Hyperspectral image compression using three-dimensional wavelet coding

Xiaoli Tang[a], William A. Pearlman[a] and James W. Modestino[b]

[a]ECSE Department, Rensselaer Polytechnic Institute, Troy, NY, USA 12180-3590
[b]ECE Department, University of Miami, Coral Gables, FL, USA 33146

ABSTRACT

A Hyperspectral image is a sequence of images generated by collecting contiguously spaced spectral bands of data. One can view such an image sequence as a three-dimensional array of intensity values (pixels) within a rectangular prism. We present a Three-Dimensional Set Partitioned Embedded bloCK (3DSPECK) algorithm based on the observation that hyperspectral images are contiguous in the spectrum axis (this implies large inter-band correlations) and there is no motion between bands. Therefore, the three-dimensional discrete wavelet transform can fully exploit the inter-band correlations. A SPECK partitioning algorithm extended to three-dimensions is used to sort significant pixels. Rate distortion (Peak Signal-to-Noise Ratio (PSNR) vs. bit rate) performances were plotted by comparing 3DSPECK against 3DSPIHT on several sets of hyperspectral images. Results show that 3DSPECK is comparable to 3DSPIHT in hyperspectral image compression. 3DSPECK can achieve compression ratios in the approximate range of 16 to 27 while providing very high quality reconstructed images. It guarantees over 3 dB PSNR improvement at all rates or rate saving at least a factor of 2 over 2D coding of separate spectral bands without axial transformation.

Keywords: SPIHT, Three dimensional image compression, Discrete Wavelet Transform (DWT), hyperspectral imaging, AVIRIS imaging

1. INTRODUCTION

Hyperspectral imaging is a powerful technique and has been widely used in a large number of applications, 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 surveilance. Hyperspectral images are generated by collecting hundreds of narrow and contiguously spaced spectral bands of data such that a complete reflectance spectrum can be obtained for 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), 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.

Many promising image compression algorithms based on wavelet transform (WT)[6] were proposed recently. They are simple, efficient and have been widely used in many applications. One of the them is Shapiro's embedded zerotree wavelet (EZW).[10] Said and Pearlman[8] refined and extended EZW subsequently to SPIHT. Islam and Pearlman[4] proposed another low complexity image encoder – Set Partitioned Embedded bloCK (SPECK). SPECK requires low dynamic memory. Related in various degrees to these earlier works on scalable image compression, the EBCOT[11] algorithm also uses a wavelet transform to generate the subband samples which are to be quantized and coded. EBCOT stands for Embedded Block Coding with Optimized Truncation, which identifies some of the major contributions of the algorithm. It is resolution and SNR scalable and with

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.

a random access property. EBCOT partitions each subband into relatively small blocks (typically, 32×32 or 64×64 pixel square), and generates a separate highly scalable (or embedded) bit stream for each block. The bit streams can be independently truncated to a given set of rates, which results in finer embedding.

The state-of-the-art encoder, SPIHT, has many attractive properties. It is an efficient embedded technique. The original SPIHT was proposed for 2-dimensional image compression, and it has been extended to 3D applications by Kim and Pearlman.[5] 3D-SPIHT is the modern-day benchmark for three dimensional image compression. It is a powerful tool to compress image sequences. It has been applied on multispectral image compression by Dragotti *et al.*.[2] They use vector quantization (VQ) and Karhunen-Loeve transform (KLT) on the spectral dimension to explore the correlation between multispectral bands. In the spatial domain, they use discrete wavelet transform, and the 3DSPIHT sorting algorithm is applied on the transformed coefficients. Dragotti *et al.*'s algorithms have promising performance. This is because VQ is theoretically the optimal block coding strategy and KLT is theoretically the optimal transform to decorrelate the data. However, VQ and KLT are computationally complex. One needs to either design code book or update statistics on time. Applying VQ and KLT on the spectral dimension of multispectral images is acceptable because multispectral images only have a small number of bands. However, on the contiguous spectrum applications such as hyperspectral imagery, it is not feasible for real-time computation. Fry[3] adopts 3DSPIHT directly on hyperspectral image compression. His work demonstrates the advantages of speed and computational complexity of 3DSPIHT. Results show that 3D-DWT is a fast and efficient means to exploit the correlations between hyperspectral bands.

