Compression of Hyperspectral Images with LVQ-SPECK

Alessandro J. S. Dutra, William A. Pearlman
ECSE Department, Rensselaer Polytechnic Institute – Troy, NY – 12180
a.dutra@ieee.org, pearlw@ecse.rpi.edu

Eduardo A. B. da Silva
PEE/COPPE - DEL, Universidade Federal do Rio de Janeiro
eduardo@lps.ufrj.br

Abstract

We discuss the use of lattice vector quantizers in conjunction with a quadtree-based sorting algorithm for the compression of multidimensional data sets, as encountered, for example, when dealing with hyperspectral imagery. An extension of the SPECK algorithm is presented that deals with vector samples and is used to encode a group of successive spectral bands extracted from the hyperspectral image original block. We evaluate the importance of codebook choice by showing that a choice of dictionary that better matches the characteristics of the source during the sorting pass has as big an influence in performance as the use of a transform in the spectral direction. Finally, we provide comparison against state-of-the-art encoders, both 2D and 3D ones, showing the proposed encoding method is very competitive, especially at small bit rates. We discuss the use of lattice vector quantizers in conjunction with a quadtree-based sorting algorithm for the compression of multidimensional data sets, as encountered, for example, when dealing with hyperspectral imagery. An extension of the SPECK algorithm is presented that deals with vector samples and is used to encode a group of successive spectral bands extracted from the hyperspectral image original block. We evaluate the importance of codebook choice by showing that a choice of dictionary that better matches the characteristics of the source during the sorting pass has as big an influence in performance as the use of a transform in the spectral direction. Finally, we provide comparison against state-of-the-art encoders, both 2D and 3D ones, showing the proposed encoding method is very competitive, especially at small bit rates.

I. Introduction

The compression of hyperspectral images has been given a lot of attention in recent years, due not only to the often sensitive nature of the acquired information but also because of the usually large amount of data needed to represent it. Methods spanning from direct quantization of spectral values [1] to those that employ the discrete wavelet transform [2] as a decorrelating step were developed, providing good compression capabilities along with good quality representation - even lossless, if desired.

In [1], Motta et al. define a partition of the spectral space whose boundaries are optimized by repeated application of a Generalized Lloyd Algorithm [3] (GLA) variant. Considering the original data set to have a dimension D, the design of a D-dimensional

1068-0314/08 $25.00 © 2008 IEEE
DOI 10.1109/DCC.2008.88

vector quantizer, which is usually computationally prohibitive, would be required. Instead, the method chooses to design N vector quantizers, each with dimension d_i, where $\sum_{i=0}^{N} d_i = D$. The resulting Partitioned Vector Quantizer is then the Cartesian product of all the lower dimensional dictionaries. In order to remove part of the remaining source redundancy, each resulting vector quantization (VQ) index is also conditionally entropy encoded based on a causal set of spatially and spectrally adjacent indices.

As opposed to the previously described method, which encodes the spectral band intensity values directly, a number of methods that apply a decorrelating transform were developed. In [2], a 3D version of the quadtree-based codec SPECK [4] was introduced. 3D-SPECK takes small portions of the hyperspectral block, e.g., 16 spectral bands at a time, applies a 3D discrete wavelet transform (DWT) and extends the concept of partitioning sets and rules to the three dimensional case. Given the energy compaction properties of the DWT, and SPECK's efficiency in the coding of significance information, the method achieves very good compression results.

The compression algorithm herein proposed is a variant of the original 2D-SPECK, tailored to deal with multidimensional data. In particular, if we consider each spectral vector as a "multidimensional pixel", the encoding steps are exactly the same, the only changes being the definition of vector significance against a threshold, the existence of a lattice-based vector codebook and a threshold scaling factor α. Being a successive approximation based method, our vector-based extension of SPECK retains those characteristics that make this class of encoders a very successful one, such as the embeddedness of the bit stream, along with its quality/rate scalability.

This article is organized as follows. Section II presents the basics of successive approximation methods based on lattice vector quantizers. The encoding algorithm and its differences to the basic (scalar) method are described in section III, while the results obtained in the compression of standard AVIRIS hyperspectral images appears in section IV. Lastly, section V presents our conclusions and perspectives of future work.

