectrometry. Atmospheric Measurement Techniques Discussions: 1–12. https://doi.org/10.5194/amt-2021-121
25. Gnutti A, Guerrini F, Adami N, Migliorati P, Leonardi R (2021) A wavelet filter comparison on multiple datasets for signal compression and denoising. Multidim Syst Sign Process 32(2):791–820. https://doi.org/10.1007/s11045-020-00753-w
26. Goetz AF (2009) Three decades of hyperspectral remote sensing of the earth: a personal view. Remote Sens Environ 113(1):S5–S16. https://doi.org/10.1016/j.rse.2007.12.014
27. Gross W, Queck F, Vögtli M, Schreiner S, Kuester J, Böhler J, Middelmann W (2021) A multi-temporal hyperspectral target detection experiment: evaluation of military setups. In Target and Background Signatures VII 11865:38–48. https://doi.org/10.1117/12.2597991
28. Hou Y, Liu G (2007) 3D set partitioned embedded zero block coding algorithm for hyperspectral image compression. Remote Sensing and GIS Data Processing and Applications; and Innovative Multispectral Technology and Applications. International Society for Optics and Photonics 6790:679056. https://doi.org/10.1117/12.750975
29. Hou Y, Liu G (2008). Hyperspectral image lossy-to-lossless compression using the 3D embedded Zeroblock coding alogrithm. International Workshop on Earth Observation and Remote Sensing Applications: 1-6. https://doi.org/10.1109/EORSA.2008.4620308
30. Hou Y, Liu G (2008) Lossy-to-lossless compression of hyperspectral image using the improved AT-3D SPIHT algorithm. International Conference on Computer Science and Software Engineering 2:963–966. https://doi.org/10.1109/CSSE.2008.1351
31. Jiang Z, Pan WD, Shen H (2020) Spatially and spectrally concatenated neural networks for efficient lossless compression of hyperspectral imagery. Journal of Imaging 6(6):38. https://doi.org/10.3390/jimaging6060038
32. Karami A, Yazdi M, Asli, AZ (2010) Hyperspectral image compression based on tucker decomposition and discrete cosine transform. In 2010 2nd international conference on image processing theory, Tools and Applications: 122-125. https://doi.org/10.1109/IPTA.2010.5586739
33. Kidwai NR, Khan E, Zm-Speck RM (2016) A fast and memoryless image coder for multimedia sensor networks. IEEE Sensors J 16(8):2575–2587. https://doi.org/10.1109/JSEN.2016.2519600

34. Laureen C, Sacré P-Y, Dispas A, De Bleye C, Fillet M, Ruckebusch C, Hubert P, Ziemons E (2021) Pixel-based Raman hyperspectral identification of complex pharmaceutical formulations. Anal Chim Acta 1155: 338361. https://doi.org/10.1016/j.aca.2021.338361
35. Lee HS, Younan NH, King RL (2002) Hyperspectral image cube compression combining JPEG-2000 and spectral decorrelation. IEEE International Geoscience and Remote Sensing Symposium 6:3317–3319. https://doi.org/10.1109/IGARSS.2002.1027168
36. Li R, Pan Z, Wang Y (2019) The linear prediction vector quantization for hyperspectral image compression. Multimed Tools Appl 78(9):11701–11718. https://doi.org/10.1007/s11042-018-6724-8
37. Liu R, Cai W, Li G, Ning X, Jiang Y (2021). Hybrid dilated convolution guided feature filtering and enhancement strategy for hyperspectral image classification. IEEE Geoscience and Remote Sensing Letters: 1–5. https://doi.org/10.1109/LGRS.2021.3100407
38. Liu R, Ning X, Cai W, Li G (2021) Multiscale dense cross-attention mechanism with covariance pooling for hyperspectral image scene classification. Mob Inf Syst 2021:1–15. https://doi.org/10.1155/2021/9962057
39. Luo Y, Qin J, Xiang X, Tan Y, Liu Q, Xiang L (2020) Coverless real-time image information hiding based on image block matching and dense convolutional network. J Real-Time Image Proc 17(1):125–135. https://doi.org/10.1007/s11554-019-00917-3
40. Medus LD, Saban M, Francés-Víllora JV, Bataller-Mompeán M, Rosado-Muñoz A (2021) Hyperspectral image classification using CNN: application to industrial food packaging. Food Control 125:107962. https://doi.org/10.1016/j.foodcont.2021.107962
41. Mishra MK, Gupta A, John J, Shukla BP, Dennison P, Srivastava SS, Kaushik NK, Misra A, Dhar D (2019) Retrieval of atmospheric parameters and data-processing algorithms for AVIRIS-NG Indian campaign data. Current Science 116(7):1089–1100. https://doi.org/10.18520/cs/v116/i7/1089-1100
