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

Emmanuel Christophe
CNES and TeSA
bpi 1219 - 18, av. E. Belin
31401 Toulouse cedex 9, FRANCE
Email: e.christophe@ieee.org

W. A. Pearlman
ECSE Department
Rensselaer Polytechnic Institute
Troy, NY 12180-3590 USA
Email: pearlw@ecse.rpi.edu

***Abstract*— With the increase of remote sensing images, fast access to some features of the image is becoming critical. This access could be some part of the spectrum, some area of the image, high spatial resolution. An adaptation of 3D-SPIHT image compression algorithm is presented to allow random access to some part of the image, whether spatial or spectral. Resolution scalability is also available, enabling the decoding of different resolution images from the compressed bitstream of the hyperspectral data. Final spatial and spectral resolutions are chosen independently. From the same compressed bitstream, various resolutions and quality images can be extracted while reading a minimum amount of bits from the coded data. All this is done while reducing the memory necessary during the compression.**

I. 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. Fast access to lower resolution images is also required. As the size of images is growing, these additional features are becoming required properties for new compression algorithms.

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 [1]. This interest in zerotree-based coding 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 zero-trees principles [2] because JPEG 2000 suffers from implementation difficulties as described in [3] (in the context of implementation compatible with space constraints).

This paper presents the adaptation of the well-known SPIHT algorithm [4] for 3D data enabling random access and resolution scalability or quality scalability. Compression performance is compared with JPEG 2000 [5].

II. Introducing block coding

A. Interest

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.

- scan-based mode compression is made possible as the whole image is not necessary.
- 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.
- transmission errors have a more limited effect in the context of separate coding; the error only affects a limited portion of the image
- reduced 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.

B. How?

The tree structure defined in [6] (illustrated on Fig.1) is used. The wavelet decomposition is completely done on each spectrum (one direction) before applying a standard multiresolution decomposition on each resulting band (alternating the two other directions). A tree is defined according to 2D SPIHT in the spatial dimensions: on the lowest spatial subband one out of four coefficient has no descendant and three out of four

have each four descendants. The spectral link in the spectral direction is kept only for the lowest spatial subbands.

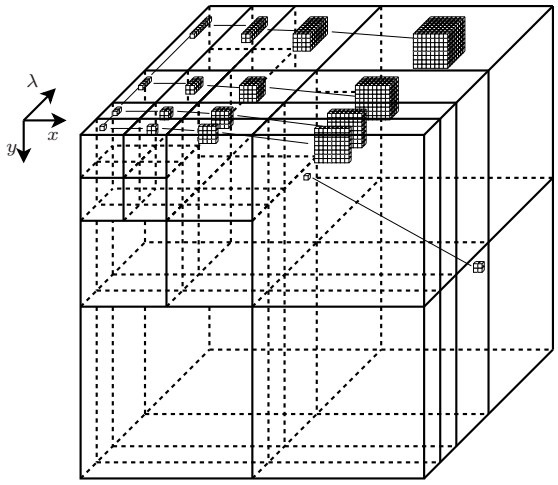


Fig. 1. Illustration of the tree structure. All descendants for a coefficient (i, j, k) with i and k being odds and j being even are shown.

With this structure a natural block organization appears. A tree-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 Fig. 1 are part of the same tree-block together with seven other trees. Each tree-block contains the same number of coefficients (which is 2^{18} for a 5 level decomposition) and describes a region of the original image. This is similar to the grouping of 2×2 in the original SPIHT patent [7].

Each of these tree-blocks (later refered as blocks) will be encoded using a modified version of the SPIHT algorithm as described in the next section.

III. Enabling resolution scalability

A. Introducing resolution scalability in SPIHT

The original SPIHT algorithm processes the coefficients bitplane by bitplane. Coefficients are stored in three different lists according to their significance. The notation used here is similar to that of the original SPIHT [4].

SPIHT does not originally distinguish between different resolution levels. To provide resolution scalability, we need to process separately the different resolutions. To enable this we keep three lists for each resolution level r. When $r = 0$ only the highest pyramid level will be processed. For a 5-level wavelet decomposition in the spectral and spatial direction, decoding only the highest pyramid level will correspond to the original image at the scale 1/32 on each dimension.

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. The bitstream structure obtained for this algorithm is shown in Fig. 2 and called resolution scalable structure.

Fig. 2. Resolution scalable bitstream structure. R_0, R_1, ...denote the different resolutions, t_16, t_15, ...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 and down to a certain bitplane. 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 resolutions 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.

