Progressive resolution coding of hyperspectral imagery featuring region of interest access

Xiaoli Tang and William A. Pearlman

ECSE Department, Rensselaer Polytechnic Institute, Troy, NY, USA 12180-3590

ABSTRACT

We propose resolution progressive Three-Dimensional Set Partitioned Embedded bloCK (3D-SPECK), an embedded wavelet based algorithm for hyperspectral image compression. The proposed algorithm also supports random Region-Of-Interest (ROI) access. For a hyperspectral image sequence, integer wavelet transform is applied on all three dimensions. The transformed image sequence exhibits a hierarchical pyramidal structure. Each subband is treated as a code block. The algorithm encodes each code block separately to generate embedded sub-bitstream. The sub-bitstream for each subband is SNR progressive, and for the whole sequence, the overall bitstream is resolution progressive. Rate is allocated amongst the sub-bitstreams produced for each block. We always have the full number of bits possible devoted to that given scale, and only partial decoding is needed for the lower than full scales. The overall bitstream can serve the lossy-to-lossless hyperspectral image compression. Applying resolution scalable 3D-SPECK independently on each 3D tree can generate embedded bitstream to support random ROI access. Given the ROI, the algorithm can identify ROI and reconstruct only the ROI. The identification of ROI is done at the decoder side. Therefore, we only need to encode one embedded bitstream at the encoder side, and different users at the decoder side or the transmission end could decide their own different regions of interest and access or decode them. The structure of hyperspectral images reveals spectral responses that would seem ideal candidates for compression by 3D-SPECK. Results show that the proposed algorithm has excellent performance on hyperspectral image compression.

Keywords: Hyperspectral image compression, 3D-SPIHT, 3D-SPECK, resolution scalable coding, ROI coding

1. INTRODUCTION

Hyperspectral imaging is a powerful technique and has been widely used in a large number of application, such as detection and identification of the surface and atmospheric constituents present, analysis of soil type, monitoring agriculture and forest status, environmental studies, and military surveillance. Hyperspectral images are generated by collecting hundreds of narrow and contiguous spectral bands of data such that a complete reflectance spectrum can be obtained for each point in the region being viewed by the instrument. However, at the time we gain high resolution spectrum information, we generate massively large image data sets. Access and transport of these data sets will stress existing processing, storage and transmission capabilities. As an example, the Airborne Visible InfraRed Imaging Spectrometer (AVIRIS) instrument, a typical hyperspectral imaging system, can yield about 16 Gigabytes of data per day. Therefore, efficient compression should be applied to these data sets before storage and transmission.[7]

However, utilization of the data in compressed form can often be inconvenient and intractable, if it requires full decompression. One would like the bitstream to have properties of scalability and random access. There are two types of scalability of interest for hyperspectral images - SNR or quality scalability and resolution scalability. SNR scalability means that a portion of the bit stream can be decoded to provide a reconstruction

Further author information: (Send correspondence to W.A.P.)
W.A.P.: E-mail: pearlw@ecse.rpi.edu, Telephone: 1 518 276 6082, Fax: 1 518 276 8715
This work was performed at Rensselaer Polytechnic Institute and was supported in part by National Science Foundation Grant No. EEC-981276. The government has certain rights in this material.

at lower rate or quality. That would allow faster or lower bandwidth transmission or a quick look at the entire data set at lower quality. Resolution scalability would permit decoding at reduced resolution from a portion of the compressed bit stream. These scalability properties enable transmission and retrieval that are progressive by quality or resolution. We presented the SNR scalable 3D-SPECK[10] in previous paper. In this paper, we concentrate on resolution scalable 3D-SPECK.

Resolution scalability is important also in the sense that it is connected to complexity scalability as the consumptions of memory and computational resource is commonly exponentially increased or reduced from one resolution level to another. Therefore, this feature can be used for the application of an image server.

For some applications, the image analyst may select only a subsection of an image to isolate a homogenous region of a material class or end members. Therefore, it is important to access or coding the ROI and it can save space and coding time. We demonstrate in this paper that 3D-SPECK also has ROI retrievability.

Hyperspectral imagery has an important property that it has numerous high frequency content. Therefore, hyperspectral image compression algorithm should also have excellent performance on images with numerous high frequency content.

