Three-Dimensional SPIHT Coding of Volume Images with Random Access and Resolution Scalability

CIPR Technical Report TR-2007-1

Emmanuel Christophe and William A. Pearlman

March 2007

Center for
Image Processing Research

Rensselaer Polytechnic Institute
Troy, New York 12180-3590
http://www.cipr.rpi.edu

Three-dimensional SPIHT Coding of Volume Images with Random Access and Resolution Scalability

Emmanuel Christophe[a], William A. Pearlman[b]

[a] *CNES - bpi 1219 - 18, av. E. Belin - 31401 Toulouse cedex 9, FRANCE*

[b] *Electrical, Computer, and Systems Engineering Department, Rensselaer Polytechnic Institute, Troy, NY 12180-3590 USA*

1 Introduction

Compression of 3D data volumes poses a challenge to the data compression community. Lossless or near lossless compression is often required for these 3D data, whether medical images or remote sensing hyperspectral images. Due to the huge amount of data involved, even the compressed images are significant in size. In this situation, progressive data encoding enables quick browsing of the image with limited computational or network resources.

For satellite sensors, the trend is toward increase in the spatial resolution, the radiometric precision and possibly the number of spectral bands, leading to a dramatic increase in the amount of bits generated by such sensors. Often, continuous acquisition of data is desired, which requires scan-based mode compression capabilities. Scan-based mode compression denotes the ability to begin the compression of the image when the end of the image is still under acquisition. When the sensor resolution is below one meter, images containing more than 30000×30000 pixels are not exceptional. In these cases, it is important to be able to decode only portions of the whole image. This feature is called random access decoding.

Resolution scalability is another feature which is appreciated within the remote sensing community. Resolution scalability enables the generation of a quicklook of the entire image using just few bits of coded data with very limited computation. It also allows the generation of low resolution images which

Email addresses: emmanuel.christophe@gmail.com (Emmanuel Christophe), pearlw@ecse.rpi.edu (William A. Pearlman).

can be used by applications that do not require fine resolution. More and more applications of remote sensing data are applied within a multiresolution framework [1, 2], often combining data from different sensors. Hyperspectral data should not be an exception to this trend. Hyperspectral data applications are still in their infancy and it is not easy to foresee what the new application requirements will be, but we can expect that these data will be combined with data from other sensors by automated algorithms. Strong transfer constraints are more and more present in real remote sensing applications as in the case of the *International Charter: space and major disasters* [3]. Resolution scalability is necessary to dramatically reduce the bitrate and provide only the necessary information for the application.

The SPIHT algorithm is a good candidate for on-board hyperspectral data compression. A modified version of SPIHT is currently flying towards the 67P/Churyumov-Gerasimenko comet and is targeted to reach in 2014 (Rosetta mission) among other examples. This modified version of SPIHT is used to compress the hyperspectral data of the VIRTIS instrument [4]. This interest is not restricted to hyperspectral data. The current development of the CCSDS (Consultative Committee for Space Data Systems, which gathers experts from different space agencies as NASA, ESA and CNES) is oriented towards zerotrees principles [5] because JPEG 2000 suffers from implementation difficulties as described in [6] (in the context of implementation compatible with space constraints).

Several papers develop the issue of adaptation from 2D coding to 3D coding using zerotree based methods. One example is adaptation to multispectral images in [7] through a Karhunen-Loeve Transform on the spectral dimension and another is to medical images where [8] uses an adaptation of the 3D SPIHT, first presented in [9]. In [10], a more efficient tree structure is defined and a similar structure proved to be nearly optimal in [11]. To increase the flexibility and the features available as specified in [12], modifications are required. Few papers focus on the resolution scalability, as is done in papers [9, 13–16], adapting SPIHT or SPECK algorithms. However none offers to differentiate the different directions along the coordinate axes to allow full spatial resolution with reduced spectral resolution. In [13] and [14], the authors report a resolution and quality scalable SPIHT, but without the random access capability to be enabled in our proposed algorithm. The problem of error resilience is developed in [17] on a block-based version of 3D-SPIHT. Adapting 3D-SPECK for region of interest (ROI) coding appears in [18] and one adaptation of SBHP for ROI coding is described in [19]. However, to the authors' knowledge, no paper presents the combination of all these features doing a rate distortion optimization between blocks and to maintain optimal rate-distortion performance and preserve the property of quality scalability.

This paper presents the adaptation of the well-known SPIHT algorithm [20] for 3D data enabling random access and resolution scalability or quality scalability. Compression performance is compared with JPEG 2000 [21].

2 Data decorrelation and tree structure

2.1 3D anisotropic wavelet transform

Hyperspectral images contain one image of the scene for different wavelengths, thus two dimensions of the 3D hyperspectral cube are spatial and the third one is spectral (in the physics sense). Medical magnetic resonance (MR) or computed tomography (CT) images contain one image for each slice of observation, in which case the three dimensions are spatial. However the resolution and statistical properties of the third direction are different. To avoid confusion, the first two dimensions are referred as spatial, whereas the third one is called spectral. An anisotropic 3D wavelet transform is applied to the data for the decorrelation. This decomposition consists of performing a classic dyadic 2D wavelet decomposition on each image plane followed by a 1D dyadic wavelet decomposition in the third direction. The obtained subband organization is represented on Figure 1. The decomposition is non-isotropic as not all subbands are regular cubes and some directions are privileged. It has been shown that this anisotropic decomposition is nearly optimal in a rate-distortion sense in terms of entropy [22] as well as real coding [11]. To the authors' knowledge, this is valid for 3D hyperspectral data as well as 3D magnetic resonance medical images and video sequences. This transform has been used in many papers about 3D image compression as [7, 9, 10, 17, 23]. Moreover, this is the only 3D wavelet transform supported by the JPEG 2000 standard in Part II [24].

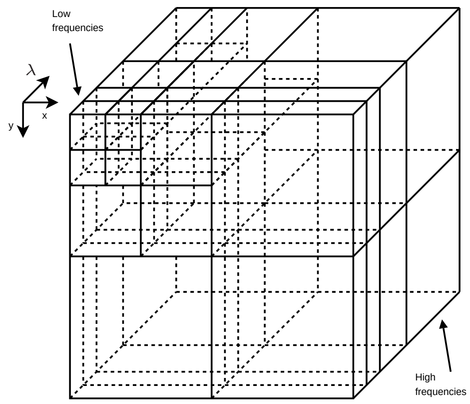


Fig. 1. Wavelet decomposition subbands. It is illustrated here with 3 decompositions levels for simplicity, 5 levels are used in practice.

The implementation of this particular wavelet transform is beyond the scope of this paper. The open source implementation QccPack [25] is used to perform the direct wavelet transform as well as the inverse transform. The 5/3 integer wavelet transform is available as well in the latest version.