The EBCOT algorithm has also been extended to 3D applications. Three Dimensional Cube Splitting EBCOT (3D CS-EBCOT)[9] initially partitions the wavelet coefficient prism into equally sized small code cubes of $64 \times 64 \times 64$ elements. The cube splitting technique is applied on each code cube to generate separate scalable bit streams. Like EBCOT, the bit streams may be independently truncated to any of a collection of different lengths to optimize the rate distortion criteria. Xu *et al.*[14] used a different method to extend EBCOT on video coding – Three-Dimensional Embedded Subband Coding with Optimized Truncation (3-D ESCOT). They treat each subband as a code cube and generate embedded bit streams for each code cube independently by using fractional bit-plane coding. Candidate truncation points are formed at the end of each fractional bit-plane. Again, the bit streams can be truncated independently into a layered stream by optimizing a rate distortion function.

In this paper, we extend 2D SPECK to 3D sources such as hyperspectral images. We call this new 3D image encoding technique Three-Dimensional Set Partitioned Embedded bloCK (3DSPECK). For an image sequence, three dimensional discrete wavelet transform (3D-DWT) is applied to obtain a wavelet coefficient prism. Since our applications are hyperspectral images, there is not motion, but tight statistical dependency along the wavelength axis of this prism. Therefore, the 3D-DWT can exploit the consequent correlation along the wavelength axis, as well as along the spatial axes. To start the algorithm, the wavelet coefficient prism is partitioned into small (three-dimensional) code blocks* with different sizes, and each subband is treated as a code block. Next, an extended and modified version of the SPECK sorting algorithm is applied to these code blocks to sort the significance of pixels. A block splitting algorithm similar to the cube splitting algorithm of 3D CS-EBCOT is used on the individual code blocks to test their significance. If a code block contains significant coefficients, it is split into several smaller sub-blocks. The descendant "significant" blocks are then further split until the significant coefficients are isolated. This block splitting algorithm can zoom in quickly to areas of high energy and code them first and therefore can exploit the presence of significant high frequency intra-band components. 3DSPECK exploits detailed underlying physical modeling properties of hyperspectral images.

This paper is organized as following: We will describe the proposed algorithm in Section 2. Section 3 presents experimental results, and Section 4 concludes the paper.

*It is customary to call the unit to be coded a block. This terminology is adopted here for the three-dimensional blocks in the shape of rectangular prisms.

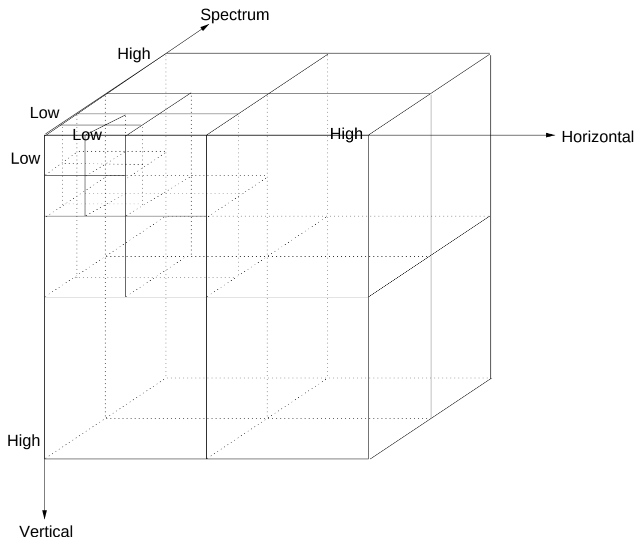


Figure 1. Structure for 3DSPECK. The numbers on the front lower left corners for each subband are marked to indicate the sorting order.

2. THREE-DIMENSIONAL SET PARTITIONED EMBEDDED BLOCK (3DSPECK)

Many researchers already extended 2-D SPIHT or SPIHT-like techniques to three-dimensional sources[5213].[9] To do this, people usually take a 3-D wavelet transform on the image sequence, and apply the extended sorting and partitioning algorithm to encode the source. As an example, 3-D SPIHT sorts coefficients along the paths of 3-D trees (one pixel corresponds to eight direct descendant pixels) instead of 2-D trees (one pixel to four direct descendants). The inter-band dependence or correlation can be exploited automatically, as was done in the spatial domain.

Following the similar idea, we extend and modify 2-D SPECK for three dimensional hyperspectral image applications. The first step is to decompose images into wavelet coefficients. Discrete wavelet transform is first applied on the spectral dimension, followed by on the horizontal (rows) and vertical (columns) axis. The resulting coefficients have a pyramid structure, and the structure has three levels of decomposition. Figure 1 illustrates this structure.

The next step is to design a sorting algorithm to encode wavelet coefficients based on their magnitudes. The motivation comes from the underlying physics of hyperspectral images. Most hyperspectral images have more high frequency content than other images in spatial domain. We illustrate this by comparing the spatial energy distribution of a hyperspectral image "coast" and an non-hyperspectral image "chess" (Figure 2) for one level of wavelet decomposition.

