COMPRESSION OF SPECTRAL METEOROLOGICAL IMAGERY

Kristo Miettinen
General Electric, Astro–Space Division
Valley Forge, Pennsylvania

Abstract. Data compression is essential to current low–earth–orbit spectral sensors with global coverage, *e.g.* meteorological sensors. Such sensors routinely produce in excess of 30 Gb of data per orbit (over 4 Mb/s for about 110 min.) while typically limited to less than 10 Gb of downlink capacity per orbit (15 minutes at 10 Mb/s). Astro–Space Division develops spaceborne compression systems for compression ratios from as little as three to as much as twenty–to–one for high–fidelity reconstructions. Current hardware production and development at Astro–Space Division focuses on discrete cosine transform (DCT) systems implemented with the GE PFFT chip, a 32x32 2D–DCT engine. Spectral relations in the data are exploited through block mean extraction followed by orthonormal transformation. The transformation produces blocks with spatial correlation that are suitable for further compression with any block–oriented spatial compression system, *e.g.* Astro–Space Division's Laplacian modeler and analytic encoder of DCT coefficients.

1.0 Introduction

Image compression typically consists of two steps: decorrelation and encoding. For spectral imagery a flexible and practical approach to compression is to transform the *n*–band spectral data into *n* spatially correlated but spectrally mostly decorrelated channels, using some approximation to the spectral Karhunen–Loève transform (KLT). Then the channels can be compressed using any spatial compression technique. The DCT is probably the most effective practical spatial decorrelator for image compression, but it produces a data stream which is difficult to encode due to its severe cyclical nonstationarity[1]. The analytic encoder was developed specifically for such data sources, and has therefore first been applied to DCT coefficient encoding.

The spatial compression system consisting of the DCT, Laplacian modeler and analytic encoder is specifically designed for superior performance at bitrates near those achievable by optimized lossless compressors, producing nearly lossless performance (85% of pixels reconstructed exactly) at the lossless compressor 's bitrate. The DCT–Laplacian system generally outperforms JPEG draft rev. 8 by about 0.5 b/p (or an MSE ratio of about 1:2) when compressing more lossily .

2.0 Spectral Decorrelation

One of the results of our investigations into spectral data compression (a.k.a. "multispectral compression" [sic]) is that data with extension in a spectral dimension can be effectively compressed by first applying a decorrelating orthonormal transformation to each pixel spectrally and then compressing blocks of like coefficients from adjacent pixels spatially as though they were image data.

This is not a new idea, but empirical investigations have demonstrated that three–dimensional (one spectral and two spatial dimensions) optimal decorrelating transforms (Karhunen–Loève or Hotelling transforms) are decomposable into separate spectral and spatial transforms. Thus, there is nothing substantial to be gained by joint spatial–spectral transformations. Whereas such transformations have previously been rejected *a priori* for practical reasons, we now reject them *a posteriori*.

1. The expected moments of transform coefficients vary within blocks, hence nonstationarity, but the pattern is repeated for each block, hence cyclical.

Thus, all that is needed for spectral compression is a decorrelating orthonormal spectral transformation. Once the spectral transformation is done, the resulting channels of coefficients can be compressed using any normal spatial compression technique.

The implementation of the spectral transform is still not simple. The basis vectors of spectral Karhunen–Loève transforms derived for several 5–band AVHRR images varied greatly from scene to scene, and attempting to use a "composite" transform derived from the pooled covariance matrix in a prototype spectral compressor lead to insignificant extra compression (less than 0.1 bits per band per pixel) over a benchmark scheme of compressing each band spatially only, with bits allocated among the bands according to information content.

2.1 Stabilizing the Karhunen–Loève Transform

One basic feature of spectral images which can be overlooked in spatial applications but becomes important in spectral applications is that different surfaces (*e.g.* cloud, water, ice, forest, desert) have different offsets between spectral bands *but similarly aligned random variations about those offsets*. Therefore subscenes with uniform underlying surfaces have similar optimum spectral decorrelating transforms regardless of the surfaces, but scenes with a variety of surface types have optimal spectral transforms which depend on the mix of surfaces, since the intersurface spectral variations swamp the smaller intrasurface variations.

This is graphically illustrated for a two–dimensional system in figure 1 below.