II. Successive-approximation Coding of Vectors

A structured method for successive refinement of vectors was presented by Mukherjee and Mitra [5], [6], in which scaled versions of a given lattice are used as quantizers over each step of the approximation process. Voronoi region encoding is the basic operation in this framework, and it is performed according to the following concepts:

- Base lattice (Λ_1): lattice coset from which the codebook is actually derived
- Shape lattice (Λ_0): higher scale lattice which determines the shape of the codebook

The resulting quantizer, called *Voronoi Lattice Vector Quantizer*, is therefore defined as

$$\mathrm{VLVQ}(\Lambda_0, \Lambda_1) = V_0(\Lambda_0) \cap \Lambda_1 \tag{1}$$

where $V_0(\Lambda_0)$ is the zero-centered Voronoi region associated with the lattice. The shape lattice is defined so that it covers the n-dimensional region of support of the data source and, in the most common case, the base lattice is just a scaled down and (possibly) translated version of the shape lattice, i.e.,

$$\Lambda_1 = \frac{\Lambda_0}{r} - \mathbf{t}, \tag{2}$$

$\mathbf{t}$ being the translation vector.

Following a different approach, da Silva and Craizer [7] showed that successive approximation of vectors, under certain conditions, is guaranteed to converge in a finite amount of time. Formally, a vector $\mathbf{v}$ is said to be successively approximated by a sequence of codewords $\mathbf{u}_l$ if the summation

$$\mathbf{v} = \sum_{l=0}^{\infty} \alpha^l \mathbf{u}_l, \tag{3}$$
$$\mathbf{u}_l \in C = \{\mathbf{c}_0, \mathbf{c}_1, ..., \mathbf{c}_K\}$$

converges, where C is the codebook, $\mathbf{c}_k$ are the codewords and α is a scaling factor to account for the fact that after each interaction the residual error is bound by a smaller N-dimensional hypersphere. For every codebook - and codeword dimension - there is a choice (often empiric) of α that proves to be optimal, i.e., that provides the best representation results.

Since in lossy coding we are interested only in obtaining a close enough approximation of the original data, that is, with a limited amount of error, a finite summation is used instead of the infinite one, resulting in

$$\mathbf{v}_L = \sum_{l=0}^{L} \alpha^l \mathbf{u}_l. \tag{4}$$

In the proposed vector version of the SPECK algorithm, the above approximation is done by choosing, at each encoding pass, the one codeword that best represents the residual error between the original data and its current reconstructed version. For the experiments reported herein the codebooks were defined based on the 1^{st} and 2^{nd} shells of the D_4 lattice, with the codewords being properly normalized to unit length. Henceforth, we will refer to those codebooks as D_4 shell-1 and D_4 shell-2.

III. LVQ-SPECK

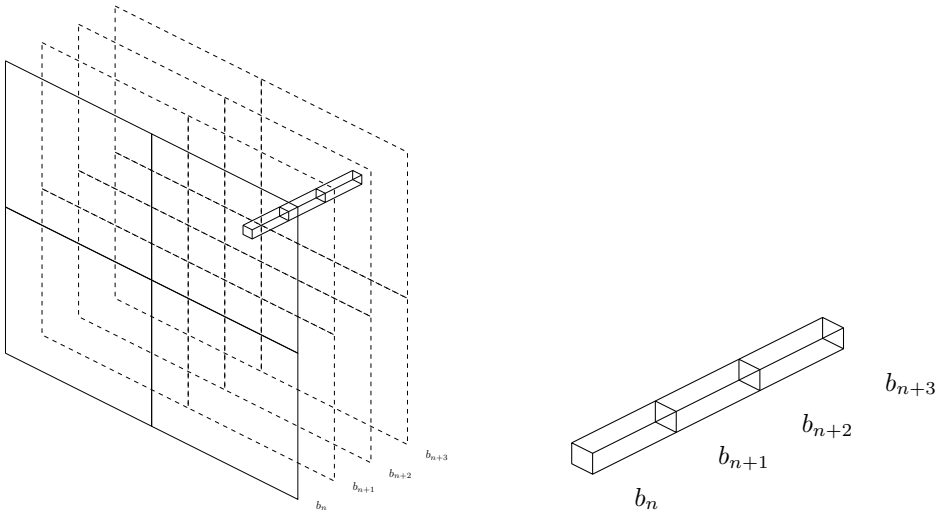
The proposed encoding method is based on the SPECK algorithm [4], and consists of a vector extension of its principles to account for the need to work with multi-dimensional samples. We will now present a description of the algorithm, pointing out the main differences between the vector and scalar case. For more details about the original (scalar) method, the reader is referred to [4].