The energy is defined as the sum of squared value of wavelet coefficients. We calculate the energy proportion for each subband. The energy proportion is calculated by dividing the energy in one sub-band over the total energy. Our results show that the energy proportions on LL subband are 76.47% and 84.73% for "coast" and "chess" images, respectively. This shows that the "coast" image has more energy distributed over the HH, HL and LH sub-bands than that of "chess" image. In other words, hyperspectral images tend to have more large (significant) coefficients on finer spatial sub-bands by comparing to non-hyperspectral images. On the temporal (wavelength) axis, it is also true that less energy tends to locate on the lowest subband of the hyperspectral image sequences than that of other image sequences.

We want our sorting algorithm to be designed according to these observations. 3DSPIHT searches significant coefficients from the root of the tree structure down to its leaves. One 3D orientation tree includes a pixel on the root and its descendants. If all descendants are found to be insignificant against a certain

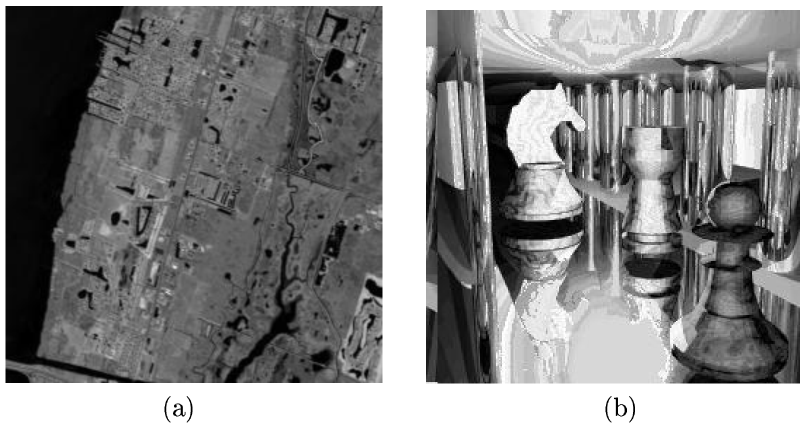


Figure 2. (a) hyperspectral image "coast"; (b) non-hyperspectral image "chess".

threshold, 3DSPIHT needs to send only one bit to indicate this information. This kind of tree structure performs excellently if energy concentrates near the root of the tree. Like EZW and SPIHT, they all take advantage of the tree structure to design their algorithms, and they are all proved to be very efficient in general cases. However, in our case of hyperspectral images, energy is not so concentrated near the root of the tree. With higher probabilities than the case of regular image sequences, the descendants of root pixels may be found to be significant during the first several passes. 3DSPIHT will send extra bits to represent the sorting path to locate a significant pixel. In order to locate the large coefficients in the high frequency bands quickly, we extend the quadri-section partitioning algorithm of 2DSPECK to octo-section partitioning in our 3D application. The wavelet coefficient prism is partitioned into code blocks of different sizes. Each subband is treated as a code block. Then, the block splitting into eight is applied on each code block following a certain order to sort significant pixels. This splitting of code blocks allows zooming in quickly to areas of high energy and coding them first. This can be proved by comparing the rate distortion curves of 2DSPECK and 2DSPIHT. The results in Islam *et al.*'s paper[4] show that 2DSPECK out-performs 2DSPIHT for images with much high frequency content. It is expected that the 3DSPECK would have the same property. 2DSPECK maintains an array of lists to process code units of varying sizes. This is simplified by maintaining only one list in 3DSPECK. 3DSPECK does not utilize the octave band partitioning in 2DSPECK. Details of the implementation are presented in the following. Later, the experimental results will show the improvements of our algorithm.

We follow all notations developed in Islam *et al.*'s paper[4] when we use the structure or terminology of SPECK.

For each small (3D) block in Figure 1, we call it type $\mathcal{S}$ set. These include 8 sub-bands under the coarsest level low-low-low (LLL) sub-band and the 7 sub-bands at the next finer level and the remaining 7 sub-bands at the finest level (e.g. HLL, LHH). All together, there are 22 type $\mathcal{S}$ sets. We are going to use these sets when we initialize the algorithm.

3DSPECK maintains two linked lists:

- **LIS** – List of Insignificant Sets. This list contains sets of type $\mathcal{S}$ of varying sizes.
- **LSP** – List of Significant Pixels. This list contains pixels that have been found significant against threshold n.

We say that the 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 write the significance of a set $\mathcal{T}$ as a function of n and the set $\mathcal{T}$:

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

The size of a set is the number of elements in the set.