2.2 Tree structure

The SPIHT algorithm [20] uses a tree structure to define a relationship between wavelet coefficients from different subbands. To adapt the SPIHT algorithm on the anisotropic decomposition, a suitable tree structure is defined. In [10], the relation between the coefficients from the lowest frequency subband and its descendants are defined in the manner of the first version of SPIHT [26]. We keep the latest version as defined in [20]. Let us define $\mathcal{O}_{spat}(i,j,k)$ as the spatial offspring of the pixel located at sample i, line j in plane k. The first coefficient in the upper front, left corner is noted as $(0,0,0)$. In the spatial direction, the relation is similar to the one defined in the original SPIHT. Except at the highest and lowest spatial frequency subbands, we have: $\mathcal{O}_{spat}(i,j,k) = \{(2i,2j,k),(2i+1,2j,k),(2i,2j+1,k),(2i+1,2j+1,k)\}$. In the highest spatial frequency subbands, there are no offspring: $\mathcal{O}_{spat}(i,j,k) = \emptyset$ and in the lowest frequency subband, coefficients are grouped in 2×2 as the original SPIHT. If we call n_s the number of samples per line and n_l the number of lines in the lowest frequency subband, we have:

- if i even and j even: $\mathcal{O}_{spat}(i,j,k) = \emptyset$
- if i odd and j even: $\mathcal{O}_{spat}(i,j,k) = \{(i+n_s,j,k),(i+n_s+1,j,k),(i+n_s,j+1,k),(i+n_s+1,j+1,k)\}$
- if i even and j odd: $\mathcal{O}_{spat}(i,j,k) = \{(i,j+n_l,k),(i+1,j+n_l,k),(i,j+n_l+1,k),(i+1,j+n_l+1,k)\}$
- if i odd and j odd: $\mathcal{O}_{spat}(i,j,k) = \{(i+n_s,j+n_l,k),(i+n_s+1,j+n_l,k),(i+n_s,j+n_l+1,k),(i+n_s+1,j+n_l+1,k)\}$

The spectral offspring $\mathcal{O}_{spec}(i,j,k)$ are defined in a similar way but only for the lowest spatial subband: if $i \geq n_s$ or $j \geq n_l$ we have $\mathcal{O}_{spec}(i,j,k) = \emptyset$. Otherwise, apart from the highest and lowest spectral frequency subbands, we have $\mathcal{O}_{spec}(i,j,k) = \{(i,j,2k),(i,j,2k+1)\}$ for $i < n_s$ and $j < n_l$. In the highest spectral frequency subbands, there is no offspring: $\mathcal{O}_{spec}(i,j,k) = \emptyset$ and in the lowest, coefficients are grouped by 2 to have a construction similar to SPIHT. Let n_b be the number of planes in the lowest spectral frequency subband:

- if $i < n_s$, $j < n_l$, k even: $\mathcal{O}_{spec}(i,j,k) = \emptyset$
- if $i < n_s$, $j < n_l$, k odd: $\mathcal{O}_{spec}(i,j,k) = \{(i,j,k+n_b),(i,j,k+n_b+1)\}$

In case of an odd number of coefficients in the lowest spectral subband, if n_b is an odd number, the above definition is slightly altered and the last even coefficient of the lowest spectral subband will have one descendant only.

With these relations, we have a separation in non-overlapping trees of all the coefficients of the wavelet transform of the image. The tree structure is illustrated in Figure 2 for three levels of decomposition in each direction. Each of the coefficients is the descendant of a root coefficient located in the lowest frequency subband. It has to be noted that all the coefficients belonging to the same tree correspond to a similar area of the original image, in the three dimensions.

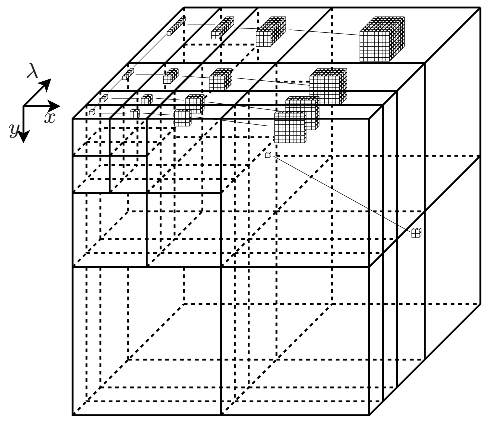


Fig. 2. Illustration of the tree structure. All descendants for a coefficient (i, j, k) with i and k being odds and j being even are shown.

We can compute the maximum number of descendants for a root coefficient (i, j, k) for a 5 level spatial and spectral decomposition. The maximum of descendants occurs when k is odd and at least either i or j is odd. For this situation, we have $1 + 2 + 2^2 + \ldots + 2^5 = 2^6 - 1$ spectral descendants and for each of these we have $1+2^2+(2^2)^2+(2^3)^2+\ldots+(2^5)^2 = 2^0+2^2+2^4+\ldots+2^{10} = (2^{12} - 1)/3$ spatial descendants. Let l_{spec} be the number of decomposition in the spectral direction and l_{spac} in the spatial direction we obtain the general formula:

$$n_{desc} = (2^{l_{spec}+1} - 1)\frac{2^{2(l_{spac}+1)} - 1}{3} \tag{1}$$

Thus the number of coefficients in the tree is at most 85995 ($l_{spec} = 5$ and $l_{spat} = 5$) if the given coefficient has both spectral and spatial descendants. Coefficient $(0, 0, 0)$, for example, has no descendant at all.

3 Block coding

3.1 Why Block Coding?

To provide random access, it is necessary to encode separately different areas of the image. Encoding separately portions of the image provides several other advantages. First, scan-based mode compression is made possible as the whole image is not necessary. Once again, we do not consider here the problem of the scan-based wavelet transform which is a separate issue. Secondly, encoding parts of the image separately also provides the ability to use different compression parameters for different parts of the image, enabling the possibility of high quality region of interest (ROI) and the possibility of discarding unused portions of the image. An unused portion of the image could be an area with clouds in remote sensing or irrelevant organs in a medical image. Third, transmission errors have a more limited effect in the context of separate coding; the error only affects a limited portion of the image. This strategy has been used for this particular purpose on the EZW algorithm in [27]. Finally, one limiting factor of the SPIHT algorithm is the complicated list processing requiring a large amount of memory. If the processing is done only on one part of the image at a a time, the number of coefficients involved is dramatically reduced and so is the memory necessary to store the control lists in SPIHT.

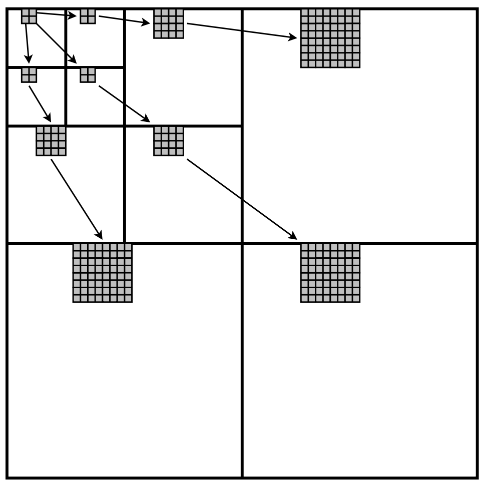


Fig. 3. Equivalence of the block structure for 2D, all coefficients in grey belong to the same block. In the following algorithm, an equivalent 3D block structure is used.

3.2 How?

With the tree structure defined in the previous section, a natural block organization appears. A tree-block (later simply referred to as *block*) is defined by 8 coefficients from the lowest subband forming a $2 \times 2 \times 2$ cube with all their descendants. All the coefficients linked to the root coefficient in the lowest subband shown on Figure 2 are part of the same tree-block together with seven other trees. Grouping the coefficients by 8 enables the use of neighbor similarities between coefficients. This is similar to the grouping of 2×2 in the original SPIHT patent [28] (see Fig. 3) and, as described in the previously mentioned patent, enables the possibility of including a Huffman coder for the 8 decision bits as the 8 coefficients values are strongly related. Results in this paper do not include this possible improvement.

Another advantage of this grouping is that the number of coefficients in each block will be the same, the only exception being the case where at least one dimension of the lowest subband is odd. The number of coefficients in one of these blocks can be calculated. In a $2 \times 2 \times 2$ root group, we have three coefficients which have the full sets of descendants, whose number is given by (1), three have only spatial descendants, one has only spectral descendants, and the last one has no descendant. The number of coefficients in a block, which determines the maximum amount of memory necessary for the compression, will finally be $262144 = 2^{18}$ (valid for 5 decompositions in the spatial and spectral directions).

Each of these blocks will be encoded using a modified version of the SPIHT algorithm as described in the next section.