LVQ-SPECK applies a DWT to each of the scalar bands, generating a group of adjacent data sets containing transform coefficients. We define a *Group Of Images* (GOI) as this set of adjacent transformed spectral bands b_i being encoded. Figure 1 shows how a vector sample $\mathbf{v}(x, y)$ is defined for a given GOI of dimension 4. Hence, for each spatial coordinate, we have

$$\mathbf{v}(x, y) = (b_n(x, y), b_{n+1}(x, y), b_{n+2}(x, y), b_{n+3}(x, y)), \tag{5}$$

where each component belongs to a distinct spectral band. Since we are now dealing with vector quantities, the significance measure used will be defined by comparing the vector's norm against the current encoding threshold T_n, that is

$$\Gamma_n(\mathcal{T}) = \begin{cases} 1, & \text{if } \max\limits_{(\mathrm{x,y})\in\mathcal{T}} \parallel \mathbf{v}(\mathrm{x,y}) \parallel \geq T_n \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$



(a) Group of Images (*spectral bands*)

(b) 4-dimensional vector sample $\mathbf{v}(x, y)$

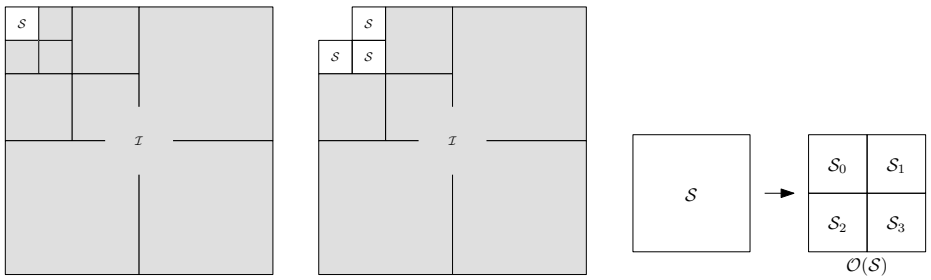
Figure 1. Spectral band Group of Images (GOI) for encoding

As in the scalar case, the initial threshold is defined based on the largest value to be encoded, which in this case is the largest norm among all the transform vectors. However, the threshold scaling rate is no longer restricted to $1/2$, as previously described in Section II.

The original SPECK algorithm defines two classes of partitioning sets, $\mathcal{S}$ and $\mathcal{I}$ (shown on Figure 2), used to convey the significance information of a group of samples. Initially, the $\mathcal{S}$ set is defined to be the set comprised of the low-low frequency sub-band coefficients of the wavelet transform, with the $\mathcal{I}$ set accounting for all remaining coefficients.

The encoding steps follow those of the original algorithm, with changes to account for the encoding of vector samples:

1) **Initialization:**
   - Partition image transform $\mathcal{X}$ into $\mathcal{S}$ and $\mathcal{I} = \mathcal{X} - \mathcal{S}$ sets.
   - The initial threshold T_0 and the threshold scaling factor α are transmitted.
   - Add $\mathcal{S}$ to the LIS, and set LSP $= \emptyset$.
2) **Sorting pass:**
   - for each set $\mathcal{S} \in$ LIS, and in increasing order of size $|\mathcal{S}|$, do `ProcessS`($\mathcal{S}$).
   - if $\mathcal{I} \neq \emptyset$, `ProcessI()`
3) **Refinement pass:**
   - for each (x, y) in the LSP, if the residual norm is larger than the current threshold, output the index of the codeword that best represents it. This procedure is the equivalent of adding a new term to the summation in Eq. 4. Otherwise, output the zero-codeword index, since there is no refinement to take place.
4) **Quantization step:**
   - update the encoding threshold, i.e., set $T_n = \alpha \times T_{n-1}$, and go to step 2.