Below we present the new encoding algorithm:

1. **Initialization**
   - Output $n = \lfloor \log_2(\max \mid c_{i,j,k} \mid) \rfloor$
   - Set LSP $= \emptyset$
   - Set LIS $= \{$all small blocks in Figure 1 (total there are 22 sets) $\}$
2. **Sorting Pass**

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

```
ProcessS($\mathcal{S}$)
{
    • Output $S_n(\mathcal{S})$ (Whether the set is significant respect to current $n$ or not)
    • if $S_n(\mathcal{S}) = 1$
        – if $\mathcal{S}$ is a pixel, output sign of $\mathcal{S}$ and add $\mathcal{S}$ to LSP
        – else CodeS($\mathcal{S}$)
        – if $\mathcal{S} \in$ LIS, remove $\mathcal{S}$ from LIS
}
CodeS($\mathcal{S}$)
{
    • Partition $\mathcal{S}$ into eight equal subsets $\mathcal{O}(\mathcal{S})$. For the situation that the original set size is odd × odd × odd, we can partition this kind of sets into different but approximate equal sizes of subsets (see Fig 3). For the situation that the size of the third dimension of the set is 1, we can partition the set into 4 approximately equal sizes of subsets.
    • For each $\mathcal{O}(\mathcal{S})$
        – Output $S_n(\mathcal{O}(\mathcal{S}))$
        – if $S_n(\mathcal{O}(\mathcal{S})) = 1$
            ∗ if $\mathcal{O}(\mathcal{S})$ is a pixel, output sign of $\mathcal{O}(\mathcal{S})$ and add $\mathcal{O}(\mathcal{S})$ to LSP
            ∗ else CodeS($\mathcal{O}(\mathcal{S})$)
        – else
            ∗ add $\mathcal{O}(\mathcal{S})$ to LIS
}
```

3. **Refinement Pass**

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

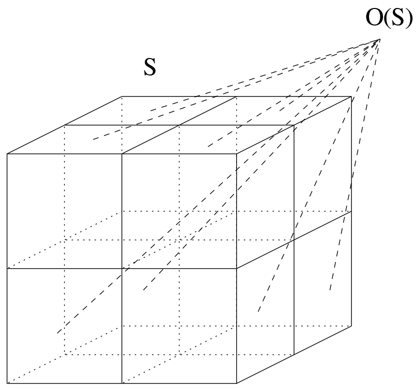


Figure 3. Partitioning of set $\mathcal{S}$

4. **Quantization Step**

   Decrement n by 1 and go to step 2.

After initialization, there are 22 sets in the LIS. We treat each sub-band on the pyramid as an initial set. All sets are put on the list in the order of extended "Z" scan. An example of the extended "Z" scan is illustrated in Figure 1, and the order of the sets is marked by numbers. We can see from Figure 1 that the list is a tree-like list. The underlying idea is based on the fact that though energy of hyperspectral images does not tend to concentrate in the LLL sub-band as much as in the case of non-hyperspectral images, a large portion of the energy still resides in the coarsest level sub-bands. In other words, the coarsest level sub-bands always convey the most important information of the original image sequence. If these bands are treated as individual sets, we know with extremely high probability that the algorithm will output 1 when testing the significance of these sets at the first iteration. Since the sizes of the coarsest level sub-bands are small, the 3DSPECK algorithm zooms into the significant coefficients quickly. On the other hand, the probability of finding significant coefficients in the sets during the early iterations decreases when going down the subbands of the pyramid. Our experiments show that sometimes no significant coefficient can be found against the first one or two bit planes on the finest sub-bands. Therefore, leaving those larger size sub-bands as individual sets can avoid using extra bit budget to present non-significance. Hence, when we test the significance of sets, we scan sets by following the path from the top of the pyramid down to its bottom. In this way, we can find significant coefficients quickly in the area of high energy and send the most important information first. This satisfies the requirement of embedded coding and progressive transmission.

During the sorting pass, we claim that sets should be tested in increasing order of size. This follows the argument in Islam *et al.*'s paper.[4] After the first pass, many sets of type $\mathcal{S}$ of varying sizes are generated and added to the LIS. The algorithm sends those sets to the LIS because it found some immediate neighbors of those sets are significant against some threshold n. Since coefficients in the same energy cluster have close magnitudes, those not found to be significant in the current pass are very likely to be found as significant against the next lower threshold. Furthermore, our experiments show that a large number of sets with size of 1 will be generated after the first iteration. Therefore, testing sets in increasing order of size can test pixel level first and locate new significant pixels immediately.