4 Enabling resolution scalability

4.1 Original SPIHT algorithm

The original SPIHT algorithm processes the coefficients bitplane by bitplane. Coefficients are stored in three different lists according to their significance. The LSP (List of Significant Pixels) stores the coefficients that have been found significant in a previous bitplane and that will be refined in the following bitplanes. Once a coefficient is on the LSP, it will not be removed from it and this list is only growing. The LIP (List of Insignificant Pixels) contains the coefficients which are still insignificant, relative to the current bitplane and which are not part of a tree from the third list (LIS). Coefficients in the LIP are transferred to the LSP when they become significant. The third list is the LIS (List of Insignificant Sets). A set is said to be insignificant if all descendants, in the sense of the previously defined tree structure, are not significant in the current bit plane. For the bitplane t, we define the significance function S_t of a set $\mathcal{T}$ of coefficients :

$$S_t(\mathcal{T}) = \begin{cases} 0 \text{ if } \forall c \in \mathcal{T}, |c| < 2^t \\ 1 \text{ if } \exists c \in \mathcal{T}, |c| \geq 2^t \end{cases} \tag{2}$$

If $\mathcal{T}$ consists of a single coefficient, we denote its significance function by $S_t(i,j,k)$.

Let $\mathcal{D}(i,j,k)$ be all descendants of (i,j,k), $\mathcal{O}(i,j,k)$ only the offspring (i.e. the first level descendants) and $\mathcal{L}(i,j,k) = \mathcal{D}(i,j,k) - \mathcal{O}(i,j,k)$, the grand-descendant set. A type A tree is a tree where $\mathcal{D}(i,j,k)$ is insignificant (all descendants of (i,j,k) are insignificant) ; a type B tree is a tree where $\mathcal{L}(i,j,k)$ is insignificant (all grand-descendants of (i,j,k) are insignificant).

4.2 Introducing resolution scalability

SPIHT does not distinguish between different resolution levels. To provide resolution scalability, we need to process separately the different resolutions. A resolution comprises 1 or 3 subbands. To enable this we keep three lists for each resolution level r. Keeping separate lists for each resolution was done in the 2D case in [13] but it is not clear how they avoid the problem posed by this separation (described in 4.3). When $r = 0$ only coefficients from the low frequency subbands will be processed. Coefficients are processed according to the resolution level to which they correspond. For a 5-level wavelet decomposition in the spectral and spatial direction, a total of 36 resolution levels will be available (illustrated on Fig. 4 for 3-level wavelet and 16 resolution levels available). Each level r keeps in memory three lists: LSP_r, LIP_r and LIS_r.

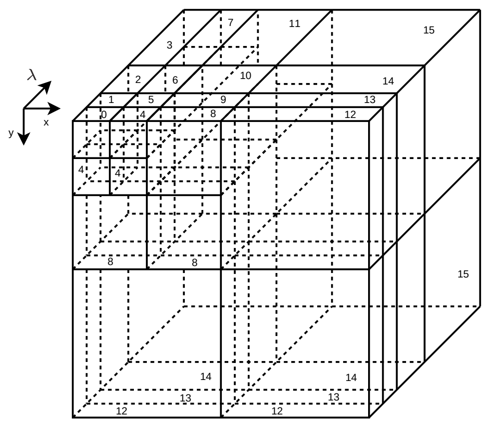


Fig. 4. Illustration of the resolution level numbering. If a low resolution image is required (either spectral or spatial), only subbands with a resolution number corresponding to the requirements are processed.

Some difficulties arise from this organization and the progression order to follow (Fig. 5). If the priority is given to full resolution scalability compared to the bit plane scalability, some extra precautions have to be taken. The different possibilities for scalability order are discussed in the next subsection. In the most complicated case, where all bit planes for a given resolution r are processed before the descendant resolution r_d (full resolution scalability), the last element to process for LSP_{r_d}, LIP_{r_d} and LIS_{r_d} for each bitplane t has to be remembered. Details of the algorithm are given below.

This new algorithm provides strictly the same amount of bits as the original SPIHT. The bits are just organized in a different order. With the block structure, the memory usage during the compression is dramatically reduced. The resolution scalability with its several lists does not increase the amount of memory necessary as the coefficients are just spread onto different lists.

4.3 Switching loops

The priority of scalability type can be chosen by the progression order of the two 'for' loops (highlighted in boldface type) in the previous algorithm. As written, the priority is resolution sclability, but these loops can be inverted to give quality scalability. The different progression orders are illustrated in Figure 5 (a) and (b). Processing the resolution completely before proceeding to the next one (Fig. 5 (b)) requires more precautions. When processing resolution r, a significant descendant set is partitioned into its offspring in r_d and its grand-descendant set. Therefore, some coefficients are added to LSP_{r_d} in the step marked **(2)** in Algorithm 2 (the problem is similar for the LIP_{r_d} and LIS_{r_d}). So even before processing resolution r_d, the LSP_{r_d} may contain some coefficients which were added at different bitplanes. The possible contents of an LSP_{r_d} are shown below in Equation (3) (the bitplane when a coefficient was added to the list is given in parentheses following the coordinate):

$$\begin{aligned} LSP_{r_d} = \{ & (i_0, j_0, k_0)(t_{19}), (i_1, j_1, k_1)(t_{19}), \ldots \\ & (i_n, j_n, k_n)(t_{12}), \ldots \\ & (i_{n'}, j_{n'}, k_{n'})(t_0), \ldots \}, \end{aligned} \tag{3}$$

19 being the highest bitplane in this case (depending on the image).

When we process LSP_{r_d} we should skip entries added at lower bitplanes than the current one. For example, there is no meaning to refine a coefficient added at t_{12} when we are working in bitplane t_{18}.

Furthermore at the step marked **(1)** in the algorithm above, when processing resolution r_d we add some coefficients in LSP_{r_d}. These coefficients have to be added at the proper position within LSP_{r_d} to preserve the order. When adding a coefficient at step **(1)** for the bitplane t_{19}, we insert it just after the other coefficient from bitplane t_{19} (at the end of the first line of Eqn. (3). Keeping the order avoids looking through the whole list to find the coefficients to process at a given bitplane and can be done simply with a pointer.

The bitstream structure obtained for this algorithm is shown in Figure 6 and called resolution scalable structure. If the resolution scalability is not a priority anymore and more SNR scalability is needed, the 'for' loops, stepping through resolutions and bitplanes, can be inverted to process one bitplane completely for all resolutions before going for the next bitplane. In this case the bitstream structure obtained is different and illustrated in Figure 7 and is called quality scalable structure. The differences between scanning order are shown on Figure 5.

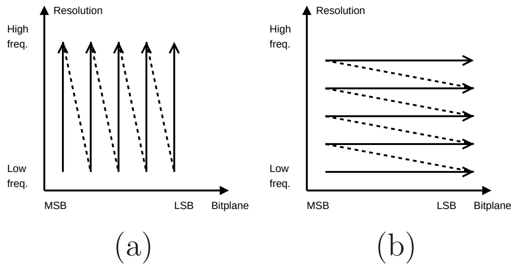


Fig. 5. Scanning order for SNR scalability (a) or resolution scalability (b)

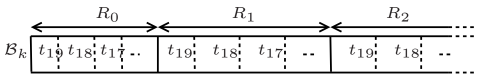


Fig. 6. Resolution scalable bitstream structure. R_0, R_1, ... denote the different resolutions, and t_{19}, t_{18}, ... the different bitplanes. This bitstream corresponds to the coding of one block $\mathcal{B}_k$.

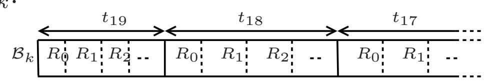


Fig. 7. Quality scalable bitstream structure. R_0, R_1, ... denote the different resolutions, and t_{19}, t_{18}, ... the different bitplanes. This bitstream corresponds to the coding of one block $\mathcal{B}_k$.

The algorithm described above possesses great flexibility and the same image can be encoded up to an arbitrary resolution level or down to a certain bitplane, depending on the two possible loop orders. The decoder can just proceed to the same level to decode the image. However, an interesting feature to have is the possibility to encode the image only once, with all resolution and all bitplanes and then during the decoding to choose which resolution and which bitplane to decode. One may need only a low resolution image with high radiometric precision or a high resolution portion of the image with rough radiometric precision.