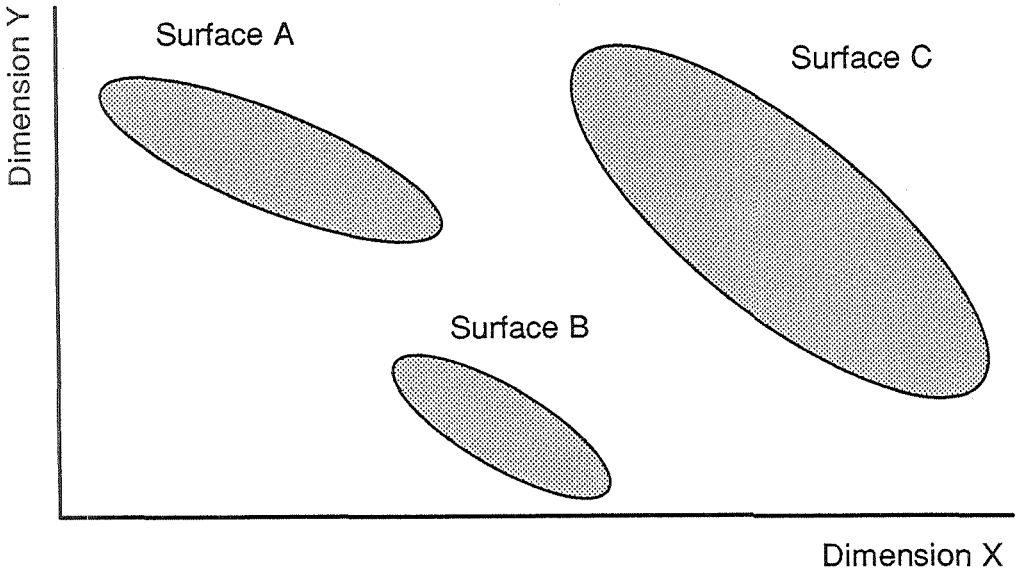

Figure 1 : Example of Alignment of Intrasurface Variations

Clearly in figure 1 the principal component of variation in the X–Y domain is similar for scenes consisting of only one surface type (the main axes of each ellipse are nearly parallel), but for any pair of surface types the principal component of variation in the X–Y domain will be more or less parallel to the vector connecting the centroids of each surface's distribution, while for a scene containing all three surface types the principal component of variation could be oriented in any manner.

Without actually trying to classify pixels and compress only within irregularly shaped regions, the easiest way to eliminate intersurface variations from data being spectrally transformed is to apply

spectral transformation in the spatial frequency domain only on those elements whose wavelength is shorter than plausible spatial feature sizes, *i.e.* those coefficients which register intrasurface variations better than intersurface variations. For DCT compression with small blocksizes (or vector quantization techniques) this may mean spectrally transforming all but the principal spatial components (the "DC terms").

2.2 The Spectral Compressor Front End

A simple and generic implementation of spectral compression as outlined above consists of a standard spatial compressor (*e.g.* the analytic encoder with Laplacian coefficient modeler applied to DCT coefficients) with a centroid extractor and deviate transformer appended to the front end. Assuming *n*–dimensional spectral data and a spatial compressor that operates on 32 by 32 pixel blocks (16x16 and 32x32 are common blocksizes for the DCT, but 4x4 and 8x8 are more common for vector quantization), the front end is depicted in figure 2 below.