We do not use any sorting mechanism to process sets of type $\mathcal{S}$ in increasing order of their size. Even the fastest sorting algorithm will slow the coding procedure significantly. This is not desirable in fast implementation of coders. However, there are simple ways of completely avoiding this sorting procedure. 2DSPECK uses an array of lists[4] to avoid sorting, whereas we use a different and simpler method achieve this goal.

Note that the way sets $\mathcal{S}$ are constructed, they lie completely within a sub-band. Thus, every set $\mathcal{S}$ is located at a particular level of the pyramidal structure. Partitioning a set $\mathcal{S}$ into eight/four subsets is equivalent to going down the pyramid one level at the corresponding finer resolution. Hence, the size of a set $\mathcal{S}$ corresponds to a particular level of the pyramid. Therefore, we only need to search the same LIS several times at each iteration. Each time we only test sets with sizes corresponding to a particular level of the pyramid. This implementation completely eliminates the need for any sorting mechanism for processing the sets $\mathcal{S}$. Thus, we can test sets in increasing order of size while keeping our algorithm running fast.

The idea of ProcessS and CodeS is an extension of 2-D ProcessS and CodeS in SPECK. We extend from a 2-D to a 3-D structure. 3DSPECK processes a type $\mathcal{S}$ set by calling ProcessS to test its significance with respect to a threshold n. If this set is significant, ProcessS will call CodeS to partition set $\mathcal{S}$ into eight/four approximately equal subsets $\mathcal{O}(\mathcal{S})$ (Fig 3). We treat each of these subsets as new type $\mathcal{S}$ set, and in turn, test their significance. This process will be executed recursively until reaching pixel level where the significant pixel in the original set $\mathcal{S}$ is located. The algorithm then sends the significant pixel to the LSP and codes the significance.

After finishing the processing of all sets in the LIS, we execute the refinement pass. During the refinement pass, the algorithm outputs the n^{th} most significant bit of the absolute value of each entry (i, j, k) in the LSP, except those included in the just-completed sorting pass. The purpose of the refinement pass of 3DSPECK is the same as that of SPIHT and SPECK. The refinement pass refines significant pixels found during previous passes progressively.

The last step of this algorithm is decreasing n by 1 and returning to the sorting pass. This step makes the whole process run recursively.

The decoder is designed to have the same mechanism as the encoder. Based on the outcome of the significance tests in the received bit stream, the decoder can follow exactly the same execution path as the encoder and therefore reconstruct the image sequence progressively.

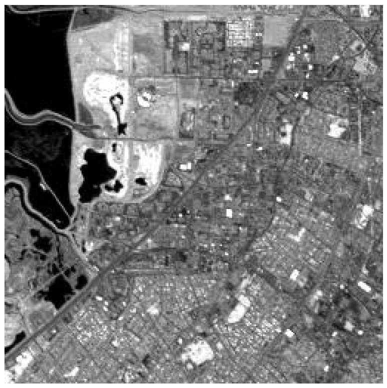
As with other coding methods such as SPIHT and SPECK, the efficiency of our algorithm can be improved by entropy-coding its output[7].[11] We use the adaptive arithmetic coding algorithm of Witten *et al.*[12] to code the significance map. Referring to the function CodeS in our algorithm, instead of coding the significance test results of the eight/four subsets separately, we code them together first before further processing the subsets. Simple context is applied for conditional coding of the significance test result of this subset group. More specifically, we encode the significance test result of the first subset without any context, but encode the significance test result of the second subset by using the context (whether the first subset is significant or not) of the first coded subset and so on. Other outputs from the encoder are entropy coded by applying simple arithmetic coding without any context.

Also, we make the same argument as 2DSPECK that if a set $\mathcal{S}$ is significant and its first seven/three subsets are insignificant, then this means that the last subset must be significant. Therefore, we do not need to test the significance of the last subset. This reduces the bit rate somewhat and provides corresponding gains.

3. NUMERICAL RESULTS

AVIRIS image sequences "coast", "moffett" and "jasper" will be tested for 3DSPECK. Among these three image sequences, "coast" and "moffett" are typical hyperspectral images with higher spatial frequency content, while the "jasper" sequence is selected to be very smooth in the spatial domain. Later experimental results will show the advantages of our algorithm when applying it on image sequences with much high frequency content.