42. Mitran T, Sreenivas K, Janakirama Suresh KG, Sujatha G, Ravisankar T (2021) Spatial prediction of calcium carbonate and clay content in soils using airborne hyperspectral data. Journal of the Indian Society of Remote Sensing 49:1–12. https://doi.org/10.1007/s12524-021-01415-5C
43. Miyoshi GT, Imai NN, Tommaselli AMG, Honkavaara E, Näsi R, Moriya ÉAS (2018) Radiometric block adjustment of hyperspectral image blocks in the Brazilian environment. Int J Remote Sens 39(15–16):4910–4930. https://doi.org/10.1080/01431161.2018.1425570
44. Mohan BK, Porwal A (2015) Hyperspectral image processing and analysis. Curr Sci 108(5):833–841
45. Morales A, Ferrer MA, Diaz-Cabrera M, Carmona C, Thomas GL (2014). The use of hyperspectral analysis for ink identification in handwritten documents. In 2014 International Carnahan Conference on Security Technology: 1-5. https://doi.org/10.1109/CCST.2014.6986980
46. Munmun B, Kumar SA, Praise SD (2021) Two-level band selection framework for hyperspectral image classification. Journal of the Indian Society of Remote Sensing 49(4):843–856. https://doi.org/10.1007/s12524-020-01262-w
47. Nadia Z, Lahdir M, Helbert D (2019) Support vector regressionbased 3D-wavelet texture learning for hyperspectral image compression. Vis Comput 36(7):1473–1490. https://doi.org/10.1007/s00371-019-01753-z
48. Nagendran R, Vasuki A (2020) Hyperspectral image compression using hybrid transform with different wavelet-based transform coding. Int J Wavelets Multiresolut Inf Process 18(01):1941008. https://doi.org/10.1142/S021969131941008X
49. Ngadiran R, Boussakta S, Sharif B, Bouridane A (2010) Efficient implementation of 3D listless SPECK. IEEE international conference on computer and communication engineering, 1–4. https://doi.org/10.1109/ICCCE.2010.5556843
50. Paul A, Kundu A, Chaki N, Dutta D, Jha CS (2021). Wavelet enabled convolutional autoencoder based deep neural network for hyperspectral image denoising. Multimedia tools and applications: 1-27. https://doi.org/10.1007/s11042-021-11689-z
51. Penna B, Tillo T, Magli E, Olmo G (2006). A new low complexity KLT for lossy hyperspectral data compression. In 2006 IEEE International Symposium on Geoscience and Remote Sensing: 3525-3528. https://doi.org/10.1109/IGARSS.2006.904
52. Penna B, Tillo T, Magli E, Olmo G (2007) Transform coding techniques for lossy hyperspectral data compression. IEEE Trans Geosci Remote Sens 45(5):1408–1421. https://doi.org/10.1109/TGRS.2007.894565
53. Plaza A, Benediktsson JA, Boardman JW, Brazile J, Bruzzone L, Camps-Valls G, Chanussot J, Fauvel M, Gamba P, Gualtieri A, Marconcini M (2009) Recent advances in techniques for hyperspectral image processing. Remote Sens Environ 113:S110–S122. https://doi.org/10.1016/j.rse.2007.07.028
54. Raikwar SC, Tapaswi S, Chakraborty S (2021) Bounding function for fast computation of transmission in single image dehazing. Multimed Tools Appl 81:1–24. https://doi.org/10.1007/s11042-021-11752-9
55. Ramakrishnan D, Bharti R (2015) Hyperspectral remote sensing and geological applications. Curr Sci 108(5):879–891

56. Ren W, Zhang J, Ma L, Pan J, Cao X, Zuo W, Liu W, Yang MH (2018). Deep non-blind deconvolution via generalized low-rank approximation. Advances in neural information processing systems: 297-307
57. Ren W, Pan J, Zhang H, Cao X, Yang MH (2020) Single image dehazing via multi-scale convolutional neural networks with holistic edges. Int J Comput Vis 128(1):240–259. https://doi.org/10.1007/s11263-019-01235-8
58. Rupali B (2018) Enhanced encrypted reversible data hiding algorithm with minimum distortion through homomorphic encryption. Journal of Electronic Imaging 27(2):023017. https://doi.org/10.1117/1.JEI.27.2.023017
59. Rupali B (2021) An improved reversible and secure patient data hiding algorithm for telemedicine applications. J Ambient Intell Humaniz Comput 12(2):2915–2929. https://doi.org/10.1007/s12652-020-02449-2
60. Saha S, Kondmann L, Zhu XX (2021) Deep no learning approach for unsupervised change detection in hyperspectral images. ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences 3:311–316. https://doi.org/10.5194/isprs-annals-V-3-2021-311-2021
61. Sahoo RN, Ray SS, Manjunath KR (2015) Hyperspectral remote sensing of agriculture. Curr Sci 108(5):848–859