There are some algorithms proposed recently for hyperspectral image compression. Ryan and Arnold[8] proposed mean-normalized vector quantization (M-NVQ) for lossless AVIRIS compression. Each block of the image is converted into a vector with zero mean and unit standard variation. Motta[5] *et al.* proposed a VQ based algorithm that involved locally optimal design of partitioned vector quantizer for the encoding of source vectors drawn from hyperspectal image. Harsanyi and Chang[2] applied Principle Component Analysis (PCA) on hyperspectral images to simultaneously reduce the data dimensionality, suppress undesired or interfering spectral signature, and classify the spectral signature of interest. All these algorithm have promising performance on hyperspectral image compression. however, none of them generates embedded bitstream.

To incorporate the embedded coding requirement and maintain other compression performances, many promising volumetric image compression algorithms based on wavelet transform were proposed recently. Several widely used ones are Three-Dimensional Context-Based Embedded Zerotrees of Wavelet coefficients (3D-CB-EZW), Three-Dimensional Set Partitioning In Hierarchical Trees (3D-SPIHT),[4] 3D-SPECK,[10] and Annex of Part II of JPEG2000[11] standard for multi-component imagery compression.

Among these wavelet based embedded image compression algorithms, 3D-SPECK has very good performance on hyperspectral image compression; it performs excellently on image sequences with numerous high frequency content. It's simple, efficient, and with low computational complexity.

Most embedded algorithms in the literature are unable to efficiently generate resolution scalable codestreams due to the entanglement in coding, modeling, and data structure across different resolution. In particular, the classical zerotree coders with individual zerotrees spanning several subband scales are not efficient for resolution scalable coding. 3D-SPECK, however, is designed to have the block based structure which is easy to support resolution scalable coding. The original 3D-SPECK supports SNR progressive coding. It is easy to implement 3D-SPECK to support resolution progressive coding as well. 3D-SPECK can also generate ROI-retrievable bitstream to support random ROI access.

This paper is organized as following: We first present resolution scalable 3D-SPECK in section 2, followed by experimental results in section 3. Section 4 will conclude this study.

2. THE RESOLUTION SCALABLE 3D-SPECK ALGORITHM

2.1. Set-up and Terminology

3D-SPECK is an extended and modified version of 2D-SPECK.[3] Consider an image sequence which has been adequately transformed using the discrete wavelet transform (we use integer wavelet transform in this paper). The transformed image sequence is said to exhibit a hierarchical pyramidal structure defined by the levels of decomposition, with the topmost level being the root. Figure 1, illustrates such a structure with three-level decomposition. The finest pixels lie at the bottom level of the pyramid while the coarsest pixels lie at the top

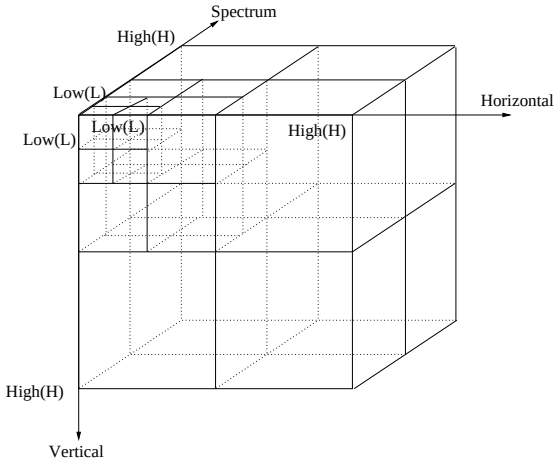

Figure 1. Structure for 3D-SPECK. The numbers on the front lower left corners for each subband are marked to indicate the sorting order.

(root) level. The image sequence is represented by an indexed set of transformed coefficients $c_{i,j,k}$, located at pixel position (i, j, k) in the transformed image sequence.

