A Block-based Inter-band Lossless Hyperspectral Image Compressor

Marko Slyz, Lei Zhang[1]

Abstract: We propose a hyperspectral image compressor called BH which considers its input image as being partitioned into square blocks, each lying entirely within a particular band, and compresses one such block at a time by using the following steps: first predict the block from the corresponding block in the previous band, then select a predesigned code based on the prediction errors, and finally encode the predictor coefficient and errors. Apart from giving good compression rates and being fast, BH can provide random access to spatial locations in the image.

We hypothesize that BH works well because it accommodates the rapidly changing image brightness that often occurs in hyperspectral images. We also propose an intra-band compressor called LM which is worse than BH, but whose performance helps explain BH's performance.

1 Introduction

Each location in a hyperspectral image records the intensity of light received at many different frequencies, both visible and invisible. These many frequencies make identifying the imaged substances easier [1], but increase the data size.

Several lossless hyperspectral image compressors compress a pixel by first predicting it from pixels in other bands. In order to account for variations in image statistics, the methods typically use more than one predictor. They can be roughly classified according to where they use any given predictor:

on an entire band [2] is an example which also shows a principled way for choosing the order in which to process bands.

on a subset of a band [3] uses each predictor on a row of pixels, [4] uses each predictor on a set of square blocks, and [5] uses each predictor on an irregularly-shaped (and usually not connected) region that lies within a single band.

on a single pixel [6] transmits side-information to indicate which of a small set of predictors to use for any given pixel, while [7] and [8] design their predictors anew for each pixel.

[1]Marko Slyz was with the Department of Electrical and Computer Engineering, McMaster University, Hamilton, ON, Canada. He is now with Bitfone Corporation, Laguna Niguel, CA, USA. Lei Zhang is with the Department of Electrical and Computer Engineering, McMaster University, Hamilton, ON, Canada. (email: {mslyz,johnray}@mail.ece.mcmaster.ca) This work was supported in part by CRESTech and PCI Geomatics.

Authorized licensed use limited to: Hong Kong Polytechnic University. Downloaded on September 10, 2009 at 04:02 from IEEE Xplore. Restrictions apply.

Other approaches to hyperspectral image compression are vector quantization and transform coding; [5, §2] briefly surveys many of these. We believe that linear predictive coding works well enough to be worth studying on its own, and often times more-complicated methods use linear predictive coding as one of their parts, e.g. [5].

One of our contributions is a compressor called BH which belongs to the subset-of-a-band category. BH uses each predictor on a single square block, and predicts only from the corresponding block in the adjacent previous band. At least the predictive part of BH is basically a variant of [3] and [7], as Section 2 discusses.

BH has a useful combination of several good properties: it gets good compression rates; requires only about twice the time for compressing or decompressing as merely reading or writing and byte-order swapping the original image; is relatively easy to implement; could be implemented to compress the different spatial regions in the image independently so that random access to them is easy; and could be implemented to require the buffering of relatively small portions of the image—maybe 4.3 Mb while compressing, and 112 Kb while decompressing.

Another of our contributions is a hypothesis—based on an observation about hyperspectral image statistics—for why BH and similar methods work well. Finally, we propose a compressor, called LM, that is in part based on this same observation but that does only intra-band prediction. Overall LM is worse than BH, but LM's performance helps explain BH's performance.

We'll cover these subjects in the following order: First we'll describe how BH and LM work, then the proposed physical model that helps explain why they work, and finally, experimental results supporting our claims.

2 The BH Compression Algorithm

Let x be an arbitrary hyperspectral image, and let $x_{i,j,k}$ denote the pixel in the ith line, jth column, and kth band. In this paper, wavelength increases with k.

Suppose that each of the image's bands is partitioned into blocks which are usually square and B pixels on a side, except that near the image's edges the blocks might be smaller. Suppose also that the borders of each band's blocks line up. Let $\underline{x}_{i,j,k}$ be a column vector containing the pixels in the block whose top left corner is at (i, j) in band k. The exact order in which pixels are placed into $\underline{x}_{i,j,k}$ doesn't matter except that it must always be the same.