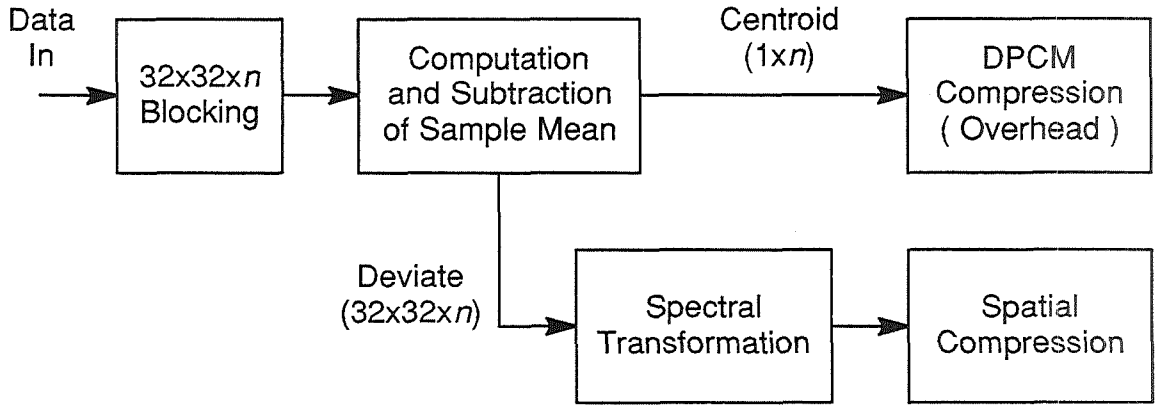

Figure 2 : Front End Block Diagram

When the spatial compressor is a discrete cosine transform–based algorithm the centroid values can be substituted for the DC coefficients of the DCT blocks in the overhead stream, since those coefficients will all be zero for blocks whose centroids have been extracted. Other spatial compressors (*e.g.* mean–residual vector quantizers) also exploit the zero–mean property of the deviates.

3.0 The Spatial Compressor

For the Astro–Space DCT compressor the sensor data are partitioned into 32–pixel square blocks[2], which are transformed into 32–point square blocks of transform coefficients. These blocks are then passed through the Laplacian (analytic) encoder one at a time, with a new arithmetic codeword generated for each block. In between the transform operator and the encoder the data are statistically characterized by a coefficient modeler so that scale predictions can be generated for the Laplacian

2. Previous studies for Mars Rover Sample and Return data compression indicated that the most efective power–of–two sized transforms were 16 pixels and 32 pixels square. The choice of 32 pixels square was based in part on the size of GE's proprietary PFFT chip (1024–point transforms, or 32 transforms in parallel with 32 points each) and in part on the desire to lower the amount of overhead information (which is typically generated on a per–block basis and compressed less efficiently than other data).

encoder. The Laplacian model is purely an empirical choice[3]. This process is depicted below in figure 3.

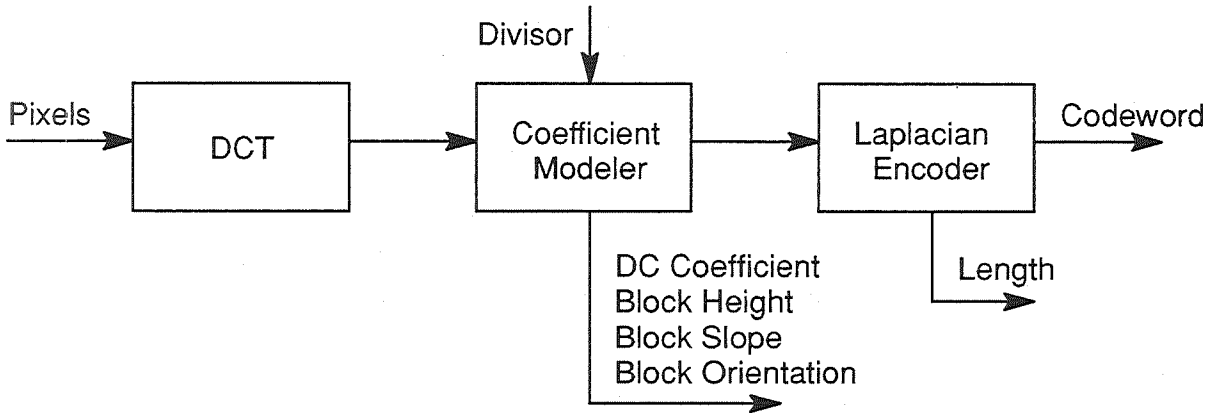

Figure 3 : System Block Diagram

The flow sizes in figure 3 are as follows:

- 1024 pixels enter the discrete cosine transformer,
- 1024 coefficients and one divisor (W_{∞}) enter the coefficient modeler,
- The principal ("DC") coefficient and block model parameters are put out as block overhead for further compression and separate transmission,
- 1023 coefficients (the "AC" coefficients) and associated predictions enter the Laplacian encoder,
- The Laplacian codeword (a long bitstream) and its length are put out, the former as the principal output, the latter as block overhead for further compression and separate transmission.

3.1 The Discrete Cosine Transformer

The Astro–Space discrete cosine transformer takes a 32–pixel square block of data and performs two passes of 32 parallel 32–point one–dimensional discrete cosine transforms. In between the two passes the data are transposed row for column so that the two passes are first horizontally across the block, then vertically across it.