Pixels are grouped together in sets which comprise regions in the transformed images. Unlike 2D-SPECK, 3D-SPECK has only one type of set: $\mathcal{S}$ set. We say a set $\mathcal{S}$ is significant with respect to n, if

$$\max_{(i,j,k)\in\mathcal{S}} |c_{i,j,k}| \geq 2^n \tag{1}$$

Where $c_{i,j,k}$ denotes the transformed coefficients at coordinate (i, j, k). Otherwise it is insignificant. For convenience, we can define the significance function of a set $\mathcal{S}$ as:

$$\Gamma_n(\mathcal{S}) = \begin{cases} 1 & : \text{ if } 2^n \leq \max_{(i,j,k)\in\mathcal{S}} |c_{i,j,k}| < 2^{n+1} \\ 0 & : \text{ else} \end{cases} \tag{2}$$

Resolution scalable 3D-SPECK makes use of rectangular prisms in the wavelet transform. Each subband in the pyramidal structure is treated as a code block or prism, henceforth referred to as sets $\mathcal{S}$, and can be of varying dimensions. The dimension of a set $\mathcal{S}$ depends on the dimension of the original images and the subband level of the pyramidal structure at which the set lies.

We define the size of a set to be the number of elements in the set. The size of a set can be 1, which means that the set consists of just one pixel. Sets of various sizes will be formed during the course of the algorithm, depending on the characteristics of pixels in the original set.

The way that 3D-SPECK encodes an $\mathcal{S}$ set is similar to 3D-SPIHT. 3D-SPECK follows closely the methodology used in the 3D-SPIHT algorithm. The difference lies in the sorting pass. 3D-SPIHT uses spatial orientation trees for significance testing, whereas 3D-SPECK uses sets of type $\mathcal{S}$ as defined above. In other words, the difference is in the partitioning. The way 3D-SPECK doing the partitioning can exploit the clustering of energy found in transformed images and code first those areas of the set with high energy. The coefficients with large information content are therefore can be coded first.

3D-SPECK maintains two linked lists:

- **LIS** – List of Insignificant Sets. This list contains $\mathcal{S}$ sets of varying sizes which have not yet been found significant against a threshold.

- **LSP** – List of Significant Pixels. This list contains pixels that have been found significant against a certain threshold n.

3D-SPECK proceeds sets $\mathcal{S}$ in the order of increase of sizes. Single pixel sets are to be tested first, and sets with larger sizes are to be tested later. This kind of ordering is the functional equivalent of separating the LIS into two lists, an LIP and an LIS, as done in SPIHT.

2.2. The Algorithm

Although the subband transform structure is inherently scalable of resolution, most embedded coders in the literature are unable to efficiently provide resolution scalable code streams. This is a consequence of entanglement in coding, modeling, and data structure across different resolutions. As an example, EZW and SPIHT, the classical zerotree coders with individual zerotrees spanning several subband scales are not efficient for resolution scalable coding.

To obtain resolution progressive bitstream, the bits in blocks belonging to subbands of coarser scales are encoded before those of finer scales.

The original 3D-SPECK supports SNR scalability, it generates SNR embedded bitstream for the whole image sequence.[10] We can modify our implementation of SNR scalable 3D-SPECK quite easily to enable resolution scalability. The idea is just to run the 3D-SPECK algorithm on each subband separately. Instead of maintaining the same significance threshold across subbands until we exhaust them, we maintain separate LIS and LSP lists for each subband and proceed through the lower threshold in every subband before moving to the next one. Therefore, we generate an array of lists, LSP_k and LIS_k, where k is the subband index. We move through the subbands in the same order as before, from lowest to highest scale. Therefore, we can truncate the bitstream corresponding to a reduced scale and decode to that scale.

Similar to SNR progressive 3D-SPECK, resolution progressive 3D-SPECK also starts by adding all sets $\mathcal{S}$ to the LIS_k. Note each set $\mathcal{S}$ at the begining belongs to one LIS_k, where $k = 1, 2, 3, ..., \text{K}$, and K is the total number of subbands.

for $k = 1, 2, 3, ..., \text{K}$

1. **Initialization**
   - Output $n = \lfloor \log_2(\max \mid c_{i,j,k} \mid) \rfloor$
   - Set $\text{LSP}_k = \emptyset$, $k = 1, 2, 3, ..., \text{K}$
   - Set $\text{LIS}_k = \{$corresponding subband of transformed images of wavelet coefficients $\}$
2. **Sorting Pass**

   In increasing order of size of sets, for each set $\mathcal{S} \in \text{LIS}_k$, $\mathsf{ProcessS}(\mathcal{S})$