When the resolution scalable structure is used (Fig. 6), it is easy to decode up to the desired resolution, but if not all bitplanes are necessary, we need a way to jump to the beginning of resolution 1 once resolution 0 is decoded for the necessary bitplanes. The problem is the same with the quality scalable structure (Fig. 7) exchanging bitplane and resolution in the problem description.

To overcome this problem, we need to introduce a block header describing the size of each portion of the bitstream. The new structures are illustrated in Figures 8 and 9. The cost of this header is negligible: the number of bits for each portion is coded with 24 bits, enough to code part sizes up to 16 Mbits. The lowest resolutions (resp. the highest bitplanes) which are using only few bits will be processed fully, whatever the specification is at the decoder as the cost in size and processing is low and therefore their sizes need not to be kept. Only the sizes of long parts are kept: we do not keep the size individually for the first few bitplanes or the first few resolutions, since they will be decoded in any case. Only lower bitplanes and higher resolutions (size of parts is in general well above 10000 bits), which comprises about 10 numbers (each coded with 32 bits to allow sizes up to 4Gb), to be written to the bitstream. Then, this header cost will remain below 0.1%.

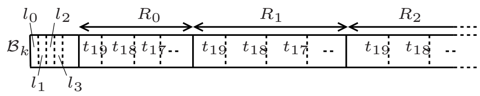


Fig. 8. Resolution scalable bitstream structure with header. The header allows the decoder to jump directly to resolution 1 without completely decoding or reading resolution 0. R_0, R_1, ... denote the different resolutions, t_{19}, t_{18}, ... the different bitplanes. l_i is the size in bits of R_i.

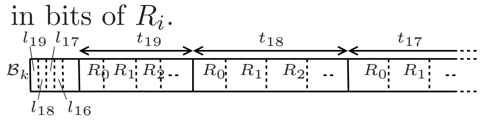


Fig. 9. Quality scalable bitstream structure with header. The header allows the decoder to continue the decoding of a lower bitplane without having to finish all the resolution at the current bitplane. R_0, R_1, ... denote the different resolutions, t_{19}, t_{18}, ... the different bitplanes. l_i is the size in bits of the bitplane corresponding to t_i.

As in [13], simple markers could have been used to identify the beginning of new resolutions of new bitplanes. Markers have the advantage to be shorter than a header coding the full size of the following block. However, markers make the full reading of the bitstream compulsory and the decoder cannot just jump to the desired part. As the cost of coding the header remains low, this solution is chosen.

5 Drawbacks of block processing and introduction of rate allocation

5.1 Rate allocation and keeping the SNR scalability

The problem of processing different areas of the image separately always resides in the rate allocation for each of these areas. A fixed rate for each area is usually not a suitable decision as complexity most probably varies across the image. If quality scalability is necessary for the full image, we need to provide the most significant bits for one block before finishing the previous one. This could be obtained by cutting the bitstream for all blocks and interleaving the parts in the proper order. With this solution, the rate allocation will not be available at the bit level due to the block organization and the spatial separation, but a trade-off with quality layers organization can be used.

5.2 Layer organization and rate-distortion optimization

The idea of quality layers is to provide in the same bitstream different targeted bitrates. For example, a bitstream can provide two quality layers: one quality layer for 1.0 bit per pixels (bpp) and one quality layer for 2.0 bpp. If the decoder needs a 1.0 bpp image, just the beginning of the bitstream is transferred and decoded. If a higher quality 2.0 bpp image is needed, the first layer is transmitted, decoded and then refined with the information from the second layer.

As the bitstream for each block is already embedded, to construct these layers, we just need to select the cutting points for each block and each layer leading to the correct bitrate with the optimal quality for the entire image. Once again, it has to be a global optimization and not only local, as complexity will vary across blocks.

A simple Lagrangian optimization method [29] gives the optimal cutting point for each block $\mathcal{B}_k$. As each block is coded in an embedded way, choosing a different cutting point will lead to a different rate R_k and a different distortion D_k. As the blocks are coded independently, their rates are additive and the final rate $R = \sum R_k$. The distortion measure can be chosen as additive to have the final distortion $D = \sum D_k$. A suitable measure is the squared error. Let c be a coefficient of the original image and $\tilde{c}$ its corresponding reconstruction, then

$$D_k = \sum_{c \in \mathcal{B}_k} (c - \tilde{c})^2 \tag{4}$$

The minimization of the Lagrangian objective function

$$J(\lambda) = \sum_k (D_k + \lambda R_k) \tag{5}$$

tells us that, given a parameter λ, the optimal cutting point for each block $\mathcal{B}_k$ is the one which minimizes the cost function $J_k(\lambda) = D_k + \lambda R_k$ [29]. For each λ and each block $\mathcal{B}_k$, it gives us an optimal function point $(R_k^\lambda, D_k^\lambda)$. The total bitrate for a given λ is $R^\lambda = \sum R_k^\lambda$ and the total distortion $D^\lambda = \sum D_k^\lambda$. By varying the λ parameter, an arbitrarily chosen bitrate is attainable. This simple algorithm appeared to be very similar to the PCRD-opt process used in JPEG 2000 for the rate allocation [30].

This optimization process leads to interleaving the bitstream for the different blocks. After the coding of each block, we need to keep the coded data in memory in order to perform this optimization. This could be seen as a high cost to keep the coded data in memory, but it has to be highlighted that in order to obtain progressive quality data we need to keep either the full image or the full bitstream in memory. Keeping the bitstream costs less than keeping the original image. However this is not compatible with the requirements for scan-based mode image coding. In this situation, a trade-off can be found doing the rate allocation for a group of blocks and using a buffer to balance a part of the complexity difference between the groups of blocks.

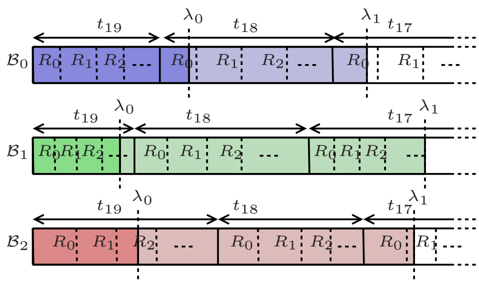


Fig. 10. An embedded scalable bitstream generated for each block $\mathcal{B}_k$. The rate-distortion algorithm selects different cutting points corresponding to different values of the parameter λ. The final bitstream is illustrated on Fig. 11.

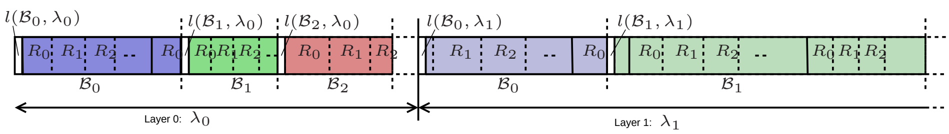


Fig. 11. The bitstreams are interleaved for different quality layers. To permit the random access to the different blocks, the length in bits of each part corresponding to a block $\mathcal{B}_k$ and a quality layer corresponding to λ_q is given by $l(\mathcal{B}_k, \lambda_q)$

5.3 Low cost distortion tracking: during the compression

In the previous part, we assumed that the distortion was known for every cutting point of the bitstream for one block. As the bitstream for one block is in general about several millions of bits, it is not conceivable to keep all this distortion information in memory. Only few hundred cutting points are remembered with their rate and distortion information.

Getting the rate for one cutting point is the easy part: one just has to count the number of bits before this point. The distortion requires more processing. The distortion value during the encoding of one block can be obtained with a simple tracking. Let us consider the instant in the compression when the encoder is adding one precision bit for one coefficient c at the bitplane t. Let c_t denote the new approximation of c in the bitplane t given by adding this new bit. c_{t+1} was the approximation of c at the previous bitplane.