The description of AVIRIS can be found in the introduction. Image "coast" is taken at a coastal region in Puerto Rico at Mayaguez. There are 204 channels (bands) in the range 400 to 2500 nm, each consisting of 397 lines of 268 pixels quantized to 8 bits per pixel (bpp). It is saved as BIS (band interleaved by sample) format. "moffett" is pictured above the Moffett Field area in CA. Its format is band interleaved by sample (channel, sample, line) with dimensions (224, 614, 512) and 8 bits per pixel. "jasper" portrays Jasper Ridge in CA. It has the same format and dimensions as "moffett". In all cases, a square region of 256 $\times$ 256 pixels which exhibits all key features present in the images is selected and used as a test set. Band 101 of "coast"


(a)

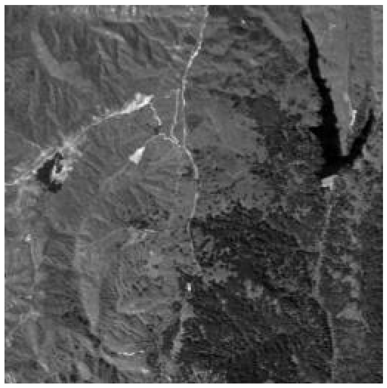

(b)

Figure 4. (a) Band 101 of "moffett" test image; (b) Band 116 of "jasper" test image.

image is shown in Figure 2 (a), band 101 of "moffett" and band 116 of "jasper" are shown in Figure 3 (a) and (b), respectively.

The 9/7 tap biorthogonal filters[1] will provide the transform for our implementation of the new algorithm. The coding unit size of 16 slices is used in our implementation. Coding 16 slices per time instead of coding the whole set of image sequence (i.e. all 204 channels for whole coast sequence) in one time saves considerable memory and coding delay. The final result is obtained by averaging over several 16-slice segments.

The coding performance of 3DSPECK is compared with that of 3D-SPIHT and 2D-SPIHT algorithms on the same sets of data, by means of the Peak Signal-to-Noise Ratio (PSNR):

$$\mathrm{PSNR} = 10\,\log_{10}\frac{(2^{\mathrm{nsb}}-1)^2}{\mathrm{MSE}}\ \mathrm{dB}, \tag{3}$$

where nsb denotes the maximum bit depth, and MSE is the mean squared-error between the original and reconstructed images.

Rate distortion curves are plotted in (a), (b) and (c) of Figure 3 for "coast", "moffett", and "jasper" sequences, respectively. The PSNR results for 2DSPIHT are obtained by first processing each band separately, then averaging all the results.

By comparing the rate distortion curves of 2DSPIHT to 3DSPIHT and 3DSPECK, we can see that 3D versions guarantee over 3 dB improvement at all rates. This implies that the hyperspectral image sequences are highly correlated, and taking 3D DWT can fully exploit these inter band correlations. Thus, 3D compression techniques achieve much better performance than the 2D version which encodes images separately.

Figure 3 (a) and (b) show that 3DSPECK performs excellently in the case when high spatial frequency content is present. If the bit rate is very low (i.e. less than 0.4 bpp), 3DSPECK out-performs the well known 3DSPIHT. As the bit rate increases, 3DSPECK achieves higher PSNR than 3DSPIHT for the "coast" image, but lower PSNR for "moffett".

When the image sequence is smooth and does not have much high frequency content, the traditional 3DSPIHT performs better than 3DSPECK. The reason is that if energy is highly concentrated on the highest pyramid level of wavelet coefficients, the sorting algorithm of 3DSPIHT will locate the important information very quickly and use the minimum number of bits to present the unimportance of the lower levels of the pyramid. However, 3DSPECK has to use extra bit budget to describe whether each set on the list is important. Hence, for this situation, 3DSPIHT will achieve better results. Again, 3DSPECK performs better compared to 3DSPIHT at low bit rate than at higher bit rate.