   $\mathsf{ProcessS}(\mathcal{S})$

   {
   - Output $\Gamma_n(\mathcal{S})$ (Whether the set is significant respect to current n or not)
   - if $\Gamma_n(\mathcal{S}) = 1$
     - if $\mathcal{S}$ is a pixel, output sign of $\mathcal{S}$ and add $\mathcal{S}$ to LSP_k
     - else $\mathsf{CodeS}(\mathcal{S})$
     - if $\mathcal{S} \in \text{LIS}_k$, remove $\mathcal{S}$ from LIS_k

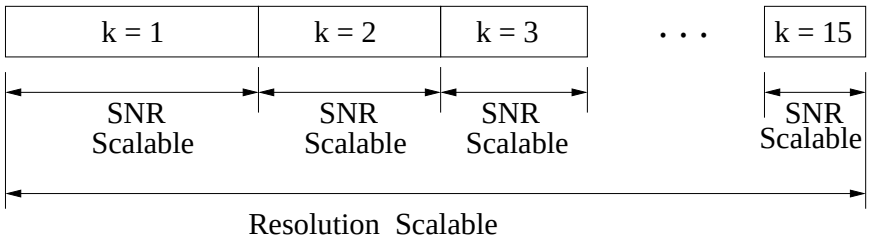

Figure 2. An example of resolution progressive 3D-SPECK

}

CodeS($\mathcal{S}$)

{

- Partition $\mathcal{S}$ into eight approximately equal subsets $\mathcal{O}(\mathcal{S})$.
- For each $\mathcal{O}(\mathcal{S})$
  - Output $\Gamma_n(\mathcal{O}(\mathcal{S}))$
  - if $\Gamma_n(\mathcal{O}(\mathcal{S})) = 1$
    - if $\mathcal{O}(\mathcal{S})$ is a pixel, output sign of $\mathcal{O}(\mathcal{S})$ and add $\mathcal{O}(\mathcal{S})$ to LSP_k
    - else CodeS($\mathcal{O}(\mathcal{S})$)
  - else
    - add $\mathcal{O}(\mathcal{S})$ to LIS_k

}

3. **Refinement Pass**

   For each entry $(i, j, k) \in \text{LSP}_k$, except those included in the last sorting pass, output the n^{th} MSB of $\mid c_{i,j,k} \mid$.

4. **Quantization Step**

   Decrement n by 1 and go to step 2.

For total K subbands, there are K LSP_k and K LIS_k at initialization. The algorithm encodes each subband and generates embedded bit stream independently. As shown in Figure 2, for two level dyadic decomposition (K = 15), resolution progressive 3D-SPECK generates SNR progressive bit stream for each subband, and overall, the whole bit stream is resolution scalable.

In order to effect multi-resolution decoding, bit stream boundaries are maintained between subbands. Adaptive arithmetic coding models are accumulated from samples within the same resolution scale. Finally, the modeling contexts do not include any neighbor from the finer scales. These conditions guarantee the decodability of the truncated code stream.

Although the full bit stream is resolution progressive, it is now not SNR progressive. The bit streams in the subbands are individually SNR progressive, so the bits belonging to the same bit planes in different subbands could be interleaved at the decoder after truncation to the desired scale to produce an SNR scalable bit stream.

For the SNR progressive coding mode, the bits are allocated optimally across subbands, according to the significance threshold of the coefficients. But, for the resolution progressive mode, for a given target bit rate,

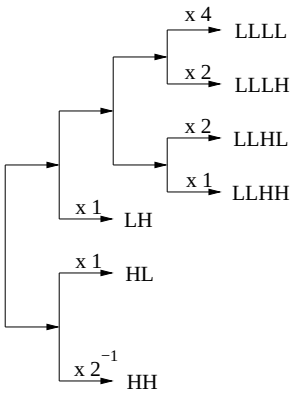

Figure 3. 1D wavelet packet structure along the spectral axis that makes the transform approximately unitary by shifting of the integer wavelet coefficients.

we need now to apply an explicit bit allocation algorithm to assign different bit rates to different subbands to minimize mean squared error. The procedure[1] we adopt to solve the rate allocation is a variance based algorithm.