(a) Initial $\mathcal{S}$ and $\mathcal{I}$ sets (b) $\mathcal{I}$-set partitioning (c) S-set partitioning

Figure 2. Set definition and partitioning for the SPECK algorithm

The procedures involved in the encoding/decoding process are defined as follows:

- `ProcessS(`$\mathcal{S}$`)`:
  1) output $\Gamma_n(\mathcal{S})$
  2) if $\Gamma_n(\mathcal{S}) = 1$
     - if $\mathcal{S}$ corresponds to a pixel, then output its codebook index and add $\mathcal{S}$ to the LSP
     - else `CodeS(`$\mathcal{S}$`)`
     - if $\mathcal{S} \in$LIS, then remove $\mathcal{S}$ from LIS
  3) else if $\mathcal{S} \notin$LIS, then add $\mathcal{S}$ to LIS
  4) return
- `CodeS(`$\mathcal{S}$`)`:
  1) partition $\mathcal{S}$ into four equal subsets $\mathcal{O}(\mathcal{S})$
  2) for each $\mathcal{S}_i \in \mathcal{O}(\mathcal{S})$
     - output $\Gamma_n(\mathcal{S}_i) = 1$
     - if $\Gamma_n(\mathcal{S}_i) = 1$
       - if $\mathcal{S}_i$ corresponds to a pixel, then output its codebook index and add $\mathcal{S}_i$ to the LSP
       - else `CodeS(`$\mathcal{S}_i$`)`
     - else add $\mathcal{S}_i$ to LIS
  3) return
- `ProcessI()`:
  1) output $\Gamma_n(\mathcal{I})$
  2) if $\Gamma_n(\mathcal{I}) = 1$, then `CodeI()`
- `CodeI()`:
  1) partition $\mathcal{I}$ into three sets $\mathcal{S}_i$ and one $\mathcal{I}$ (see Fig. 2)
  2) for each $\mathcal{S}_i$, do `ProcessS(`$\mathcal{S}_i$`)`
  3) `ProcessI()`
  4) return

Examination of the algorithm shows us that the encoding power of LVQ-SPECK stems from the fact that it sorts out those vectors with larger magnitude and immediately starts sending information about their spatial location and orientation on the n-dimensional hypersphere. Subsequent passes provide refinement information, further reducing the

reproduction distortion. It is also worth noticing that, as in the original (scalar) SPECK codec, the generated bit stream is still an embedded one.

Comparison with 3D transform coding

As discussed in [8], the performance of an image compression algorithm based on successive approximation of vectors is not only dependent on how well distributed in space the codewords are, but also on how well-matched those are to the source (n-dimensional) statistics in terms of correlation.

For each vector to be quantized, the better the first approximation step, the least amount of residue will remain and therefore a smaller number of refining steps will be needed, resulting in better overall performance. As such, in the present case, it is of paramount importance that the quantizer used during the sorting pass possesses characteristics that are similar to those of the source being compressed. During the remaining refinement steps, this requirement becomes less stringent since the residue's statistics tend to be more uniformly distributed over the n-dimensional space, making the packing properties of the lattice more important.

When one applies a unitary transform A across the spectral dimension, thereby inducing an orthonormal basis rotation, there generally is a reduction in the correlation among the components of the source vectors, but the euclidean distances are maintained due to the norm preserving property. Therefore, as far as the first approximation pass is concerned, matching the transformed (by A) source vectors to a codebook is equivalent to matching the original source vectors to an appropriately transformed (by A^{*T}) codebook.

In subsequent passes, since the residuals tend to have orientations evenly distributed over an hypersphere, what matters is just the relative orientation of the vectors in the codebooks [8], and therefore all codebook rotations have equivalent performance from the second pass on. This implies that the codebook of choice should be the one whose vectors best match the spectral features of the source. In the next section we validate the above assumption by comparing the performance of LVQ-SPECK with and without a transform (a 4-point DCT) in the spectral dimension using different rotated versions of a codebook.

IV. Experimental Results

The LVQ-SPECK algorithm was used to compress scenes of the AVIRIS hyperspectral images "Moffet Field" and "Jasper Ridge" (obtained from http://aviris.jpl.nasa.gov), both cropped to 512×512×224. A pre-processing step was added to remove all the zero-energy spectral bands from the hyperspectral block, with the indices of those bands being sent as (negligible) overhead. The spectral bands were then grouped into 4-dimensional blocks to be encoded.

The DWT kernel used was the 9/7 wavelet [10], and a 5-stage transform was applied to each spectral band. Bit allocation across sub-bands is done implicitly based on the significance of each vector being encoded. Each significance test accounts for one bit in the final bit-stream and, since both 4-dimensional codebooks used contain 24 vectors, in the worst case vector index transmission will demand $\log_2 24 = 4.59$ bits during the sorting pass and $\log_2 25 = 4.64$ bits during the refinement ones (to account for the zero codeword).