SPIHT uses a deadzone quantizer so if the refinement bit is 0 we have $c_t = c_{t+1} - 2^{t-1}$ and if the refinement bit is 1 we have $c_t = c_{t+1} + 2^{t-1}$. Let call D^a the total distortion of the block after this bit was added and D^b the total distortion before. We have:

- with a refinement bit of 0:

$$\begin{aligned} D^a - D^b &= (c - c_t)^2 - (c - c_{t+1})^2 \\ &= (c_{t+1} - c_t)(2c - c_t - c_{t+1}) \\ &= 2^{t-1}\Big(2(c - c_{t+1}) + 2^{t-1}\Big) \end{aligned} \tag{6}$$

  giving

$$D^a = D^b + 2^{t-1}\Big(2(c - c_{t+1}) + 2^{t-1}\Big) \tag{7}$$

- with a refinement bit of 1:

$$D^a = D^b - 2^{t-1}\Big(2(c - c_{t+1}) - 2^{t-1}\Big) \tag{8}$$

Since this computation can be done using only right and left bit shifts and additions, the computational cost is low. The algorithm does not need to know the initial distortion value as the rate-distortion method holds if *distortion* is replaced by *distortion reduction*. The value can be high and has to be kept internally in a 64 bit integer. As seen before, we have 2^{18} coefficients in one block, and for some of them, the value can reach 2^{20}. Therefore 64 bits seems a reasonable choice and remains valid for the worst cases.

The evaluation of the distortion is done in the transform domain, directly on the wavelet coefficients. This can be done only if the transform is orthogonal. The 9/7 transform from [31] is approximately orthogonal. In [32], the computation of the weight to apply to each wavelet subband for the rate allocation is detailled. In our case with 5 decompositions on 3 dimensions, it has to be noted that these energy-based weights can be as low as 0.773 and as high as 1.126. Then equations 7 and 8 are modified to introduce the weight:

$$D^a = D^b + w_{x,y,l} * 2^{t-1}\Big(2(c - c_{t+1}) + 2^{t-1}\Big) \tag{9}$$

$$D^a = D^b - w_{x,y,l} * 2^{t-1}\Big(2(c - c_{t+1}) - 2^{t-1}\Big) \tag{10}$$

5.4 λ search: final bitstream formation

Usually, we are interested in specifying a certain bitrate R for a given quality layer rather than a meaningless parameter λ. To specify a targeted bitrate, we have to find the right value for λ that will give this global bitrate $R(\lambda) = R$.

Theorem 1 ([29]) *Let λ_1 and λ_2 be two different Lagrangian parameters as $\lambda_1 < \lambda_2$. Let (R_1, D_1) be the solution of $\min\{D + \lambda_1 R\}$ and (R_2, D_2) be the solution of $\min\{D + \lambda_2 R\}$. Then we have $R_1 \geq R_2$.*

(R_1, D_1) is the solution of $\min\{D + \lambda_1 R\}$ thus $D_1 + \lambda_1 R_1 \leq D_2 + \lambda_1 R_2$. We have likewise $D_2 + \lambda_2 R_2 \leq D_1 + \lambda_2 R_1$. Adding these inequalities, we get

$$\begin{aligned} \lambda_1 R_1 + \lambda_2 R_2 &\leq \lambda_1 R_2 + \lambda_2 R_1 \\ (\lambda_1 - \lambda_2) R_1 &\leq (\lambda_1 - \lambda_2) R_2 \\ R_1 &\geq R_2 \end{aligned}$$

Using this property, we can use a fast search algorithm to find the value of λ which is going to give the targeted bitrate. From a starting value λ, the bitrate $R(\lambda)$ is calculated. According to the relative value of $R(\lambda)$ and R, the value of λ is modified. A dichotomic search is particularly efficient in this situation. It has to be emphasized that this computation for the bitstream ordering occurs after the block compression and only involves the cutting points stored in memory. The search does not need to reprocess or access the original or compressed data. Once the λ giving the desired bitrate is found, we proceed to the next step and perform the bitstream interleaving to obtain the final bitstream (Fig. 11).

Table 1
Data sets

Image	Type	Dynamic	Size
moffett3	Hyperspectral	16 bits	$512 \times 512 \times 224$
jasper1	Hyperspectral	16 bits	$512 \times 512 \times 224$
cuprite1	Hyperspectral	16 bits	$512 \times 512 \times 224$
CT_skull	CT	8 bits	$256 \times 256 \times 192$
CT_wrist	CT	8 bits	$256 \times 256 \times 176$
MR_sag_head	MR	8 bits	$256 \times 256 \times 56$
MR_ped_chest	MR	8 bits	$256 \times 256 \times 64$

6 Results

6.1 Data

The hyperspectral data subsets originate from the Airborne Visible Infrared Imaging Spectrometer (AVIRIS) sensor. This hyperspectral sensor from NASA/JPL collects 224 contiguous bands in the range 400 nm to 2500 nm. Each band is approximately 10 nm spectral resolution. Depending on the sensor altitude, spatial resolution is between 4 and 20 m. We use radiance unprocessed data. The original AVIRIS scenes are $614 \times 512 \times 224$ pixels. For the simulations here, we crop the data to $512 \times 512 \times 224$ starting from the upper left corner of the scene. To make comparison easier with other papers, we use well-known data sets: particularly the scene 3 of the f970620t01p02_r03 run from AVIRIS on Moffett Field, but also scene 1 from the f970403t01p02_r03 run over Jasper Ridge and scene 1 from the f970619t01p02_r02 run over Cuprite site. MR and CT medical images are also used. The details of all the images are given in Table 1.

Error is given in terms of PSNR, RMSE and maximum error. For AVIRIS sets, PSNR (Peak Signal to Noise Ratio) is computed compared to the dynamic value of 16 bits: $\text{PSNR} = 10 \log_{10}(2^{16}-1)^2/\text{MSE}$, MSE being the Mean Square Error. RMSE is the Root Mean Square Error. All errors are measured in the final reconstructed dataset compared to the original data. Choosing a distortion measure suitable to hyperspectral data is not easy matter as shown in [33]. The rate-distortion optimization is based on the additive property of the distortion measure and optimized for the MSE. Our goal here is to choose an acceptable distortion measure for general use on different kinds of volume images. The MSE-based distortion measures here are appropriate and popular and are selected to facilitate comparisons.

Final rate, including all headers and required side information, is given in terms of Bit Per Pixel Per Band (bpppb).

An optional arithmetic coder is included to improve rate performance. The coder is coming from [25]. In the context of a reduced complexity algorithm, the slight improvement in performance introduced by the arithmetic coder does not seem worth the complexity increase. Results with arithmetic coder are given for reference. Unless stated otherwise, results in this paper do not include the arithmetic coder. Several particularities have to be taken into account to preserve the bitstream flexibility. First, contexts of the arithmetic coder have to be reset at the beginning of each part to be able to decode the bitstream partially. Secondly, the rate recorded during the rate-distortion optimization has to be the rate provided by the arithmetic coder.

6.2 Compression performance

The raw compression performances of the previously defined 3D-SPIHT-RARS (Random Access with Resolution Scalability) are compared with the best up-to-date method without taking into account the specific properties available for the previously defined algorithm. The reference results are obtained with the version 5.0 of Kakadu software [34] using the JPEG 2000 part 2 options: wavelet intercomponent transform to obtain a transform similar to the one used by our algorithm. PSNR values are similar to the best values published in [35]. The results were also confirmed using the latest reference implementation of JPEG 2000, the *Verification Model* (VM) version 9.1. Our results are not expected to be better but are here to show that the increase in flexibility does not come with a prohibitive cost in performance. It also has to be noted that the results presented here for 3D-SPIHT of 3D-SPIHT-RARS do not include any entropy coding of the SPIHT sorting output.