BH compresses one block at a time. It compresses the stack of blocks covering a given spatial location all at once, in order of increasing k. When compressing a block in the first band, $\underline{x}_{i,j,0}$, the algorithm assumes that the previous block, call it $\underline{x}_{i,j,-1}$, is all zeros. To compress $\underline{x}_{i,j,k}$ the algorithm

1. computes the number g that makes g times $\underline{x}_{i,j,k-1}$ as similar as possible in the least-squares sense to $\underline{x}_{i,j,k}$, namely

$$g \overset{def}{=} \frac{\underline{x}^T_{i,j,k-1}\underline{x}_{i,j,k}}{\underline{x}^T_{i,j,k-1}\underline{x}_{i,j,k-1}} \tag{1}$$

IEEE COMPUTER SOCIETY

Authorized licensed use limited to: Hong Kong Polytechnic University. Downloaded on September 10, 2009 at 04:02 from IEEE Xplore. Restrictions apply.

2. produces a quantized version of g, called $\hat{g}$, with a uniform scalar quantizer that has 1024 levels stretching from 0.1 to 3;

3. computes the error vector

$$\underline{e}_{i,j,k} \overset{def}{=} \hat{g}\underline{x}_{i,j,k-1} - \underline{x}_{i,j,k} \tag{2}$$

4. computes the mean absolute error,

$$M \overset{def}{=} \text{average}\{|\underline{e}_{i,j,k}|\} \tag{3}$$

5. applies to M a nonuniform scalar quantizer each of whose bins corresponds to a different precomputed entropy code (Section 3 describes how to choose the quantizer and codes);

6. transmits $\hat{g}$, transmits the bin number from Step 5 and, using the code from Step 5, transmits the elements of $\underline{e}_{i,j,k}$.

BH uses the same quantizer and set of entropy codes for every image, and doesn't adapt them as it runs. To do decompression, BH simply receives $\hat{g}$ and the bin number, then uses the code corresponding to that bin number to uncompress the elements of $\underline{e}_{i,j,k}$, and finally sets $\underline{x}_{i,j,k}$ equal to $\hat{g}\underline{x}_{i,j,k-1} - \underline{e}_{i,j,k}$.

As mentioned, BH is particularly similar to [3], especially to [3]'s SE-o1B predictor which predicts each pixel from its corresponding pixel in the previous band plus an affine term. The major difference is that BH uses each predictor on a spatially adjacent region of pixels which are likely to have similar characteristics, and not on a very elongated region of pixels which aren't. BH is also particularly similar to [7], differing in that BH doesn't do a predictor design for every single pixel. Finally, many video compressors are similar to BH in that they compress blocks in the current frame (band) by referencing blocks in the previous one. Some differences are that BH doesn't compensate for motion since there shouldn't be any, and that BH first multiplies the reference block by a scalar before subtracting it from the current block. [9] examined the use of standard video compressors for lossy hyperspectral compression.

Excluding Step 6, BH requires about eight arithmetic operations per pixel. Specifically, on a per block basis, Step 1 requires about $4B^2 - 1$ operations, Steps 3 and 4 each require about $2B^2$ operations and, for our quantizer implementations, Steps 2 and 5 require about 7 and 25 operations respectively on average. This doesn't include memory access, coding, and I/O times, which seem to be very significant, but because counting their operations is more difficult we instead refer the reader to Section 6's experimental results.

BH's random access capability comes about because the compression of a stack of blocks corresponding to a particular spatial region, $\{\underline{x}_{i,j,0}, \ldots, \underline{x}_{i,j,\beta-1}\}$, where β is the number of bands in the image, doesn't depend on pixels outside this stack.

BH's small buffering capability comes about because during decompression BH needs to keep just the data corresponding to a single stack in memory. Compression also works on only a single stack at a time, but at least for frequency-sequential storage

Authorized licensed use limited to: Hong Kong Polytechnic University. Downloaded on September 10, 2009 at 04:02 from IEEE Xplore. Restrictions apply.

orders (like in [10]'s images) reading a whole row of stacks at once is convenient. [7] could also be made to use small buffers by storing only the rows of the image that are necessary for prediction, but their A&P methods would need to be redesigned.