Table I
AVERAGE SNR (IN DB) FOR AVIRIS HYPERSPECTRAL IMAGES. (VALUES IN PARENTHESIS INDICATE AVERAGE RMSE)

Jasper Ridge (scene 01)				
Rate (bpppb)	0.1	0.2	0.5	1.0
3D-SPIHT[9]	19.59	23.59	31.48	38.36
3D-SPECK[9]	19.7	23.66	31.75	38.55
JPEG2000 Multi Component[9]	18.25	22.17	29.81	36.63
LVQ-SPECK (D4-sh1)	15.31 (244.04)	17.13 (201.94)	20.41 (143.66)	24.18 (96.16)
LVQ-SPECK (D4-sh1) + DCT	15.41 (241.98)	17.22 (200.98)	20.63 (140.85)	24.43 (94.33)
LVQ-SPECK (D4-sh2)	**17.72 (189.65)**	**20.39 (144.85)**	**25.65 (84.81)**	**31.61 (46.27)**
LVQ-SPECK (D4-sh2) + DCT	17.71 (190.10)	20.38 (145.16)	25.65 (85.51)	31.39 (47.96)
2D-SPECK	14.60 (262.04)	16.23 (221.09)	19.28 (161.13)	22.78 (111.84)

Moffet Field (scene 01)				
Rate (bpppb)	0.1	0.2	0.5	1.0
3D-SPIHT[9]	16.57	21.46	29.88	38.54
3D-SPECK[9]	16.67	21.52	29.91	38.60
JPEG2000 Multi Component[9]	15.29	19.92	28.19	36.56
LVQ-SPECK (D4-sh1)	18.46 (344.50)	20.15 (301.50)	23.35 (232.03)	27.18 (162.27)
LVQ-SPECK (D4-sh1) + DCT	18.47 (343.74)	20.32 (294.76)	23.52 (223.05)	27.30 (153.57)
LVQ-SPECK (D4-sh2)	**20.82 (283.46)**	**23.39 (232.48)**	**28.69 (156.40)**	**35.00 (97.61)**
LVQ-SPECK (D4-sh2) + DCT	20.80 (284.96)	23.33 (235.37)	28.58 (161.71)	34.67 (104.33)
2D-SPECK	17.76 (360.76)	19.47 (311.04)	22.45 (239.15)	25.86 (173.81)

Moffet Field (scene 03)				
Rate (bpppb)	0.1	0.2	0.5	1.0
3D-SPIHT[9]	12.92	18.25	27.28	35.62
3D-SPECK[9]	12.60	17.98	26.99	35.37
JPEG2000 Multi Component[9]	10.79	16.81	25.82	33.44
LVQ-SPECK (D4-sh1)	15.35 (327.59)	17.71 (278.08)	22.57 (200.61)	27.74 (138.41)
LVQ-SPECK (D4-sh1) + DCT	15.38 (324.37)	18.10 (266.58)	22.79 (191.01)	28.02 (127.46)
LVQ-SPECK (D4-sh2)	**18.72 (255.20)**	**22.51 (200.99)**	**29.40 (135.59)**	**35.16 (91.86)**
LVQ-SPECK (D4-sh2) + DCT	18.57 (260.46)	22.30 (207.35)	28.81 (145.90)	33.86 (103.44)
2D-SPECK	14.61 (339.25)	16.89 (283.77)	21.34 (202.63)	26.43 (139.21)

Table I presents a comparison among the reconstruction results for each of the hyperspectral blocks considered, when processed by LVQ-SPECK, the 3D-SPIHT and 3D-SPECK algorithms [9], Multi Component feature of JPEG2000 [11], and the original 2D-SPECK codec applied to each of the spectral bands individually. The figure of merit utilized here is the signal-to-quantization noise ratio (SNR), defined as

$$\mathrm{SNR} = 10 \log_{10} \frac{P_x}{\mathrm{MSE}} \,\mathrm{dB} \tag{7}$$

where P_x is the power of the original signal and MSE is the reproduction mean-squared error. Table I contains RMSE results for our simulations as well.