The discrete cosine transform has the property that for typical image data, the transform coefficients are nearly uncorrelated. Specifically, the principal components of samples from first–order Markov processes approach the basis vectors of the discrete cosine transform as the adjacent element correlation coefficient approaches unity. The two–dimensional Markov model is intuitively appealing as an interpretation of images, and typical images have adjacent pixel correlation coefficients in excess of 0.9, so the discrete cosine transform is widely used.

3.2 Decorrelation Efficiency of the Two–Dimensional Discrete Cosine Transform

The covariance hypercube of the coefficients of a two–dimensional transform is simply the four–dimensional transform of the covariance hypercube of the original two–dimensional data block. Given

3. See section 4.4 of Clarke, "Transform Coding of Images", Academic Press, London 1985, for a thorough survey of coefficient distribution models.

an optimal decorrelator (the KLT) the transform domain covariance matrix of one–dimensional data would be diagonal, and the transform domain covariance hypercube of two–dimensional data would be diago–planar (have nonzero entries only on the "diagonal" plane). To quantify the extent to which a transform is optimal, we define two measures of optimality: decorrelation efficiency (the traditional measure) is the proportional reduction in off–diagonal energy (sum of absolute values of covariances) from the data domain to the transform domain[4], and relative remaining energy is the ratio of the total energy in the transform domain to the (planar) trace of the data domain covariance hypercube. Ideally both measures would be one, with decorrelation efficiency approaching from below and relative remaining energy approaching from above.

Figure 4 shows scatter plots of decorrelation efficiency and relative remaining energy as a function of adjacent–pixel correlation[5]. In interpreting these plots it is important to keep in mind that the "optimum" decorrelator corresponding to complete efficiency and no extra remaining energy is only optimum within the following constraints:

- Decorrelation shall only be accomplished within blocks; interblock correlation is not considered,
- Decorrelation shall be accomplished by linear transformation, *and the same transformation must be used on all blocks.*