First, coding results are compared with the original SPIHT in Table 2. To be able to cope with the memory requirements of the original 3D-SPIHT, processing was limited to 256×256×224 data set which correspond to the lower right part of the f970620t01p02_r03 run from AVIRIS on Moffett Field (same area as in [33]). The source of performance decrease is the separation of the wavelet subbands at each bitplane which causes different bits to be kept if the bitstream is truncated. Once again, if lossless compression is required, the two algorithms, SPIHT and SPIHT-RARS provide exactly the same bits reordered (apart from the headers). We can see that the impact on performance remains low at high bitrate. The impact of taking into account the non-orthogonality of the 9/7 wavelet transform remains very low (within the rate accuracy). Every block has a similar structure and contains coefficients from all subbands with all weights. Adding weights in the distortion estimation has a much lower

Table 2
Impact of the modifications (rate accuracy better than 0.003 bpppb).

	1.0 bpppb	0.5 bpppb
Original 3D-SPIHT	75.79 dB	70.05 dB
3D-SPIHT-RARS (no weight)	75.75 dB	69.63 dB
3D-SPIHT-RARS	75.76 dB	69.62 dB

Table 3
Lossless performances (bpppb)

Image	JPEG 2000	SPIHT-RARS	SPIHT-RARS (with AC)
CT_skull	2.93	2.21	**2.16**
CT_wrist	1.78	1.31	**1.28**
MR_sag_head	**2.30**	2.42	2.37
MR_ped_chest	2.00	1.96	**1.94**
moffett3	**5.14**	5.47	5.38
jasper1	**5.54**	5.83	5.75
cuprite1	**5.28**	5.62	5.54

impact than in the case of JPEG 2000 where each coding unit (codeblock) has a different weight.

Computational complexity is not easy to measure, but one way to get a rough estimation is to measure the time needed for the compression of one image. The version of 3D-SPIHT here is a demonstration version and there is a lot of room for improvement. The compression time with similar options is 20 s for Kakadu v5.0, 600 s for VM 9.1 and 130 s for 3D-SPIHT-RARS. These values are given only to show that compression time is reasonable for a demonstration implementation and the comparison with the demonstration implementation of JPEG 2000, VM9.1 shows that this is the case. The value given here for 3D-SPIHT-RARS includes the 30 s necessary to perform the 3D wavelet transform with QccPack.

Table 3 compares the lossless performance of the two algorithms. For both, the same integer 5/3 wavelet transform is performed with the same number of decompositions in each direction. Performances are quite similar for the MR images. SPIHT-RARS outperforms JPEG 2000 on the CT images but JPEG 2000 gives a lower bitrate for hyperspectral images.

Tables 4 to 6 compare the lossy performances of the two algorithms. It is confirmed that the increase in flexibility of the 3D-SPIHT-RARS algorithm does not come with a prohibitive impact on performances. We can observe less than 1 dB difference between the two algorithms. A non contextual arithmetic

Table 4
PSNR for different rates for Moffett sc3

Rate (bpppb)	2.0	1.0	0.5	0.1
Kakadu v5.0	89.01	82.74	77.63	67.27
3D-SPIHT-RARS	88.18	81.95	76.60	66.39

Table 5
RMSE for different rates for Moffett sc3

Rate (bpppb)	2.0	1.0	0.5	0.1
Kakadu v5.0	2.32	4.78	8.61	28.39
3D-SPIHT-RARS	2.56	5.24	9.69	31.42

Table 6
Maximum error magnitude for different rates for Moffett sc3

Rate (bpppb)	2.0	1.0	0.5	0.1
Kakadu v5.0	24	66	157	1085
3D-SPIHT-RARS	37	80	161	1020

coder applied directly on the 3D-SPIHT-RARS bitstream already reduces this difference to 0.4 dB (not used in the presented results).

6.3 Resolution scalability from a single bitstream

Different resolutions and different quality levels can be retrieved from one bitstream. Table 7 presents different results on Moffett Field scene 3 changing the number of resolutions and bitplanes to decode the bitstream. The compression is done only once and the final bitstream is organized in different parts corresponding to different resolution and quality. From this single compressed bitstream, all these results are obtained changing the decoding parameters. Different bit depths and different resolutions are chosen arbitrarily to obtain a lower resolution and lower quality image. Distortion measures are provided for the lower resolution image as well as the bitrate necessary to transmit or store this image.

For the results presented in this table, similar resolutions are chosen for spectral and spatial directions but this is not mandatory as illustrated in Figure 12. The reference low resolution image is the low frequency subband of the wavelet transform up to the desired level. To provide an accurate radiance value, coefficients are scaled properly to compensate gains due to the wavelet filters (depending on the resolution level).

Table 7 shows for example that discarding the 6 lower bitplanes, a half resolution image can be obtained with a bitrate of 0.203 bpppb and a PSNR of

Table 7
Bits read for different parameters and quality for the image moffett3. The compression is done only once.

Number of non decoded bitplanes: 0

Resolution	Full	1/2	1/4	1/8
bpppb read	5.309	1.569	0.247	0.038
PSNR (dB)	106.57	105.58	108.27	114.77
RMSE	0.31	0.34	0.25	0.12
Time (s)	59.43	21.82	7.17	3.54

Number of non decoded bitplanes: 2

Resolution	Full	1/2	1/4	1/8
bpppb read	2.857	0.989	0.198	0.033
PSNR (dB)	91.89	99.45	104.43	109.54
RMSE	1.67	0.70	0.39	0.22
Time (s)	42.33	18.03	6.86	3.62

Number of non decoded bitplanes: 4

Resolution	Full	1/2	1/4	1/8
bpppb read	1.016	0.475	0.132	0.027
PSNR (dB)	82.03	90.16	97.99	103.52
RMSE	5.18	2.03	0.82	0.44
Time (s)	18.18	10.05	5.34	3.45

80 dB (for this resolution).

In Figure 12, we can see different hyperspectral cubes extracted from the same bitstream with different spatial and spectral resolutions. The face of the cube is a color composition from different subbands. The spectral bands chosen for the color composition in the sub-resolution cube correspond to those of the original cube. Some slight differences from the original cube can be observed on the sub-resolution one, due to weighted averages from wavelet transform filtering spanning contiguous bands.

6.4 ROI coding and selected decoding

The main interest of the present algorithm is in its flexibility. The bitstream obtained in the resolution scalable mode can be decoded at variable spectral

Number of non decoded bitplanes: 6				
Resolution	Full	1/2	1/4	1/8
bpppb read	0.327	0.203	0.079	0.020
PSNR (dB)	74.02	80.11	87.61	95.68
RMSE	13.05	6.47	2.73	1.08
Time (s)	7.70	6.14	4.40	3.36
Number of non decoded bitplanes: 8				
Resolution	Full	1/2	1/4	1/8
bpppb read	0.104	0.077	0.039	0.013
PSNR (dB)	66.72	70.77	76.74	84.34
RMSE	30.23	18.97	9.53	3.98
Time (s)	4.51	4.26	3.68	3.26
Number of non decoded bitplanes: 10				
Resolution	Full	1/2	1/4	1/8
bpppb read	0.030	0.025	0.016	0.007
PSNR (dB)	59.50	62.39	66.81	73.04
RMSE	69.41	49.76	29.92	14.60
Time (s)	3.47	3.45	3.29	3.16