With the stream structure shown in Fig. 2, 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.

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 3.

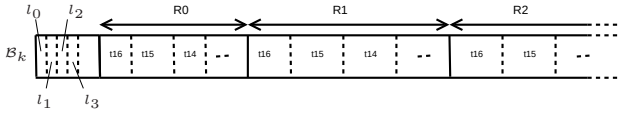


Fig. 3. Resolution scalable bitstream structure with header. The header allows the decoder to jump directly to resolution 1 without completely decoding or reading resolution 0. R_0, R_1, ...denote the different resolutions, t_16, t_15, ...the different bitplanes. l_i is the size in bits of R_i.

As in [8], 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.

IV. Drawbacks of block processing and introduction of rate allocation

A. 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.

B. 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 [9] 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.

The Lagrangian optimization [9] tells that given a parameter λ, the optimal cutting point for each block $\mathcal{B}_k$ is the one which minimizes the cost function $J_k(\lambda) = D_k + \lambda R_k$. 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 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 differences between the groups of blocks.

C. 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.

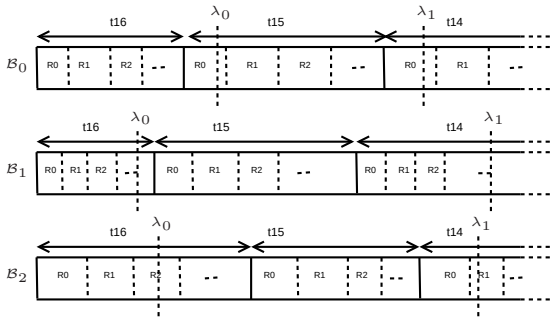


Fig. 4. An embedded scalable bitstream generated for each block $\mathcal{B}_k$. The rate-distortion algorithm selects different cutting points corresponding to different values of the parameter λ. The final bitstream is illustrated on Fig. 5.

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:

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

  giving

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

- with a refinement bit of 1:

$$D^a = D^b - 2^{t-1}\Big(2(c - c_{t+1}) - 2^{t-1}\Big) \quad (3)$$

Since this computation can be done using only right and left bit shifts and additions, the computational cost is low.

D. 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 $R(\lambda) = R$.

Using the property highlighted in [9], 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

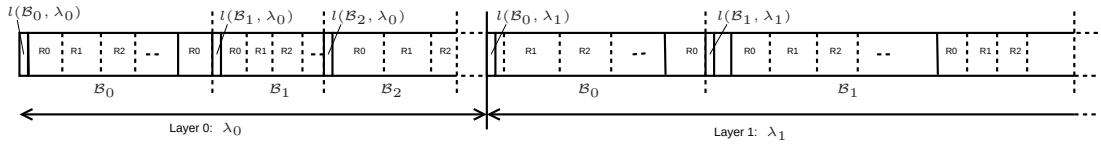


Fig. 5. The bitstreams are interleaved for different quality layers. To permit the random access to the different blocks, the length in bits of each part corresponding to a block $\mathcal{B}_k$ and a quality layer corresponding to λ_q is given by $l(\mathcal{B}_k, \lambda_q)$

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. 5).

V. Results

A. 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. Classical MR and CT medical images are also used.

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: $\mathrm{PSNR} = 10\log_{10}(2^{16}-1)^2/\mathrm{MSE}$, 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 as shown in [10]. Distortion measures here are popular and are selected to facilitate comparisons.

B. Compression performances

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 [11] 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 [12]. The performances were also confirmed using the lastest 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.

TABLE I

Lossless performances (bpppb)

Image	JPEG 2000	SPIHT-RARS
CT_skull	2.93	**2.21**
CT_wrist	1.78	**1.31**
MR_sag_head	**2.30**	2.42
MR_ped_chest	2.00	**1.96**
moffett3	**5.14**	5.47
jasper1	**5.54**	5.83
cuprite1	**5.28**	5.62