3 The Training Process

The training process to find BH's quantizer and set of entropy codes is as follows: run the compression algorithm's first four steps on some training data; collect the resulting values of M and $\underline{e}_{i,j,k}$; sort the values of M, partition them into some number n of contiguous equinumerous groups, and record the maximum (or minimum) values of M in every group as the quantizer thresholds; design a code for each group's errors; and, finally, store the quantizer and codes with the rest of BH—in the current implementation they're compiled into the executable. Section 6's experiments all use a version of BH trained on the M and $\underline{e}_{i,j,k}$ values from 18068 blocks, each 16×16 pixels in size except near the image's edges, that were selected from random locations in the Moffet Field radiance image [10].

We pick n and the block size B by trial-and-error. This is straightforward because the algorithm's performance doesn't seem very sensitive to the choices for n and B once they reach a particular range of values. Figure 1 shows an example of this for block size. Here the overall compression rate doesn't change much for blocks of size 8×8 up to blocks of size 256×256. The values we use in Section 6's experiments are $B = 16$ and $n = 25$.

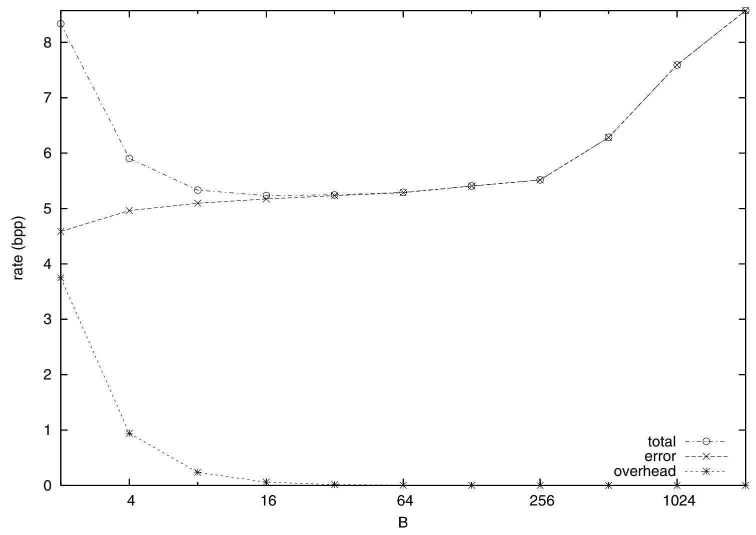

Figure 1: The total compression rate (top line) on the Cuprite image [10], the rate for just the errors (middle line), and for just the overhead (bottom line). The quantizer and entropy codes are kept fixed to the Moffet Field-trained ones described above.

[4]'s use of an equiprobable quantizer is similar to BH's, while [11, §2.2] uses such quantizers in a somewhat less related way. Other kinds of quantizers likely would work as well.

Authorized licensed use limited to: Hong Kong Polytechnic University. Downloaded on September 10, 2009 at 04:02 from IEEE Xplore. Restrictions apply.

The current BH implementation uses canonical minimal redundancy codes as described in [12], using their Table-Lookup method with $x = 8$ for decompression. Code lengths for this design essentially come from the Kraft-coder algorithm with tree filling that [13] describes, although the ordinary Huffman algorithm would have worked just as well.

4 The LM Compression Algorithm

Let $\underline{w}_{i,j,k}$, the *context*, be a vector of pixels that are all in band k, and that are taken from some fixed set of locations near i, j. A simple compressor, call it LMi, predicts $x_{i,j,k}$ from its context $\underline{w}_{i,j,k}$, using a predictor designed to be good at predicting the previous few pixels at location i, j from the corresponding contexts in each of their bands. LMi is basically a backward-adaptive version of [3], with the compression proceeding first in the frequency direction. A more sophisticated compressor, call it LM, trains several predictors each using a different kind of context, and then selects the predictor that worked best on the past few pixels.

A more detailed description of LM is as follows. Let u vary from 1 to some number r. Let $\underline{w}(u)_{i,j,k}$ be one of r different contexts, chosen so that contexts with the same index u but from different bands—e.g. $\underline{w}(u)_{i,j,k}$ and $\underline{w}(u)_{i,j,m}$ for m not equal to k—take their pixels from the same spatial locations and in the same order. Also let u index a set of positive integers $\{M_u\}$. Then to code the pixel at $x_{i,j,k}$, LM

1. for each u forms and solves the equation

$$\begin{bmatrix} \underline{w}(u)_{i,j,k-1} \\ \vdots \\ \underline{w}(u)_{i,j,k-M_u} \end{bmatrix} \underline{a}_u \approx \begin{bmatrix} x_{i,j,k-1} \\ \vdots \\ x_{i,j,k-M_u} \end{bmatrix} \tag{4}$$

   for $\underline{a}_u$ in the least-squares sense.