Based on Table I, we see that the performance attained by the LVQ-SPECK algorithm is quite competitive, especially at low bit rates, where it outperforms by a large margin even 3D-based codecs, when applied to scenes of the Moffet Field image. That is, in fact, quite impressive, considering that in the case of 3D algorithms, the decorrelating transform across the spectral direction has length 16, compared to a vector of dimension

4 in our case. It also shows that, in the early phases of the approximation process, LVQ-SPECK does a better job of representing the transformed data. As encoding rates go up, however, 3D-based codecs generally present better results, given the high degree of energy compaction provided by the additional transform. One possible way of improving the performance of LVQ-SPECK would be to process a larger number of adjacent spectral bands together and use higher dimensional lattices, with better packing properties, such as the Λ_{24} lattice [12]. That is quite important, since during the refinement steps the residues to be encoded are of a highly uncorrelated nature, as discussed in Section III.

It is also clear from Table I that simultaneously encoding a group of spectral bands using LVQ-SPECK provides much better results than the individual compression of each one of them. For instance, for a rate of 1.0 bpp, there is an approximate gain of 10dB in SNR for all the images tested. It is worth mentioning that, for all studied images, the codebook based on the D4 shell-2 lattice was the one that provided the best performance. However, we may also say, based on the presented results, that the chosen rotation of the lattice codebook was suitable for the Moffet image, but perhaps not so suitable for Jasper, which shows us that this method is quite promising, provided one uses the proper lattice rotation.

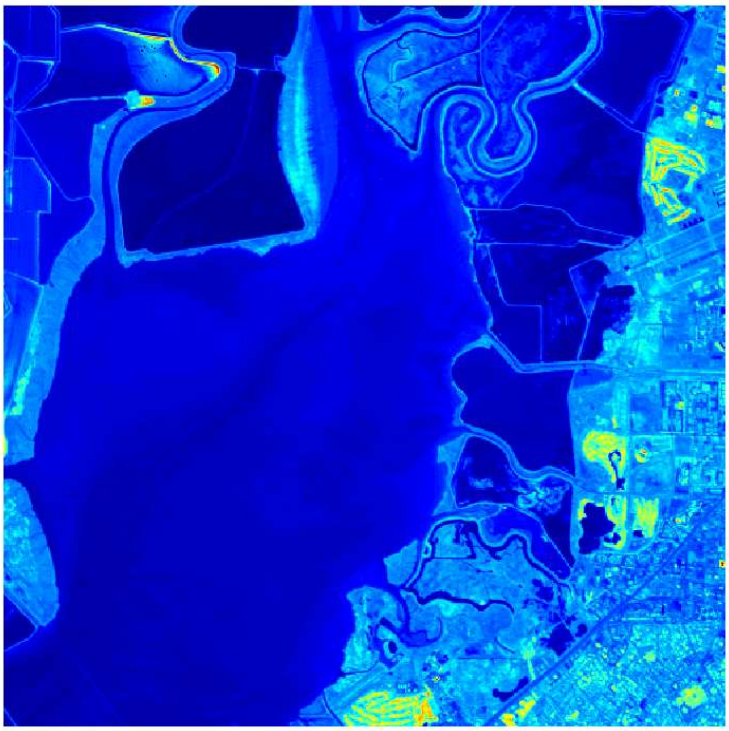
Figure 3 presents a visual comparison between the original data and a reconstructed version at 0.2 bpppb, for spectral band 48 of the Moffet Field image, Scene 03, with the points from D_4 shell-2 lattice serving as codebook.

V. Conclusions

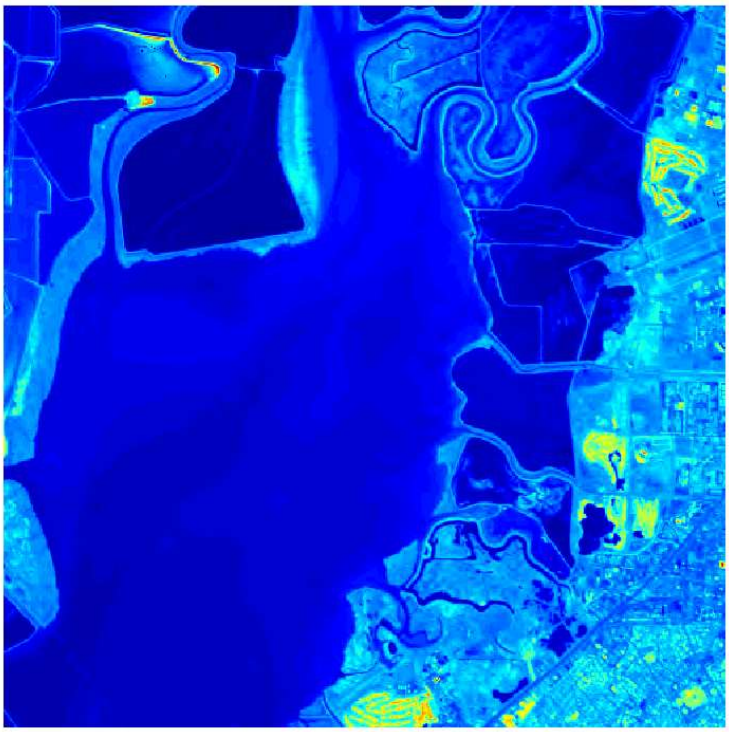
We presented a new multidimensional image codec based on the SPECK state-of-the-art algorithm, that makes use of a lattice vector quantizer codebook, suited to the encoding of hyperspectral images. We showed that the proposed algorithm, using a 2D DWT kernel applied independently to each spectral band, attains performance comparable to those methods that employ a 3D transform and clearly outperforms the alternative of separately encoding each spectral band. Moreover, for low bit rates, LVQ-SPECK produces reproduction results that are, in some cases, overwhelmingly better than those of 3D encoding methods.