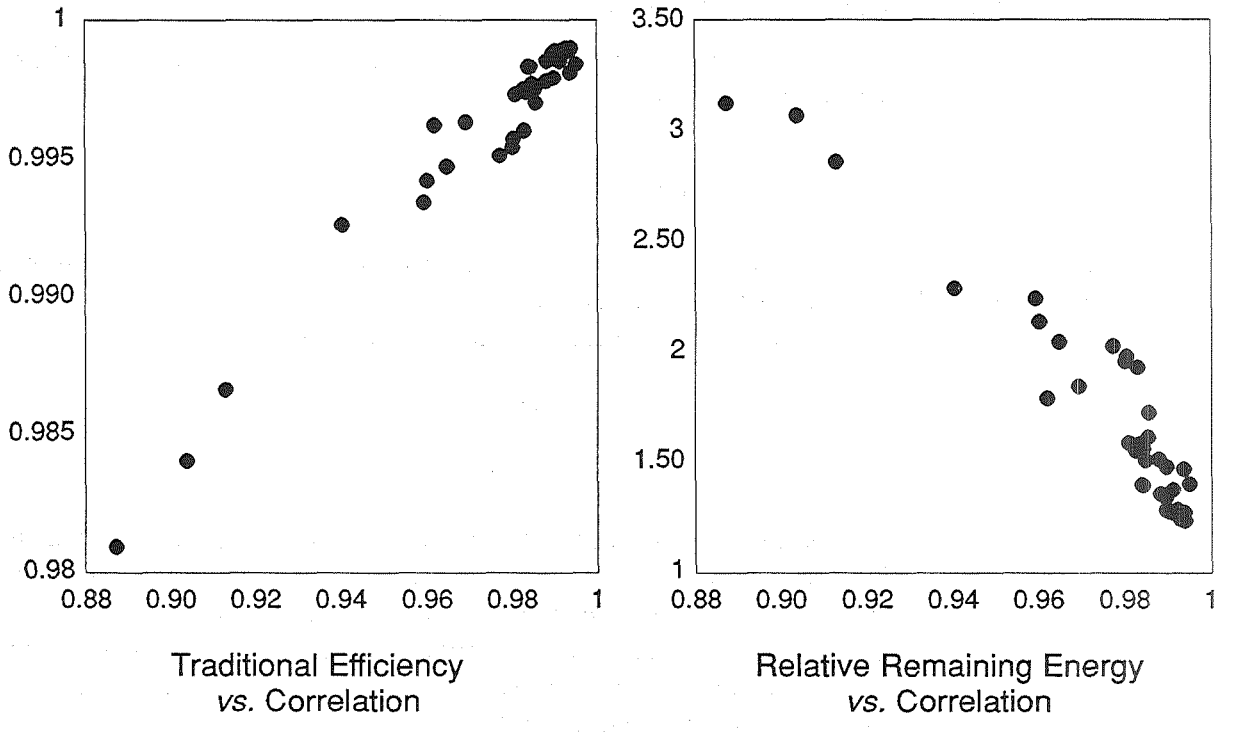

Figure 4 : Decorrelation Efficiency *vs.* Adjacent Element Correlation

4. Diagonal energy is unchanged since: 1) the trace of a matrix is invariant under orthonormal transformation; 2) the covariance matrix diagonal elements are all variances, hence positive; 3) the covariance matrix is real symmetric, hence the transform–domain diagonal elements are all positive.

5. Actually, the independent variable is diagonal correlation with pixels one row up or down *and* one column left or right, raised to the root–one–half power. This is sort of a geometric mean of horizontal and vertical adjacent–pixel correlations, which are not equal for real sensors.

The plots in figures 4 and 5 are based on 35 frames (7 scenes with 5 bands each) of AVHRR data from the 1990 GE benchmark satellite image test suite. The frames are 512x512x10 bits, so efficiency was computed for 16x16 transforms since a 512–pixel square frame does not have sufficient degrees of freedom for a 32^4–element covariance hypercube of full rank.

As the figures suggest, decorrelation efficiency and relative remaining energy are directly related. In the limit, if we assume that all correlations within the 16 by 16 pixel data block are near unity, then the 16^4–element covariance hypercube has 16^2 units of information energy out of 16^4 total units of energy. Expressing relative remaining energy in terms of decorrelation efficiency η (the portion of the 16^4–16^2 units of redundant energy removed),

$$\frac{E_r}{E_o} \approx \frac{16^2 + (1-\eta)\left(16^4 - 16^2\right)}{16^2} = 256 - 255\eta.$$

This approximate relationship is apparent in figures 4 and 5. Clearly relative remaining energy is more sensitive than the traditional measure of efficiency. Note that changing the size of the block and the assumed amount of non–diagonal energy in the covariance hypercube changes the two coefficients in the right–hand expression above without affecting their unit difference. For 32–pixel square blocks the coefficients in the expression are about one thousand.

Figure 5 indicates a typical relative remaining energy of two for the test suite. Using the relationship of entropy (ϕ) as a function of energy (or variance, σ^2)

$$\phi_\sigma \approx \log_4 \sigma^2 + 2$$

a relative remaining energy of two means a half a bit per coefficient (or pixel) of extra information is needed to encode the not–quite–decorrelated data set. Using the relationship between lost resolution b (bits per symbol) and mean squared error for densely quantized data

$$MSE = \frac{2^{2b} - 1}{12}$$

a half of a bit of resolution lost causes a mean squared error of $1/12$ at a bitrate that would otherwise have been lossless, or doubles the mean squared error at other bitrates. While the discrete cosine transform is good, it is still not nearly optimal even within the constraints given above for this efficiency analysis. For this reason Astro–Space division continues to study DCT pre– or postprocessing, *e.g.* sparse matrix pre– or postmultiplication to exploit the DCT as a first (or last) easily computable step on the way to a good approximation of the KLT.

3.3 Coefficient Modeling and Quantization

The coefficient modeler, quantizer, and analytic encoder are optimized for data having the Laplacian distribution

$$f_x(x) = \frac{1}{2s} e^{-|x|/s}.$$

The coefficient modeler predicts the value of |x| from block overhead parameters indicating general block energy, energy decay with sequency, and energy orientation, and from coefficient reconstructions that will be available to the decompressor at the time it is working to decode any particular coefficient. The prediction of |x| for any coefficient is used as s (the dispersion parameter of the Laplacian distribution) for quantizing and encoding that coefficient. Clearly, s is not constrained to be constant for any block of coefficients, nor even to vary slowly.

The quantizer design is based upon the following premises:

- A tradeoff ratio $-R$ is set for expected squared error σ^2 *vs.* expected code length λ,
- The center quantizer bin will be symmetric about zero,
- The center bin will be set to that width (W_0) for which the marginal increase in σ^2 for expanding that bin at each end is R times the marginal reduction in λ,
- Subsequent bins will extend outward toward positive and negative infinity from the endpoints of previous bins until the marginal increase in σ^2 for expanding those bins is R times the marginal reduction in λ, at which point those bins will end (at width W_0) and new bins will begin.