2. picks the $\underline{a}_u$ that minimizes the error in predicting some number N of past pixels, i.e. that minimizes

$$\left\| \begin{bmatrix} \underline{w}(u)_{i,j,k-1} \\ \vdots \\ \underline{w}(u)_{i,j,k-N} \end{bmatrix} \underline{a}_u - \begin{bmatrix} x_{i,j,k-1} \\ \vdots \\ x_{i,j,k-N} \end{bmatrix} \right\| \tag{5}$$

3. uses this $\underline{a}_u$ to predict $x_{i,j,k}$, and selects a Golomb code based on (5) as in [14]. A slightly more flexible approach is to allow N to be different for this code selection than it was for the $\underline{a}_u$ selection.

The decompressor can do these same steps since all of them depend only on pixels that it has already seen.

IEEE COMPUTER SOCIETY

Authorized licensed use limited to: Hong Kong Polytechnic University. Downloaded on September 10, 2009 at 04:02 from IEEE Xplore. Restrictions apply.

The motivation for LMi and LM is the idea that the nearby spectra, or maybe some weighted average of them, are similar to the current one, and that the difficulty is picking which weighted average to use.

It might seem at first that LM isn't any more powerful than LMi, since one could choose to let LMi's context vector contain the pixels in the union of all of LM's contexts, and then least-squares would effectively pick a good subset. Unfortunately a large context like that one would require more training data, but this data might not be available if the image statistics change near i, j, k. [15]

5 A Hypothesis About Why BH Works

BH does well on a particular model which seems to fit hyperspectral images at least to some extent. The model says that the spectra near any arbitrary location i, j in the image are each the product of a constant *underlying radiance spectrum* $\underline{v}$, and a scalar *brightness factor* $c_{i,j}$. In other words,

$$\underline{x}_{i,j} \approx c_{i,j}\underline{v} \quad \text{for} \quad (i,j) \in R \tag{6}$$

where R is a relatively small region of the imaged area. A rationale is that the $c_{i,j}$ takes into account rapidly varying brightness changes due to changes in terrain inclination and possibly other factors [1, p.12], leaving the components of the spectrum that vary slowly—like the substance being imaged, the atmospheric effects, and the spectral properties of the illumination—in $\underline{v}$.

Visual inspection of groups of spectra selected from randomly-chosen locations in some of the [10] images helps support this model, though this is somewhat subjective. Figure 2 shows a common situation where adjacent spectra from a region in the Jasper image follow (6) in that they have essentially the same shape, and vary mainly in brightness. That's at least the main trend; a point against this, which is observable in the color version of Figure 2, is that the spectra occasionally change their relative positions, especially near sharp transitions. The other common situation is when spectra taken from a small region change their separation with increasing wavelength, still having peaks and troughs in the same places, but it seems that these changes are usually gradual enough that the model is still approximately correct over small intervals of bands.

The standard Spectral Angle Mapper (SAM) algorithm [16, p.12-13] is also essentially based on the idea of different brightness factors multiplying an underlying spectrum (SAM ignores these brightness factors), although SAM doesn't use the idea that spatially nearby spectra are similar.

Now (6) is interesting because BH and similar methods can perfectly predict pixels that follow it. To see this, let $\underline{x}_{i,j,k}$ be a block of size $B \times B$ at an arbitrary location in the image; let g be the minimizer, as in (1), of $\|g\underline{x}_{i,j,k-1} - \underline{x}_{i,j,k}\|_2$; and let $\underline{c}$ be a vector having B^2 elements that contains the brightness factors for each of the B^2 spectra in the block. If (6) holds for the block, with R containing the block's B^2 locations, then

$$\|g\underline{x}_{i,j,k-1} - \underline{x}_{i,j,k}\|_2 = \|g\underline{c}v_{k-1} - \underline{c}v_k\|_2 \tag{7}$$

IEEE COMPUTER SOCIETY

Authorized licensed use limited to: Hong Kong Polytechnic University. Downloaded on September 10, 2009 at 04:02 from IEEE Xplore. Restrictions apply.