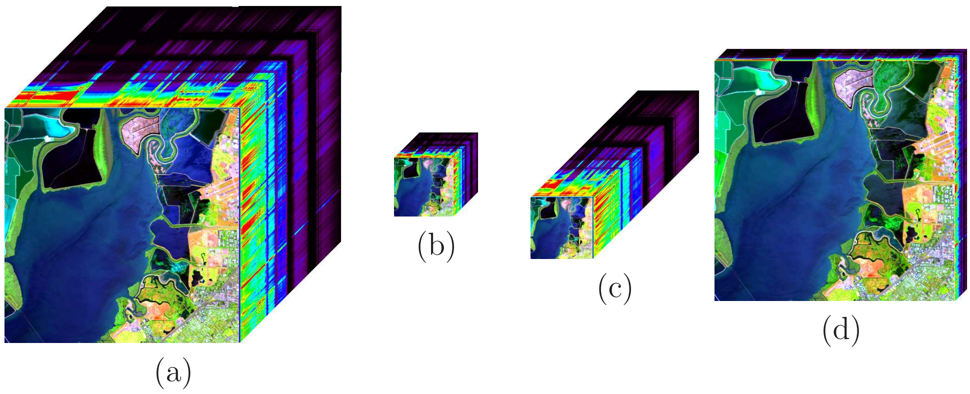
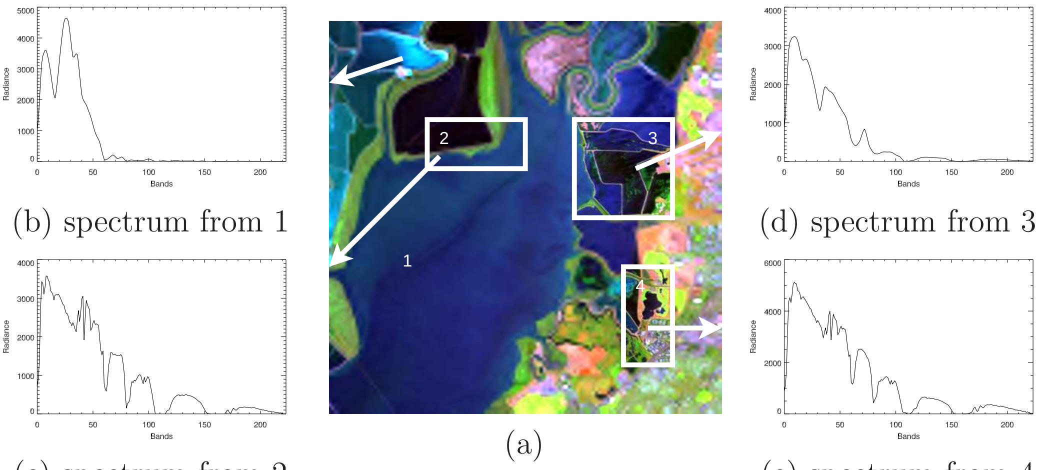


Fig. 12. Example of hyperspectral cube with different spectral and spatial resolution decoded from the same bitstream. (a) is the original hyperspectral cube. (b) is 1/4 for spectral resolution and 1/4 for spatial resolution. (c) is full spectral resolution and 1/4 spatial resolution. (d) is full spatial resolution and 1/8 spectral resolution.

and spatial resolutions for each data block. This is done reading, or transmitting, a minimum number of bits. Any area of the image can be decoded up to any spatial resolution, any spectral resolution and any bitplane. This property



(b) spectrum from 1
(d) spectrum from 3
(a)
(c) spectrum from 2
(e) spectrum from 4

Fig. 13. Example of a decompressed image with different spatial and spectral resolution for different areas. Background (area 1) is with low spatial resolution and low spectral resolution as is can be seen on the spectrum (b). Area 2 has low spatial resolution and highspectral resolution (c), area 3 has high spatial resolution but low spectral resolution (d). Finally, area 4 has both high spectral and spatial resolutions. This decompressed image was obtained from a generic bitstream, reading the minimum amount of bits.

is illustrated on Figure 13. Most of the image background (area 1) is decoded at low spatial and spectral resolutions, dramatically reducing the amount of bits. Some specific areas are more detailled and, offer the full spectral resolution (area 2), the full spatial resolution (area 3) or both (area 4). The image from Figure 13 was obtained reading only 16907 bits from the original 311598 bits bitstream.

The region of interest can also be selected during the encoding by adjusting the number of bitplanes to be encoded for a specific block. In the context of on-board processing, it would enable further reduction of the bitrate. The present encoder provides all these capabilities. For example, an external clouds detection loop could be added to adjust the compression paremeter to reduce the resolution when clouds are detected. This would decrease the bitrate on these parts.

7 Conclusion

An adaptation of the 3D-SPIHT algorithms is presented. The 3D-SPIHT-RARS algorithm enables resolution scalability for spatial and spectral dimensions independently. Coding different areas of the image separately enables random access and region of interest coding with a reduction in memory usage during the compression. Thanks to the rate-distortion optimization between the different areas, all this is done without sacrificing compression capabilities. All these features seem also possible with the JPEG 2000 standard. However, implementation providing multiresolution transforms is very recent and does not provide yet all the flexibility proposed here, particularly on the spectral direction.

The use of an arithmetic coder slightly increases compression performance, but at the cost of an increase in the complexity. It has to be highlighted that the 3D-SPIHT-RARS algorithm does not need to rely on arithmetic coding to obtain competitive results to JPEG2000.

Acknowledgments

This work has been carried out primarily at Rensselaer Polytechnic Institute under the financial support of *Centre National d'Études Spatiales* (CNES), *TeSA*, *Office National d'Études et de Recherches Aérospatiales* (ONERA) and *Alcatel Alenia Space*. Partial support was also provided by the Office of Naval Research under Award No. N0014-05-10507. The authors wish to thank their supporters and *NASA/JPL* for providing the hyperspectral images used during the experiments.

References

[1] S. Krishnamachari, R. Chellappa, Multiresolution Gauss-Markov random field models for texture segmentation, IEEE Transactions on Image Processing 39 (2) (1997) 251–267.

[2] L. M. Bruce, C. Morgan, S. Larsen, Automated detection of subpixel hyperspectral targets with continuous and discrete wavelet transforms, IEEE Transactions on Geoscience and Remote Sensing 39 (10) (2001) 2217–2226.

[3] International charter: Space and major disasters, http://www.disasterscharter.org/main_e.html.

[4] Y. Langevin, O. Forni, Image and spectral image compression for four experiments on the ROSETTA and Mars Express missions of ESA, in: Applications of Digital Image Processing XXIII, Vol. 4115, SPIE, 2000, pp. 364–373.

[5] P.-S. Yeh, P. Armbruster, A. Kiely, B. Masschelein, G. Moury, C. Schaefer, C. Thiebaut, The new CCSDS image compression recommendation, in: IEEE Aerospace Conference, IEEE, 2005.

[6] D. Van Buren, A high-rate JPEG2000 compression system for space, in: IEEE Aerospace Conference, IEEE, 2005, pp. 1–7.

[7] P. L. Dragotti, G. Poggi, A. R. P. Ragozini, Compression of multispectral images by three-dimensional spiht algorithm, IEEE Transactions on Geoscience and Remote Sensing 38 (1) (2000) 416–428.

[8] Z. Xiong, X. Wu, S. Cheng, J. Hua, Lossy-to-lossless compression of medical volumetric data using three-dimensional integer wavelet transforms, IEEE Transactions on Medical Imaging 22 (3) (2003) 459–470.

[9] B.-J. Kim, Z. Xiong, W. A. Pearlman, Low bit-rate scalable video coding with 3-D set partitioning in hierarchical tress (3-D SPIHT), IEEE Transactions on Circuits and Systems for Video Technology 10 (8) (2000) 1374–1387.

[10] C. He, J. Dong, Y. F. Zheng, Optimal 3-D coefficient tree structure for 3-D wavelet video coding, IEEE Transactions on Circuits and Systems for Video Technology 13 (10) (2003) 961–972.

[11] E. Christophe, C. Mailhes, P. Duhamel, Hyperspectral image compression: Adapting SPIHT and EZW to anisotropic 3D wavelet coding, IEEE Transactions on Image Processing(submitted).

[12] W. A. Pearlman, Trends of tree-based, set-partitioning compression techniques in still and moving image systems, in: Proceedings Picture Coding Symposium 2001 (PCS 2001), 2001, pp. 1–8, (invited, keynote paper).

[13] H. Danyali, A. Mertins, Fully spatial and SNR scalable, SPIHT-based image coding for transmission over heterogenous networks, Journal of Telecommunications and Information Technology (2) (2003) 92–98.