To decode the image sequence to a particular level at a given rate, we need to encode each subband at a higher rate so that the algorithm can truncate the sub-bitstream to the assigned rate. However, unlike the JPEG2000 method, with this method we can achieve the target rate at the encoding stage without over-coding.

2.3. Scaling Wavelet Coefficients by Bit Shifts

We use integer filter in this paper. However, the integer filter transform with dyadic decomposition structure is not unitary. This does not affect the performance of lossless compression. However, to achieve good lossy coding performance, it is important to have an unitary transform. If the transform is not unitary, the quantization error in the wavelet domain is, thus, not equal to the mean squared error (MSE) in the time domain. Therefore, the lossy coding performance will be compromised. Appropriate transform structure and scaling of the integer wavelet coefficients can make the transform approximately unitary before quantization. It is therefore possible to keep track of the final quantizer coding error with the integer transform.

We adopt the transform structure mentioned by Xiong *et al.*.[13] As shown in Figure 3, a 4-level 1D wavelet packet tree structure is applied on the spectral axis. The scaling factors for each subband is indicated in the figure. As each scaling factor is some power of two, we can implement the scaling factor by bit shifting.

For the spatial axes, we keep the same 2D dyadic wavelet transform to each slice. As shown in Figure 4, 4-level dyadic decomposition structure with scaling factor for each subband is plotted. Each of the scaling factors is some power of two and therefore can be implemented by bit shifting.

To summarize, the 3D integer wavelet packet transform we use here is first to apply 1D packet decomposition and bit shifting along the spectral axis, followed by the basic 2D dyadic decomposition and bit shifting on the spatial axes. An example is shown in Figure 5, where scaling factors associated with some subbands are indicated. The factors are the multiplications of the corresponding scaling factors in Fig. 3 and Fig. 4. This 3D integer wavelet packet structure makes the transform approximately unitary, and thus leads to much better lossy coding performance.

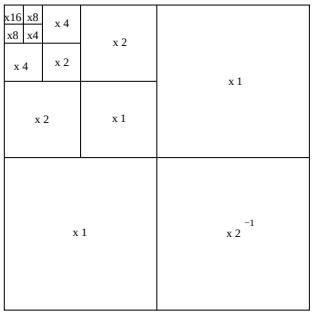

Figure 4. Same 2D spatial dyadic wavelet transform for each slice.

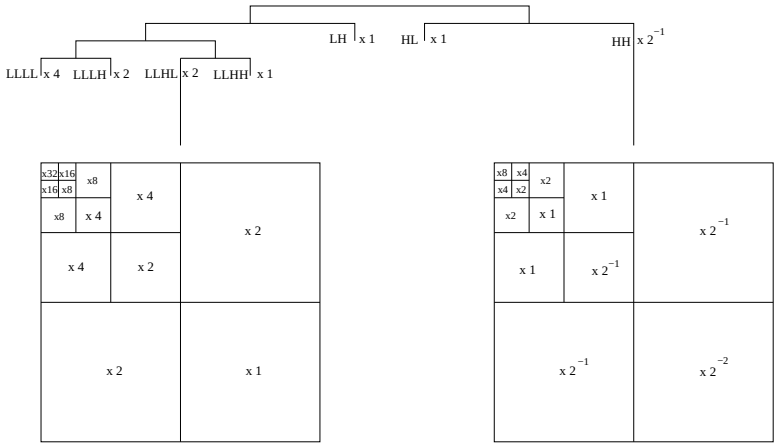

Figure 5. 3D integer wavelet packet transform.

2.4. Random ROI Access

This section describes how to apply resolution scalable 3D-SPECK to generate ROI retrievable bitstream. Figure 1 demonstrates the dyadic structure of the subband after wavelet decomposition. Figure 6 then gives an example of the parent-offspring dependencies in the 3D orientation tree after two-level dyadic decomposition. All the coefficients can be organized by trees with roots located at the lowest subband. Except the coefficient on the upper-left corner and the coefficients at the highest and lowest pyramid levels, the parent-offspring relationships for the coordinate (i, j, k) is

$$\begin{aligned} \mathcal{O}(i,j,k) \quad = \quad & (2i,2j,2k),(2i,2j,2k+1),(2i,2j+1,2k),(2i+1,2j,2k), \\ & (2i,2j+1,2k+1),(2i+1,2j,2k+1),(2i+1,2j+1,2k), \\ & (2i+1,2j+1,2k+1). \end{aligned} \tag{3}$$

A 3D region can be reconstructed by the coefficients of a tree independently. For an image sequence of size $I \times J \times K$ decomposed to level L, the number of trees, denoted as N, equals to the number of coefficients in the lowest frequency subband. That is $N = \frac{I}{2^L} \times \frac{J}{2^L} \times \frac{K}{2^L}$.