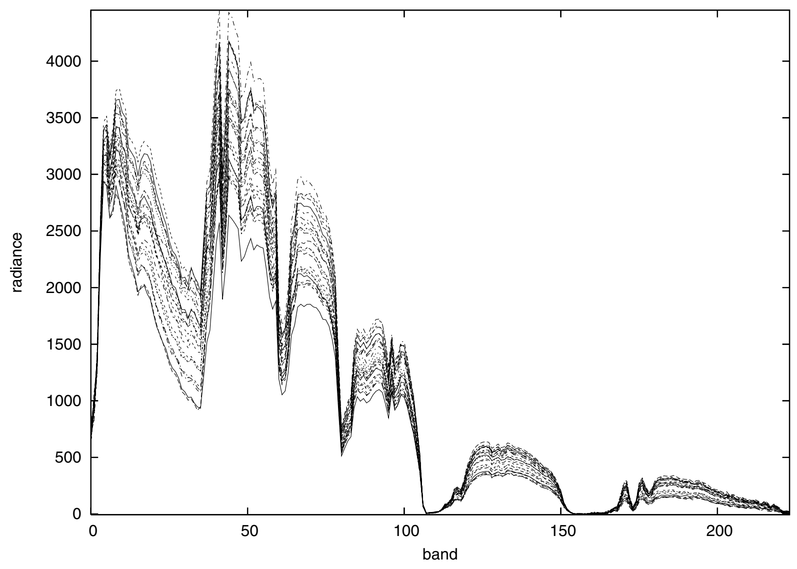

Figure 2: A set of 25 image spectra from near location 320,1020 in Jasper.

where v_{k-1} and v_k are the $k-1$st and kth elements of the underlying radiance spectrum $\underline{v}$. Now by choosing g to equal v_k/v_{k-1}, (7) will be zero.

Instead of predicting across frequency bands like BH, LM predicts a spectrum from an adaptively weighted combination of its neighbors. This ought to accommodate (6), but it should also be able to accommodate more significant differences. One disadvantage of LM is that, to the extent the model is true, BH has many data samples ($B \times B$ different pixel pairs) to do its estimation while LM typically has fewer.

Finally, Figure 1 suggests, as does direct examination of the underlying spectra, that R is not necessarily all that small. The spectra appear similar to some extent for even quite large blocks, although for large blocks there are more cases of spectra having significant differences that the model doesn't even approximately account for.

6 Experimental Results

Table 1 shows compression rates on the raw AVIRIS radiance images from [10]. The image names are listed in the left-hand column, with the "AVG\l.a." row denoting the average excluding the Low Altitude image.

BG is a version of BH that uses Golomb coding, selecting the coding parameter for each block based on the block's errors as in [14]. LM (1/1) uses two first-order predictors for every pixel—one predicting from the spectrum at $i-1, j$, and one from the spectrum at $i, j-1$—and trains these on the previous two image bands. LM (5/1) also used two predictors, a fifth order one and a first order one, training them on the previous 30 and 2 image bands respectively. We chose these particular parameters as the best performing ones out of many candidates.

The remaining results are from the literature. [5] uses vector quantization to clas-

Authorized licensed use limited to: Hong Kong Polytechnic University. Downloaded on September 10, 2009 at 04:02 from IEEE Xplore. Restrictions apply.

sify the image spectra, but then uses inter-band linear prediction. LPVQ's predictor [17] is based on vector quantization and isn't linear. It's somewhat similar to LM in that it predicts within bands, and that the predictions in one band indirectly influence the predictions in others. SLSQ-OPT[8] uses linear prediction based both on pixels within and outside the current band. The standardized JPEG-LS's results are from [17].