62. Sharma D, Prajapati YK, Tripathi R (2018) Spectrally efficient 1.55 Tb/s Nyquist- WDM superchannel with mixed line rate approach using 27.75 Gbaud PM-QPSK and PM-16QAM. Optical Engineering 57(7):076102. https://doi.org/10.1117/1.OE.57.7.076102
63. Sharma D, Prajapati YK, Tripathi R (2018) Success journey of coherent PM-QPSK technique with its variants: a survey. IETE Tech Rev 37(1):36–55. https://doi.org/10.1080/02564602.2018.1557569
64. Subrahmanyam KV, Kumar KK, Reddy NN (2019) New insights into the convective system characteristics over the Indian summer monsoon region using space-based passive and active remote sensing techniques. IETE Tech Rev 37(2):211–219. https://doi.org/10.1080/02564602.2019.1593890
65. Sudha VK, Sudhakar R (2013) 3D listless embedded block coding algorithm for compression of volumetric medical images. J Sci Ind Res 72:735–748
66. Suresh KR, Manimegalai P (2019) Near lossless image compression using parallel fractal texture identification. Biomedical Signal Processing and Control 58:101862. https://doi.org/10.1016/j.bspc.2020.101862
67. Tang X, Pearlman WA (2004) Lossy-to-lossless block-based compression of hyperspectral volumetric data. IEEE International Conference on Image Processing, Singapore 5:3283–3286. https://doi.org/10.1109/ICIP.2004.1421815
68. Tang X, Pearlman WA (2006) Three-dimensional wavelet-based compression of hyperspectral images. In hyperspectral data compression springer, Boston, MA: 273-308. https://doi.org/10.1007/0-387-28600-4_10
69. Tausif M, Kidwai NR, Khan E, Reisslein M, FrWF-based LMBTC (2015) Memory-efficient image coding for visual sensors. IEEE Sensors J 15(11):6218–6228. https://doi.org/10.1109/JSEN.2015.2456332
70. Uddin MP, Mamun MA, Hossain MA (2021) PCA-based feature reduction for hyperspectral remote sensing image classification. IETE Tech Rev 38(4):377–396. https://doi.org/10.1080/02564602.2020.1740615
71. UmaMaheswari S, SrinivasaRaghavan V (2021) Lossless medical image compression algorithm using tetrolet transformation. J Ambient Intell Humaniz Comput 12(3):4127–4135. https://doi.org/10.1007/s12652-020-01792-8
72. Valsesia D, Magli E (2017) Fast and lightweight rate control for onboard predictive coding of hyperspectral images. IEEE Geosci Remote Sens Lett 14(3):394–398. https://doi.org/10.1109/LGRS.2016.2644726
73. Vura S, Patil P, Patil SB (2021) A study of different compression algorithms for multispectral images. Materials Today: Proceedings. https://doi.org/10.1016/j.matpr.2021.06.175
74. Wang X, Tao J, Shen Y, Qin M, Song C (2018) Distributed source coding of hyperspectral images based on three-dimensional wavelet. J Indian Soc Remote Sens 46(4):667–673. https://doi.org/10.1007/s12524-017-0735-1
75. Wei P, Yi Zou, Lu AO (2008). A compression algorithm of hyperspectral remote sensing image based on 3-D wavelet transform and fractal. 3rd International Conference on Intelligent System and Knowledge Engineering 1: 1237–1241. https://doi.org/10.1109/ISKE.2008.4731119
76. Wildenstein D, George AD (2021). Towards intelligent compression of hyperspectral imagery. In 2021 IEEE international conference on electronics, Computing and Communication Technologies: 1-6. 10.1/CONECCT52877.2021.9622585
77. Wu J, Wu Z, Wu C (2006) Lossy to lossless compressions of hyperspectral images using three-dimensional set partitioning algorithm. Opt Eng 45(2):027005. https://doi.org/10.1117/1.2173996
78. Yaman D, Kumar V, Singh RS (2020) Comprehensive review of hyperspectral image compression algorithms. Opt Eng 59(9):090902. https://doi.org/10.1117/1.OE.59.9.090902

79. Yaman D, Kumar V, Singh RS (2021) Parallel lossless HSI compression based on RLS filter. Journal of Parallel and Distributed Computing 150:60–68. https://doi.org/10.1016/j.jpdc.2020.12.004
80. Yaman D, Singh RS, Parwani K, Lunagariya S, Kumar V (2021) Convolution neural network based lossy compression of hyperspectral images. Signal Process Image Commun 95:116255. https://doi.org/10.1016/j.image.2021.116255
81. Zhang L, Zhang L, Tao D, Huang X, Du B (2015) Compression of hyperspectral remote sensing images by tensor approach. Neurocomputing 147:358–363. https://doi.org/10.1016/j.neucom.2014.06.052

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.