Varying R in this scheme will sweep through the various points on the limiting curve of output rate *vs.* distortion, with each value of R producing that point on the rate–distortion curve where the derivative of distortion with respect to rate is $-R$.

Solving the equation

$$\frac{\partial \sigma^2}{\partial \lambda} = -R$$

for the upper bound b of a quantization bin as a function of the lower bound a, the cost constraint R, and the dispersion s of the laplacian–distributed variable x, and also solving for a symmetrical bin about zero (for which $a = -b$) yields quantizers with uniform size bins of f zero, a slightly larger bin about zero, bin sizes that increase slightly with s, and everything varying in the same proportions when R is varied. In particular, if we define W_∞ to be the width of a quantizer bin for the uniform distribution given cost constraint R, then the bin widths and reconstruction points (constrained expected values of positive x given $a \le x \le b$, where a and b are the bin boundaries for positive nonzero bins) are tabulated below in proportion to W_∞ for some values of s.

From Table 1 it is apparent that the quantizer favors finer quantization of low–energy signals (high–sequency DCT coefficients) than high–energy signals, since distortion reduction is "cheaper" for them in proportion to output expansion.

Such a quantization scheme obviously fixes neither block distortion nor block output rate. High–energy blocks will obviously consume more bits than low–energy blocks given the same value for W_∞ (the parameter used to control overall compression quality, maintained constant for long periods of time), but they will not consume enough extra bits to match the output quality of the low–energy blocks because marginal distortion improvements are more expensive for high–energy signals.

Dispersion of Distribution	Width of Bin About Zero	Width of Nonzero Bins	Reconstruction Point
s	W_0/W_∞	W_0/W_∞	$E(x-a)/W_\infty$
0	$2/\sqrt{6}$	$1/\sqrt{6}$	0
0.01	0.81699	0.41837	0.01000
0.02	0.81845	0.42874	0.02000
0.05	0.82852	0.46123	0.04995
0.1	0.85530	0.51548	0.09701
0.2	0.89775	0.60094	0.16867
0.5	0.94784	0.73621	0.28087
1	0.97151	0.82913	0.35792
2	0.98509	0.89898	0.41593
5	0.99387	0.95441	0.46203
∞	1	1	$1/2$

Table 1 : Quantizer Parameters

The actual quantizer is simply realized as two interpolated lookup tables: one yielding the inverse of the nonzero bin size and the other yielding half the difference between the zero and nonzero bin sizes. Using these values the coefficient x is quantized into the integer i according to

$$i = \frac{x \pm \frac{W_0 - W_0}{2}}{W_0}$$

with i rounded to the nearest integer, and the alternating sign matching the sign of x. The formula for i above is designed to map the quantization bin boundaries a and b to odd multiples of $1/2$, so that all values of x in the same bin round to the same i.

3.4 The Laplacian (Analytic) Encoder

The analytic encoder is an arithmetic encoder which uses an analytic expression for the c.d.f. of the distribution that a particular datum is assumed to have been drawn from, rather than a preprogrammed table or prior experience. Thus, along with each datum the encoder expects as many parameters as are needed to define the c.d.f. In the case of the laplacian distribution, the only parameter needed is s, the expected absolute value.

The Laplacian encoder is a state machine which takes in Laplacian–distributed data coupled with scale predictions and puts out a bit stream called a codeword. The bit stream is unlimited in length, and is restarted for each 32–pixel square block in Astro–Space division's current implementation of DCT compression. The Laplacian encoder can terminate a codeword in either of two ways: simply

by putting out (as compressible overhead) the length of the codeword, or by extending the codeword (theoretically by $2\frac{1}{3}$ bits on average) so that regardless of what follows the codeword will be properly decoded. The former option is preferable for applications where data losses are possible, as it allows a total restart at the beginning of each block.

Freedom to encode diverse data in any order, rather than in statistically homogeneous groups, brings a number of benefits:

- Data can be grouped spatially, for instance encoding all of the coefficients of a single discrete cosine transform block as one symbol set. This obviously simplifies buffering and also provides robustness. For instance, if a 32 by 32 pixel block is compressed to one bit per pixel, then one thousand consecutive bits will all be from the same block, so that burst errors in communication are likely to affect only one block at a time (or two at most).
- A carefully designed predictor of |x| used to choose the distribution dispersion parameter s will be able to adjust immediately to any nonstationarity in the data source, tracking variations in source statistics on the single–sample level rather than from sample set to sample set.
- No symbol distribution or similar overhead needs to be transmitted.