[7] doesn't generally have results broken down by image, but does quote 4.95 bits per pixel as their best rate on the set excluding Low Altitude.

image	BH	BG	LM (1/1)	LM (5/1)	[5]	LPVQ	SLSQ-OPT	JPEG-LS
cuprite	5.12	5.15	5.49	5.28	4.68	5.11	4.94	7.66
jasper	5.23	5.25	5.51	5.42	4.62	5.67	4.95	8.38
low altitude	5.52	5.52	5.77	5.60		5.54	5.26	8.00
lunar lake	5.12	5.15	5.48	5.26	4.75	4.95	4.95	7.48
moffet	5.26	5.26	5.48	5.35	4.62	5.44	4.98	8.04
AVG	5.25	5.27	5.55	5.38		5.34	5.02	7.91
AVG\l.a.	5.18	5.20	5.49	5.33	4.67	5.29	4.96	7.89

Table 1: Comparison of Compression Rates. All values are in bits per (16-bit) pixel.

Table 2 shows BH's compression and decompression speeds. It also has columns for NULL, a trivial compressor that just writes (for "compression") and reads (for "decompression") the image in its raw format, only doing conversion from big-endian to little-endian order. We did these experiments on a commodity PC with a 2.53GHz Pentium 4 with 753Mb of RAM, and an IDE hard drive with DMA turned on. In implementing BH we did pay some attention to run-time—for example, by using our own, simpler, I/O buffering rather than the one that comes with `getchar()` and `putchar()`.

The compression speed on Low Altitude seems anomalous. This probably has to do with memory paging, which is necessary for Low Altitude and not for the others because the current BH implementation buffers the entire image, and only in Low Altitude's case won't this buffer fit into core memory.

image	size	BH		NULL	
		Ct	Dt	Ct	Dt
cuprite	579	58	51	29	22
jasper	678	68	64	34	29
low altitude	968	455	247	49	54
lunar lake	375	41	33	19	15
moffet	533	51	48	27	23

Table 2: The size of each image in megabytes, and the times in seconds for compressing (in the Ct columns) and decompressing (in the Dt columns).

The current, not optimized implementations of LM take about 1030 seconds to compress Cuprite in the (5/1) case, and about 590 seconds in the (1/1) case. Decompression times are similar.

IEEE COMPUTER SOCIETY

Authorized licensed use limited to: Hong Kong Polytechnic University. Downloaded on September 10, 2009 at 04:02 from IEEE Xplore. Restrictions apply.

[7]'s best method (as quoted in Table 1) compresses at 1150 Kb/s, and decompresses at 2860 Kb/s on an 800 MHz Pentium-III. Multiplying these speeds by 3 to account for the differences in machine speed, which is optimistic, gives about 172 seconds to compress Cuprite and about 70 seconds to decompress it. A different variant in [7] that gets 5.10 bits/pixel would similarly require about 50 seconds to compress Cuprite, and about 51 seconds to decompress it. [5] says that their method requires 1180 seconds, also on an 800 MHz PC, to compress an image. Although [5] doesn't give a time for decompression, it's likely a lot faster since there's no VQ design then.

The other papers don't provide specific run times. LPVQ would seem to have slow compression since it requires multiple unconstrained VQ designs. But LPVQ's forward-adaptive nature likely makes decompression about as fast as BH's. SLSQ-OPT seems like it would take a long time for both compression and decompression since it needs to design an order-19 linear predictor for each pixel in either case.

7 Discussion

BH's average compression rate is about 11% worse than [5]'s, the lowest-rate lossless hyperspectral compressor that we're aware of, but BH ought to be a lot faster. BH also gets worse compression than SLSQ-OPT, but again ought to be much faster.

BH's compression rates are more similar to [7]'s, especially to their faster methods. It's not surprising that [7] compresses better, because [7] designs a different predictor for every pixel, and not just for groups of them like BH; but for this same reason [7] should be significantly slower and it doesn't seem to be. Some possible reasons for this are the overly-optimistic adjustment for clock speed differences mentioned above, that [7] codes some of the bands without prediction, and that I/O and memory access is so significant—taking up at least 1/3 to 1/2 of the time in BH's case according to Table 2—that [7]'s extra few operations per pixel don't matter much.

The major difference between BH and [5] is that [5] can accommodate spatial differences in image statistics on a finer scale, since it groups together similar spectra before doing prediction, while BH just uses a fixed partition into blocks. The relatively small gap between BH's and [5]'s performance seems to support the idea that most of the ability to compress comes from being able to accommodate differences in brightness. LM's performance seems consistent with this as well since, as mentioned in Section 5, LM can also accommodate spatial differences in radiance spectra on a finer scale than BH, but in fact LM does a little worse.