For instance, if the hyperspectral image sequence has 224 bands of 512 $\times$ 512 images. For $L = 4$ level decomposition, there are 14366 trees, with each tree representing a $32 \times 32 \times 14$ 3D region.

We use the coordinate of the root coefficient to represent a tree. For instance, the tree shown in Figure 6

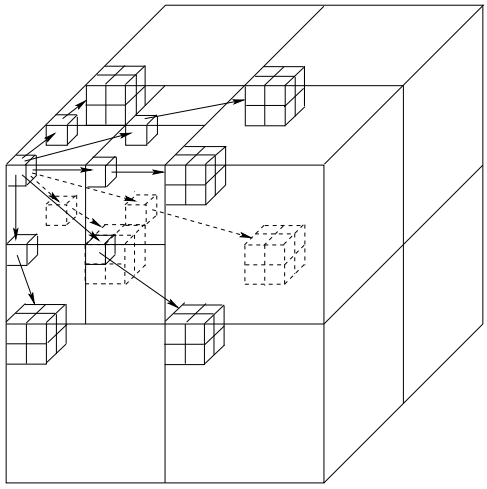

Figure 6. Parent-offspring dependencies in the 3D orientation tree.

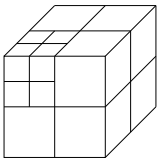

Figure 7. The 3D orientation tree shown in Figure 6.

is the tree with root coordinate located at (0, 0, 0), and this tree can be used to reconstruct the region from (0, 0, 0) to (31, 31, 13).

Similar to the idea presented by Xie,[12] to generate ROI retrievable bitstreams, we can apply resolution scalable 3D-SPECK on each tree independently. Figure 7 shows all the coefficients of the tree shown in Figure 6. We can see that the tree has the similar pyramid structure as that of the entire wavelet coefficients of the sequence. Hence, it is straightforward to apply the resolution scalable 3D-SPECK on each tree. The coefficients in each tree can then be put together to construct a block of the same size as its corresponding ROI. Therefore, we can randomly access any 3D regions in the sequence.

3. NUMERICAL RESULTS

We apply S+P (B) integer filter for resolution scalable 3D-SPECK. Dyadic wavelet transform is applied on the spatial domain, and wavelet packet structure with appropriate scaling factors is used for the spectrum (the third dimension) domain to make the transform approximate unitary.

We perform coding experiments on a signed 16-bit reflectance AVIRIS image volume. AVIRIS has 224 bands and 614 $\times$ 512 pixel resolution that corresponds to an area of approximately 11 km $\times$ 10 km on the ground. We have the 1997 version of Jasper Ridge scene 1[6] . For our experiments, we cropped the scene to 512 $\times$ 512 $\times$ 224 pixels.

To quantify fidelity, the coding performances are reported using rate-distortion results, using root mean square error (RMSE) calculated over the whole sequence as the distortion measure. Note that the 16-bit value range is -32,768 to 32,767. For comparison, we also provide results of the original SNR scalable 3D-SPECK at the partial and full scales.

Bit Rate (Full) (bpppb)	RMSE			
	1/8	1/4	1/2	Full
Resolution scalable				
0.1	7.3	10.5	22.6	65.3
0.5	6.9	10.1	18.1	23.3
1.0	4.1	7.8	9.7	11.2
2.0	2.7	3.8	4.3	5.0
SNR scalable				
0.1	7.7	11.3	23.2	64.2
0.5	7.4	10.5	18.6	22.9
1.0	4.4	8.3	10.2	10.8
2.0	2.8	4.0	4.4	4.9

Table 1. RMS error at a variety of resolution and coding bit rates using integer filter resolution scalable and SNR scalable 3D-SPECK.