[14] H. Danyali, A. Mertins, Flexible, highly scalable, object-based wavelet image compression algorithm for network applications, Vision, Image and Signal Processing, IEE Proceedings 151 (6) (2004) 498–510.

[15] X. Tang, W. A. Pearlman, Scalable hyperspectral image coding, in: IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP'05, Vol. 2, 2005, pp. 401–404.

[16] Y. Cho, W. A. Pearlman, A. Said, Low complexity resolution progressive image coding algorithm: Progres (progressive resolution decompression), in: IEEE International Conference on Image Processing, Vol. 3, 2005, pp. 49–52.

[17] S. Cho, W. A. Pearlman, Multilayered protection of embedded video bitstreams over binary symmetric and packet erasure channels, Journal of Visual Communication and Image Representation 16 (3) (2005) 359–378.

[18] X. Tang, W. A. Pearlman, Progressive resolution coding of hyperspectral imagery featuring region of interest access, in: SPIE (Ed.), SPIE Defense & Security 2005, Visual Information Processing XIV, Vol. 5817, 2005, pp. 570–580.

[19] Y. Liu, W. A. Pearlman, Region of interest access with three-dimensional SBHP algorithm, in: SPIE Electronic Imaging 2006, 2006.

[20] A. Said, W. A. Pearlman, A new, fast, and efficient image codec based on set partitioning in hierarchical trees, IEEE Transactions on Circuits and Systems for Video Technology 6 (3) (1996) 243–250.

[21] D. S. Taubman, M. W. Marcellin, JPEG2000 Image Compression Fundamentals, Standards and Practice, Kluwer Academic Publishers, Boston, MA, 2002.

[22] B. Penna, T. Tillo, E. Magli, G. Olmo, Progressive 3-D coding of hyperspectral images based on JPEG 2000, IEEE Geoscience and Remote Sensing Letters 3 (1) (2006) 125–129.

[23] X. Tang, W. A. Pearlman, J. W. Modestino, Hyperspectral image compression using three-dimensional wavelet coding, in: Image and Video Communications and Processing, Vol. 5022, SPIE, 2003, pp. 1037–1047.

[24] Information technology – jpeg 2000 image coding system: Extensions (2004).

[25] J. E. Fowler, QccPack - Quantization, Compression, and Coding Library, http://qccpack.sourceforge.net/ (2006).

[26] A. Said, W. A. Pearlman, Image compression using the spatial orientation tree, in: IEEE International Symposium on Circuits and Systems, No. 1, IEEE, 1993, pp. 279–282.

[27] C. D. Creusere, A new method of robust image compression based on the embedded zerotree wavelet algorithm, IEEE Transactions on Image Processing 6 (10) (1997) 1436–1442.

[28] W. A. Pearlman, A. Said, Data compression using set partitioning in hierarchical trees, United States Patent 5,764,807 (9 Jun. 1998).

[29] Y. Shoham, A. Gersho, Efficient bit allocation for an arbitrary set of quantizers, IEEE Transactions on Acoustics, Speech, and Signal Processing 36 (9) (1988) 1445–1453.

[30] D. Taubman, High performance scalable image compression with EBCOT, IEEE Transactions on Image Processing 9 (7) (2000) 1158–1170.

[31] M. Antonini, M. Barlaud, P. Mathieu, I. Daubechies, Image coding using wavelet transform, IEEE Transactions on Image Processing 1 (2) (1992) 205–220.

[32] B. Usevitch, Optimal bit allocation for biorthogonal wavelet coding, in: Data Compression Conference, IEEE, 1996, pp. 387–395.

[33] E. Christophe, D. Léger, C. Mailhes, Quality criteria benchmark for hyperspectral imagery, IEEE Transactions on Geoscience and Remote Sensing 43 (09) (2005) 2103–2114.

[34] D. Taubman, Kakadu Software v 5.0, http://www.kakadusoftware.com/ (2006).

[35] J. T. Rucker, J. E. Fowler, N. H. Younan, JPEG2000 coding strategies for hyperspectral data, in: IEEE International Geoscience and Remote Sensing Symposium, IGARSS'05, Vol. 1, 2005, pp. 128–131.

Initialization step:

- Initialize t to the number of bitplanes
- LSP $= \emptyset$
- LIP : all the coefficients without any parents (coefficients from the lowest frequency subband)
- LIS : all coefficients from the LIP with descendants

Sorting pass : For each entry (i, j, k) of the LIP

- Output $S_t(i, j, k)$
- If $S_t(i, j, k) = 1$, move (i, j, k) in LSP and output the sign of $c_{i,j,k}$

For each entry (i, j, k) of the LIS

- If the entry is type A
  - · Output $S_t(\mathcal{D}(i, j, k))$
  - · If $S_t(\mathcal{D}(i, j, k)) = 1$ then

    For all $(i', j', k') \in \mathcal{O}(i, j, k)$: output $S_t(i', j', k')$; If $S_t(i', j', k') = 1$, add (i', j', k') to the LSP and output the sign of $c_{i',j',k'}$ else, add (i', j', k') to the end of the LIP

    If $\mathcal{L}(i, j, k) \neq \emptyset$, move (i, j, k) to the end of the LIS as a type B entry

    Else, remove (i, j, k) from the LIS
- If the entry is type B
  - · Output $S_t(\mathcal{L}(i, j, k))$
  - · If $S_t(\mathcal{L}(i, j, k)) = 1$

    Add all the $(i', j', k') \in \mathcal{O}(i, j, k)$ to the end of the LIS as a type A entry

    Remove (i, j, k) from the LIS

Refinement pass:

- For all entries (i, j, k) of the LSP, except those included in the last sorting pass, output the t^{th} most significant bit of $c_{i,j,k}$

Decrement t and return to the sorting pass.

Resolution scalable 3D SPIHT
Initialization step:

- Initialize t to the number of bitplanes
- $\text{LSP}_0 = \emptyset$
- LIP_0 : all the coefficients without any parents (the 8 root coefficients of the block)
- LIS_0 : all coefficients from the LIP_0 with descendants (7 coefficients as only one has no descendant)
- For $r \neq 0$, $\text{LSP}_r = \text{LIP}_r = \text{LIS}_r = \emptyset$

For each r from 0 to maximum resolution
For each t from the highest bitplane to 0 (bitplanes)
Sorting pass : For each entry (i,j,k) of the LIP_r which had been added at a threshold strictly greater to the current t

- Output $S_t(i,j,k)$
- If $S_t(i,j,k) = 1$, move (i,j,k) to LSP_r and output the sign of $c_{i,j,k}$ **(1)**

For each entry (i,j,k) of the LIS_r **which had been added at a threshold greater or equal to the current t**

- If the entry is type A
  - · Output $S_t(\mathcal{D}(i,j,k))$
  - · If $S_t(\mathcal{D}(i,j,k)) = 1$ then

    For all $(i',j',k') \in \mathcal{O}(i,j,k)$: output $S_t(i',j',k')$; If $S_t(i',j',k') = 1$, add (i',j',k') to the LSP_{r_d} and output the sign of $c_{i',j',k'}$ else, add (i',j',k') to the end of the LIP_{r_d} **(2)**

    If $\mathcal{L}(i,j,k) \neq \emptyset$, move (i,j,k) to the LIS_r as a type B entry

    Else, remove (i,j,k) from the LIS_r
- If the entry is type B
  - · Output $S_t(\mathcal{L}(i,j,k))$
  - · If $S_t(\mathcal{L}(i,j,k)) = 1$

    Add all the $(i',j',k') \in \mathcal{O}(i,j,k)$ to the LIS_{r_d} as a type A entry

    Remove (i,j,k) from the LIS_r

Refinement pass:

- For all entries (i,j,k) of the LSP_r **which had been added at a threshold strictly greater than the current** t : Output the t^{th} most significant bit of $c_{i,j,k}$