TABLE II

Distortion for different rates for Moffett sc3

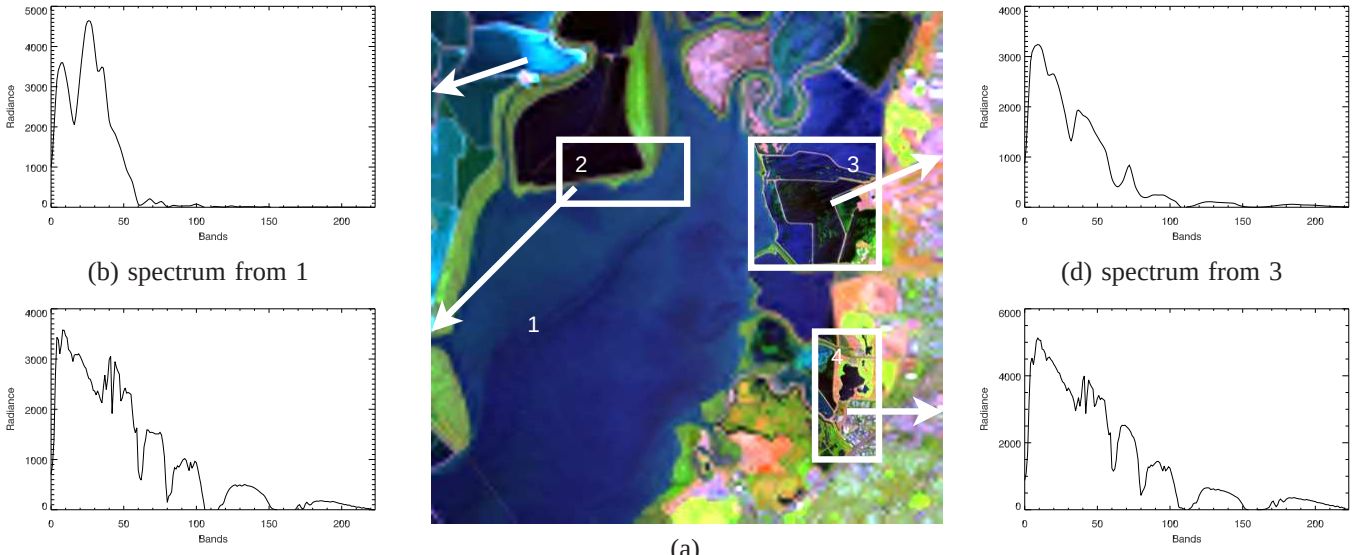
Rate (bpppb)	2.0	1.0	0.5	0.1
Kakadu v5.0 (PSNR)	89.01	82.74	77.63	67.27
3D-SPIHT-RARS (PSNR)	88.18	81.95	76.60	66.39
Kakadu v5.0 (RMSE)	2.32	4.78	8.61	28.39
3D-SPIHT-RARS (RMSE)	2.56	5.24	9.69	31.42
Kakadu v5.0 (Emax)	24	66	157	1085
3D-SPIHT-RARS (Emax)	37	80	161	1020

Table I 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 II compares 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.

C. 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 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.



(b) spectrum from 1

(d) spectrum from 3

(a)

(c) spectrum from 2

(e) spectrum from 4

Fig. 6. 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.

This property is illustrated on Fig. 6. 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 Fig. 6 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.

VI. 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.

Acknowledgments

This work has been carried out 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] Y. Langevin and 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.

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

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

[4] A. Said and 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*, vol. 6, no. 3, pp. 243–250, June 1996.

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

[6] E. Christophe, C. Mailhes, and P. Duhamel, "Hyperspectral image compression: Adapting SPIHT and EZW to anisotropic 3D wavelet coding," *IEEE Transactions on Image Processing*, 2006, (submitted).

[7] W. A. Pearlman and A. Said, "Data compression using set partitioning in hierarchical trees," United States Patent 5,764,807, 9 June 1998.

[8] H. Danyali and A. Mertins, "Fully spatial and SNR scalable, SPIHT-based image coding for transmission over heterogenous networks," *Journal of Telecommunications and Information Technology*, no. 2, pp. 92–98, 2003.

[9] Y. Shoham and A. Gersho, "Efficient bit allocation for an arbitrary set of quantizers," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 36, no. 9, pp. 1445–1453, Sept. 1988.

[10] E. Christophe, D. Léger, and C. Mailhes, "Quality criteria benchmark for hyperspectral imagery," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 43, no. 09, pp. 2103–2114, Sept. 2005.

[11] D. Taubman, "Kakadu Software v 5.0," http://www.kakadusoftware.com/, 2006.

[12] J. T. Rucker, J. E. Fowler, and N. H. Younan, "JPEG2000 coding strategies for hyperspectral data," in *Geoscience and Remote Sensing Symposium. IGARSS'05*, vol. 1. IEEE, July 2005, pp. 128–131.