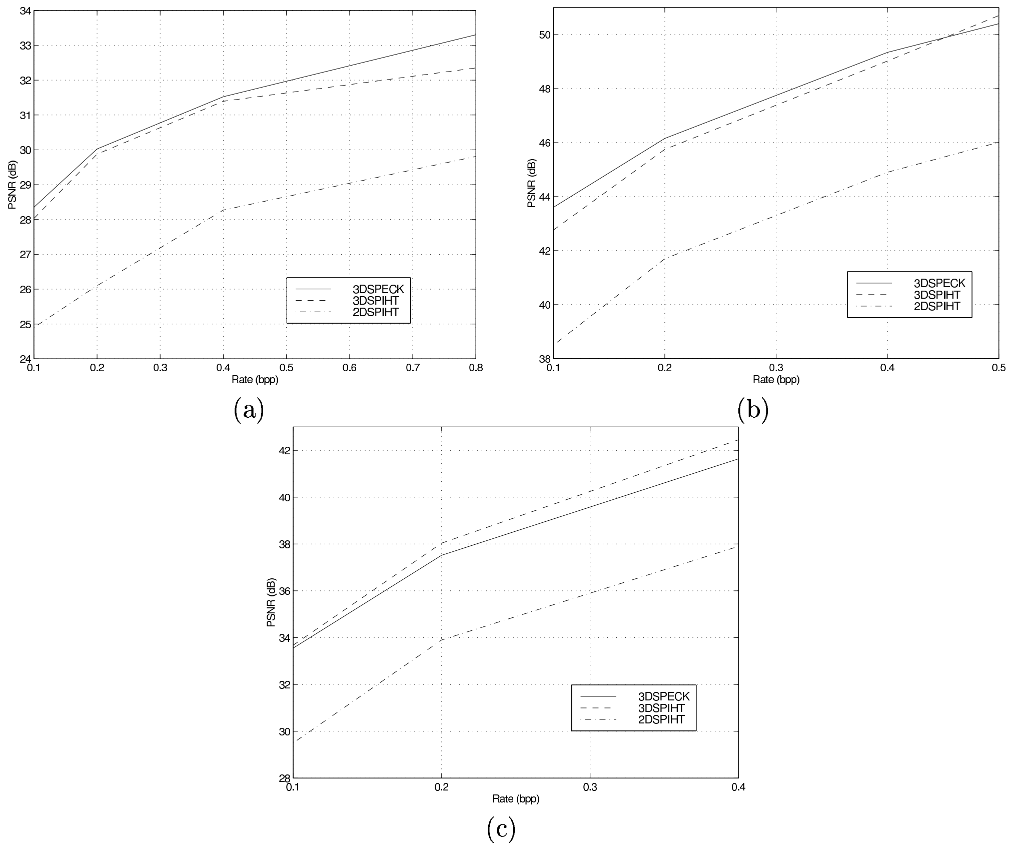


Figure 5. Comparative evaluation the rate distortions of 3DSPECK, 3DSPIHT and 2DSPIHT for the (a) "coast" image; (b) "moffett" image; and (c) "jasper" image.

The rate distortion results suggest that if the sources have much high frequency content, and the compression rate is required to be very high (bpp very low), 3DSPECK is an excellent candidate. Overall, it is comparable to 3DSPIHT.

Since our application is hyperspectral imagery, the visual effect is very important. Therefore, it is necessary to further assess the quality of the compressed images.

Images (a) (b) and (c) Figure 3 show the reconstructed "moffett" image at 0.1, 0.3 and 0.5 bpp, respectively. Many fine details are lost in image (a) and the borders between different regions are blurred. Such image quality can be used for quickly browsing image to check whether it is the sequence of interest. Since our coded sequences are embedded, higher quality images can be obtained by gradually decoding more of the available bits.

As the bit rate is increased, we can get finer and finer images. The noise presented in the original image can also be refined. In other words, air dust and other sources rendered noise during capturing images. However, lossy compression has the effect of removing such noise. This makes the compressed images "look" better than the original ones. Images (b) and (c) in Figure 3 illustrate high quality of compressed images even at very low bit rates.

We also conducted an experiment called "3D-DWT + 2DSPECK" to establish the effectiveness of extension of SPECK from 2D to 3D separate from the 3D transform. As what the name suggests, "3D-DWT + 2DSPECK" does 3D-DWT transform followed by 2DSPECK coding on separate frame (different rate is assigned to frames in different temporal subbands to minimize the overall rate distortion), therefore can assess

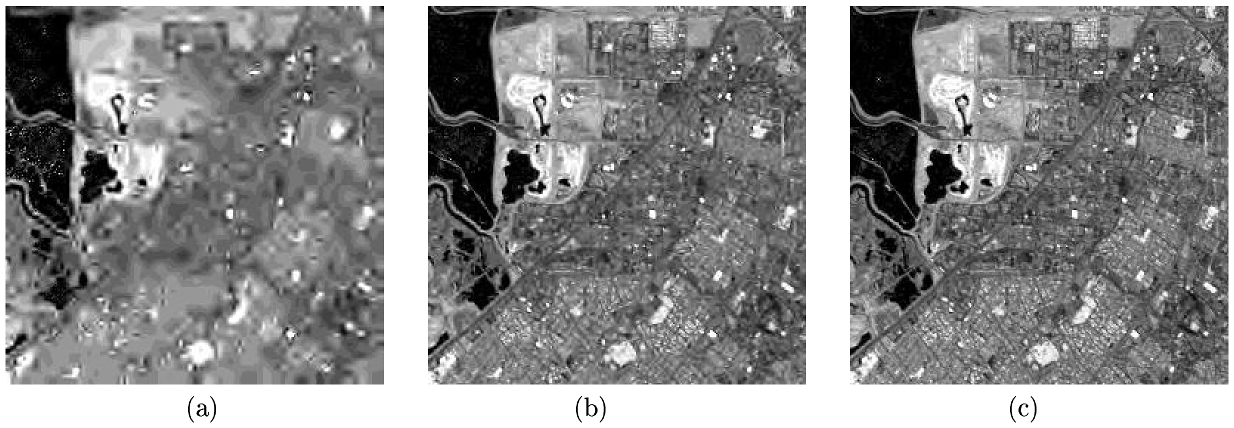


Figure 6. Results for "moffett" hyperspectral image at channel 116: (a) 0.1 bit/pixel (b) 0.3 bit/pixel (c) 0.5 bit/pixel.