The importance of choosing the right codebook was discussed by comparing equivalent codebooks, where one is a rotated version of the other. It was verified that the version of the codebook which more closely matches the source characteristics over the sorting pass is the one yielding best performance results. This properly rotated codebook produced performance comparable to 3D wavelet coders.

Further improvements are expected with the simultaneous encoding of a larger number of spectral bands and the use of higher-dimensional lattices (e.g., E_8, Λ_{16} and Λ_{24}) as the basis for the codebook, and methods of fast computation to determine the best rotation angle to define the codebook.



(a) Original Data



(b) Reconstructed at 0.2 bpppb

Figure 3. Moffet Field, Scene 03

ACKNOWLEDGMENTS

This work was performed at Rensselaer Polytechnic Institute and was supported in part by Fundação CAPES, Brazil, under Grant No. 1535-98/6.

REFERENCES

[1] G. Motta, F. Rizzo, and J. A. Storer, "Compression of Hyperspectral Imagery," in *Proceedings of the Data Compression Conference*, pp. 333–342, March 2003.

[2] X. Tang, W. A. Pearlman, and J. W. Modestino, "Hyperspectral image compression using three-dimensional image coding," in *SPIE/IS&T Electronic Imaging 2003*, vol. 5022 of *Proceedings of the SPIE*, Jan. 2003.

[3] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*. Kluwer Academic Publishers, 1991.

[4] W. A. Pearlman, A. Islam, N. Nagaraj, and A. Said, "Efficient, low-complexity image coding with a set-partitioning embedded block coder," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 14, pp. 1219–1235, 2004.

[5] D. Mukherjee and S. K. Mitra, "Successive refinement lattice vector quantization," *IEEE Transactions on Image Processing*, vol. 11, pp. 1337–1348, Dec. 2002.

[6] D. Mukherjee and S. K. Mitra, "Vector SPIHT for embedded wavelet video and image coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, pp. 231–246, Mar. 2003.

[7] E. A. B. da Silva and M. Craizer, "Generalized bit-planes for embedded codes," in *Image Processing, 1998. ICIP 98. Proceedings. 1998 International Conference on*, vol. 2, pp. 317–321 vol.2, 1998.

[8] L. H. Fonteles, R. Caetano, and E. A. B. da Silva, "Improved dictionaries for generalized bitplanes-based matching pursuits video coding using ridgelets," in *Image Processing, 2004. ICIP '04. 2004 International Conference on*, vol. 2, pp. 1129–1132 Vol.2, 2004.

[9] X. Tang and W. A. Pearlman, *Three-Dimensional Wavelet-Based Compression of Hyperspectral Images*, ch. Hyperspectral Data Compression. Kluwer Academic Publishers, 2006.

[10] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies, "Image coding using wavelet transform," *IEEE Transactions on Image Processing*, vol. 1, pp. 205–220, April 1992.

[11] D. S. Taubman and M. W. Marcellin, *JPEG2000 Image Compression: Fundamentals, Standards and Practice*. (The International Series in Engineering and Computer Science), Kluwer Academic Publishers, 2002.

[12] J. H. Conway and N. J. A. Sloane, *Sphere Packings, Lattices and Groups (Grundlehren der mathematischen Wissenschaften)*. Springer, 3rd ed., 1998.