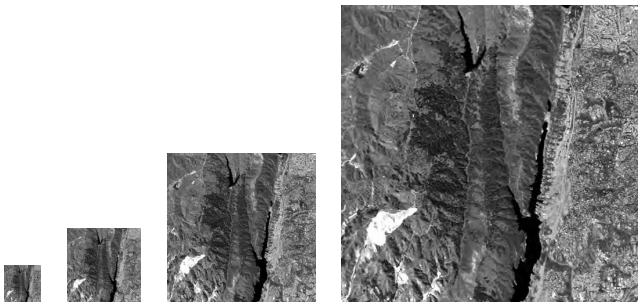

Figure 8. A visual example of resolution progressive 3D-SPECK. From left to right: 1/8, 1/4, 1/2, and full resolution at 1.0 bpppb.

Integer filter implementation supports lossy-to-lossless coding, and thus lossy and lossless reconstruction can be generated from the same embedded bitstream.

The RMSE values for a variety of bit rates in bits per pixel per band (bpppb) for the sequence are listed in Table 1 by comparing to the results of SNR scalable 3D-SPECK. The RMSE values listed in Table 1 for low resolution image sequences are calculated with respect to the lossless reconstructions of the corresponding resolutions. Since our bit allocation algorithm is not optimal, SNR scalable version performs slightly better at the full scale. To get reduced resolution from SNR scalable version, the full bitstream is decoded, but inverse DWT is performed only to partial scales. As there is significant high frequency content in hyperspectral images, the magnitudes of the coefficients are not close to monotone decreasing from coarse to fine scale. Instead, many coefficients that are on the highest bit planes are located in the higher subbands. To a partial scale, more bit budget is assigned to resolution scalable 3D-SPECK than assigned to SNR scalable version. Therefore resolution scalable 3D-SPECK yields lower RMSE values at partial scales.

Figure 8 demonstrates the reconstructed band 20, one band from the reconstructed sequence, decoded from a single resolution scalable bitstream at 1.0 bpppb to a variety of resolutions. Even a low resolution, we can get very high quality images. When the reconstructed sequences are presented at same display resolution, the perceived distortion for viewing a sample image at half resolution is equivalent to that at full resolution but from twice a distance.

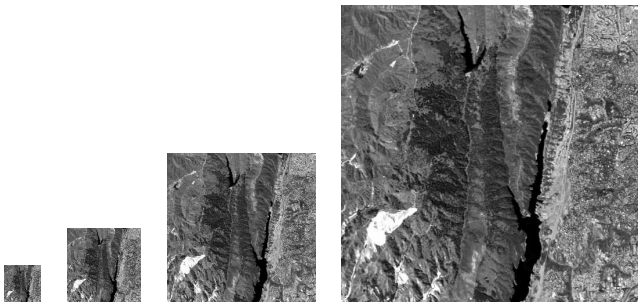

Figure 9. A visual example of lossless resolution progressive 3D-SPECK. From left to right: 1/8, 1/4, 1/2, and full resolution (original).

Bit Rate	**Bit budget (accumulated Kbytes)**			
(bpppb)	**1/8**	**1/4**	**1/2**	**Full**
0.1	61	274	734	734
0.5	68	290	963	3670
1.0	77	357	1505	7340
2.0	91	471	2422	14680

Table 2. Corresponding bit budgets for resolution scalable 3D-SPECK results for Table 1

The lossless decoding of the same band is demonstrated in Figure 9 at different resolutions.

The corresponding byte budgets for the individual resolutions of the resolution scalable 3D-SPECK for Table 1 are provided in Table 2. We can see that the computational cost of decoding reduces from one resolution level to the next lower one. The total bit cost decreases rapidly with successive reductions in resolution. However, for SNR scalable 3D-SPECK, the decoder needs to visit the whole full bitstream in order to decode to a certain resolution level.