separately gain from 3D transform and 3D coding. Our results show that for all cases, the rate distortion curves of "3D-DWT + 2DSPECK" fall between the curves of 3DSPECK and 2DSPECK. In other words, 3D-DWT contributes only part of the coding gain, while our 3D coding accounts for the rest.

4. CONCLUSION AND FUTURE WORK

This paper proposed a three dimensional set partitioned embedded block coder for hyperspectral image compression. The three dimensional wavelet transform automatically exploits inter-band dependence. The modified SPECK sorting algorithm can find the most important information efficiently and therefore provides a corresponding coding gain against two dimensional compression algorithm.

The proposed 3DSPECK is completely embedded and can be used for progressive transmission. These features make the proposed coder a good candidate to compress (encode) hyperspectral images before transmission and to decompress (decode) them at another end for image storage.

Compressed hyperspectral bit streams require protection from channel errors during transmission. For efficient end-to-end transport, a suitable forward-error-correcting (FEC) algorithm should be applied on the embedded bit streams before transmission. In the future, we will investigate this topic and implement appropriate channel coding algorithms.

REFERENCES

1. M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies, "Image coding using wavelet transform", in *IEEE Trans. Image Processing*, **vol. 1**, pp. 205–220, 1992.
2. P. L. Dragotti, G. Poggi, and A. R. P. Ragozini, "Compression of multispectral images by three-dimensional SPIHT algorithm", in *IEEE Trans. on Geoscience and remote sensing*, **vol. 38, No. 1**, pp. 416–428, Jan 2000.
3. Thomas W. Fry, *Hyperspectral image compression on reconfigurable platforms*, Master Thesis, Electrical Engineering, University of Washington, 2001.
4. 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.
5. B. Kim and W. A. Pearlman, "An embedded wavelet video coder using three-dimensional set partitioning in hierarchical tree", in *IEEE Data Compression Conference*, pp. 251–260, March 1997.
6. S. Mallat, "Multifrequency channel decompositions of images and wavelet models", in *IEEE Trans. Acoust., Speech, Signal Processing*, **vol. 37**, pp. 2091–2110, Dec 1989.

7. E. Ordentlich, M. Weinberger, and G. Seroussi, "A low-complexity modeling approach for embedded coding of wavelet coefficients", in *Proc. IEEE Data Compression Conf.*, pp. 408–417, Snowbird, UT, Mar 1998.
8. A. Said and W. A. Pearlman, "A new, fast and efficient image codec based on set partitioning in hierarchical trees", in *IEEE Trans. on Circuits and Systems for Video Technology*,**vol 6**, pp. 243–250, June 1996.
9. P. Schelkens, *Multi-dimensional wavelet coding algorithms and implementations*, Ph.D dissertation, Department of Electronics and Information Processing, Vrije Universiteit Brussel, Brussels, 2001.
10. J. M. Shapiro, "Embedded image coding using zerotrees of wavelet coefficients", in *IEEE Trans. Signal Processing*, **vol. 41**, pp. 3445–3462, Dec 1993.
11. D. Taubman, "High performance scalable image compression with EBCOT", in *IEEE Trans. on Image Processing*, **vol. 9**, pp. 1158–1170, July 2000.
12. I. H. Witten, R. M. Neal, and J. G. Cleary, "Arithmetic coding for data compression", in *Commun. ACM*, **vol. 30**, pp. 520–540, June 1987.
13. Z. Xiong, X. Wu, D. Y. Tun, and W. A. Pearlman, "Progressive coding of medical volumetric data using three-dimensional integer wavelet packet transform", in *Medical Technology Symposium, 1998. Proceedings*, pp. 384–387, Pacific, 1998.
14. J. Xu, Z. Xiong, S. Li, and Y. Zhang, "Three-dimensional embedded subband coding with optimized truncation (3-D ESCOT)", in *J. Applied and Computational Harmonic Analysis: Special Issue on Wavelet Applications in Engineering*, **vol. 10**, pp. 290–315, May 2001.