Our final observation is that BG gets only about 1% worse compression than BH. Implementing Golomb codes in a straightforward way is easier than implementing canonical codes, but canonical codes can be somewhat faster.

Acknowledgements

We're grateful to Xiaolin Wu for guiding us and helping us to design the compression algorithms, and to Kristi Markham, William Morris, and Richard Pollock for helpful discussions.

Authorized licensed use limited to: Hong Kong Polytechnic University. Downloaded on September 10, 2009 at 04:02 from IEEE Xplore. Restrictions apply.

References

[1] R. B. Smith, “Introduction to hyperspectral imaging.” `http://www.microimages.com/getstart/hyprspec.htm`, 2001.

[2] S. Tate, “Band ordering in lossless compression of multispectral images,” *IEEE T. on Computers*, vol. 46, pp. 477–83, April 1997.

[3] R. Roger and M. Cavenor, “Lossless compression of AVIRIS images,” *IEEE T. on Image Processing*, vol. 5, pp. 713–19, May 1996.

[4] B. Aiazzi, L. Alparone, and S. Bartoni, “Near-lossless compression of 3-d optical data,” *IEEE T. on Geoscience and Remote Sensing*, vol. 39, pp. 2547–2557, November 2001.

[5] J. Mielikainen and P. Toivanen, “Clustered DPCM for the lossless compression of hyperspectral images,” *IEEE T. on Geoscience and Remote Sensing*, vol. 41, pp. 2943–2946, December 2003.

[6] N. Memon, K. Sayood, and S. Magliveras, “Lossless compression of multispectral image data,” *IEEE T. on Geoscience and Remote Sensing*, vol. 32, pp. 282–89, March 1994.

[7] J. Mielikainen, P. Toivanen, and A. Kaarna, “Linear prediction in lossless compression of hyperspectral images,” *Optical Engineering*, vol. 42, pp. 1013–1017, April 2003.

[8] F. Rizzo, B. Carpentieri, G. Motta, and J. Storer, “High performance compression of hyperspectral imagery with reduced search complexity in the compressed domain,” in *Data Compression Conference*, pp. 479–88, 2004.

[9] T. Wilkinson and V. Vaughn, “Application of video-based coding to hyperspectral imagery,” in *Hyperspectral Remote Sensing and Applications* (S. Shen, ed.), vol. 2821, pp. 44–54, SPIE, 1996.

[10] JPL, “AVIRIS free standard data products.” http://aviris.jpl.nasa.gov/html/aviris.freedata.html, 1997.

[11] M. Weinberger, G. Seroussi, and G. Sapiro, “LOCO-I: A low complexity, context-based, lossless image compression algorithm,” in *Data Compression Conference*, pp. 140–149, 1996.

[12] A. Moffat and A. Turpin, “On the implementation of minimum redundancy prefix codes,” *IEEE T. on Communications*, vol. 45, pp. 1200–1207, October 1997.

[13] M. Liddell and A. Moffat, “Length-restricted coding using modified probability distributions,” in *Proceedings of the 24th Australasian Conference on Computer Science*, (Gold Coast, Queensland, Australia), pp. 117–124, IEEE, 2001.

[14] M. Slyz and D. Neuhoff, “Some simple parametric lossless image compressors,” in *Proceedings of the IEEE International Conference on Image Processing*, vol. I, (Vancouver), pp. 124–7, September 2000.

[15] H. Ye, G. Deng, and J. Devlin, “Adaptive linear prediction for lossless coding of greyscale images,” in *Proceedings of the IEEE International Conference on Image Processing*, (Vancouver), September 2000.

[16] F. Kruse, A. Lefkoff, J. Boardman, K. Heidebrecht, A. Shapiro, P. Barloon, and A. Goetz, “The spectral image processing system (SIPS)—interactive visualization and analysis of imaging spectrometer data,” *Remote Sensing of Environment*, vol. 44, pp. 145–163, May-June 1993.

[17] G. Motta, F. Rizzo, and J. Storer, “Compression of hyperspectral imagery,” in *Data Compression Conference*, pp. 333–42, 2003.

IEEE COMPUTER SOCIETY

Authorized licensed use limited to: Hong Kong Polytechnic University. Downloaded on September 10, 2009 at 04:02 from IEEE Xplore. Restrictions apply.