Applying resolution scalable 3D-SPECK on each 3D orientation tree independently, we can generate an ROI retrievable bitstream. Figure 10 includes both 3D and 2D visual examples of ROI decoding. The original image sequence is lossless, and the ROI is also lossless. As shown in the third image of Figure 10, the brighter region is the ROI, and it corresponds to the original sequence from (250, 250, 0) to (410, 410, 223). In order to decode this region, we only need to decode the trees with root coordinates from (15, 15, 0) to (26, 26, 13). Note that to decode the ROI losslessly, we need to select trees to cover a slightly larger region to make sure the reconstruction is lossless even around the border of that ROI.

4. CONCLUSION

An embedded, block based, image wavelet transform coding algorithm of low complexity has been proposed. The algorithm has excellent performance for hyperspectral image compression. It supports resolution scalable coding and random ROI access.

REFERENCES

1. M. Balakrishnan and W.A. Pearlman, *Hexagonal subband image coding with perceptual weighting*, Optical Engineering, Vol. 32, No. 7, pp.1430-1437, July, 1993.

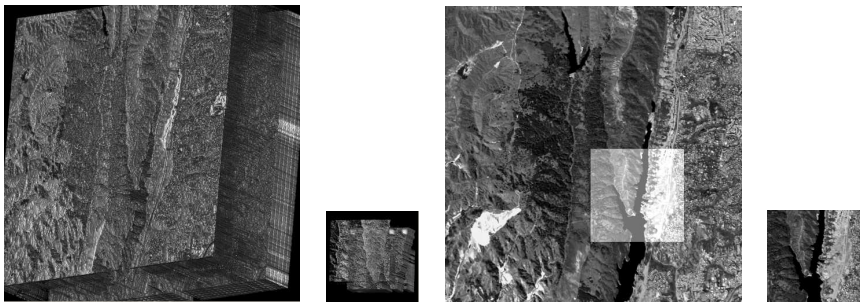

Figure 10. A visual example of ROI decoding from resolution scalable 3D-SPECK bit stream. Left two is a 3D demonstration, and the right two a 2D demonstration. From left to right: the original lossless image sequence; the ROI decoded image sequence; one band of the original sequence; and the same band of the previous 3D ROI.

2. J.C. Harsanyi and C.I. Chang, "Hyperspectral image classification and dimensionality reduction: an orthogonal subspace projection approach", IEEE Trans. Geoscience and Remote Sensing. Vol. 32, No. 4, July 1994.
3. A. Islam and W.A. Pearlman, *An embedded and efficient low-complexity hierarchical image coder*, in Proc. SPIE Visual Comm. and Image Processing, Vol. 3653, pp.294-305, 1999.
4. B. Kim and W.A. Pearlman, *An embedded wavelet video coder using three-dimensional set partitioning in hierarchical tree*, IEEE Data Compression Conference, pp.251-260, March 1997.
5. G. Motta, F. Rizzo, and J.A. Storer, "Compression of hyperspectral imagery," Data Compression Conference, Proceedings, DCC 2003, pp.25-27, March 2003.
6. http://makalu.jpl.nasa.gov.
7. A.N. Netravali and B.G. Haskell, *Digital pictures, representation and compression*, in Image Processing, Proc. of Data Compression Conference, pp.252-260, 1997.
8. M.J. Ryan and J.F. Arnold, "The lossless compression of AVIRIS images by vector quantization," IEEE Trans. Geoscience and Remote Sensing, Vol. 35, No. 3, May 1997.
9. A. Said and W.A. Pearlman, *An image multiresolution representation for lossless and lossy compression*, IEEE Trans. Image Process. 5, pp.1303-1310, 1996.
10. X. Tang, W.A. Pearlman and J.W. Modestino, *Hyperspectral image compression using three-dimensional wavelet coding*, SPIE/IS&T Electronic Imaging 2003, Proceedings of SPIE, Vol. 5022, Jan. 2003..
11. D. Taubman, *High performance scalable image compression with EBCOT*, IEEE Trans. on Image Processing, Vol. 9, pp. 1158–1170, July 2000.
12. G. Xie *A highly scalable image coder*, ICIP 2004, pp. 1297-1300, Sept. 2004.
13. Z. Xiong, X. Wu, S. Cheng, and J. Hua, *Lossy-to-lossless compression of medical volumetric data using three-dimensional integer wavelet transforms*, IEEE Trans. on Medical Imaging, Vol. 22, No. 3, March 2003.