The codeword put out by the Laplacian encoder is the principal compression product for each block. For instance, for a compressor operating at four bits per pixel the codeword would be, on average, about four thousand bits for each 32x32 block (though unusual blocks could yield much longer or much shorter codewords for the same setting of the control parameter W_∞).

3.4.1 Encoder Operation

The Laplacian encoder differs from normal arithmetic coding by not grouping similar data together: each datum is encoded individually as though it came from a large sample, but each datum is in fact a sample of one. Instead of having a probability table the Laplacian encoder has embedded in it the analytic expression for the cumulative density function of a Laplacian distribution,

$$F_x(x) = \frac{1}{2}e^{x/s} \quad | \; x \le 0, \qquad F_x(x) = 1 - \frac{1}{2}e^{-x/s} \quad | \; x > 0.$$

Each datum is modelled as coming from a Laplacian distribution with mean absolute value s, with a unique value of s for each datum. Obviously, s cannot be a measured sample statistic (there is only one datum in the sample); it must be a prediction entirely independent of the datum to be encoded. As long as the predictor is unbiased and has the power to discriminate, i.e. as long as

$$E(x)|s = s$$

while

$$0 \ll \sigma_s < \mu_s$$

the encoder will operate as efficiently as, and often more efficiently than, encoders which rely on grouping similar data together.

3.5 Implementation of Laplacian Encoder/Decoder

The encoder follows the general procedure for arithmetic encoders, maintaining a bounded interval within which the ultimate codeword will fall, and transmitting bits (and shifting up the boundaries) as the boundaries converge to the point where leading bits match. Specifically, a pass through the encoder (with one symbol) consists of the following:

(1) Computation of Hi and Lo, the boundaries of the interval on [0, 1] where the codeword will be. The computation yields a subinterval on the stored previous interval Hi– and Lo+ (derived by truncating the previous Hi and Lo inward to some buffer size, e.g. 16 bits) corresponding to the c.d.f. interval between the bin boundaries a and b for the input datum i and the dispersion s. The new Hi and Lo are temporarily maintained at their full precision, e.g. 32 bits if Hi–, Lo+, and the c.d.f. points are all 16–bit numbers.

(2) Transmission of as many of the leading bits as match in Hi and Lo (actually, for exception handling and sneaky storage purposes, transmission of "0" if Hi $\leq \frac{1}{2}$ and "1" if Lo $\geq \frac{1}{2}$, repeated as often as possible). As the bits are transmitted, Hi and Lo are shifted up (padding with zeroes) so that in the end

$$0 \leq \text{Lo} < \frac{1}{2} < \text{Hi} \leq 1.$$

(3) Truncation of Hi and Lo into new values of Hi– and Lo+. As the labels imply, Hi– is derived by rounding Hi down while Lo+ is derived by rounding Lo up. This rounding towards the inside of the interval guarantees that adjacent intervals do not overlap after truncation. If they did, a codeword in the overlapping region could be misinterpreted.

Mathematically the decoding process for one coefficient consists merely of solving the actual codeword, Hi, and Lo, for a c.d.f. value, and then evaluating s and the inverse of the Laplacian c.d.f. for a value of x', which is quantized to i (note that the actual value of x' produced by the inverse c.d.f will lie in the same quantization bin as the original value of x, but beyond that it carries no information, and therefore should be reduced to its bin label i). The true value of x will then be estimated by adding (subtracting for negative i) a tabulated and interpolated expected value (the last column in table 1 above) to the quantization bin boundary a (or $-a$ for negative i).

4.0 Block Overhead Compression

For each block of 1024 pixels the compressor puts out four pieces of overhead information that need to be transmitted along with the codeword for reconstruction:

- The raw DC coefficients (a 5–dimensional quantity),
- The codeword length (in bits),
- The "height" of the scaled, quantized coefficient block absolute values (a 5–dimensional quantity),
- The "slope" of the scaled, quantized coefficient block absolute values (a 5–dimensional quantity).

- The "orientation" of the scaled, quantized coefficient block absolute values (a 5–dimensional quantity).

The "height" of the coefficient block is a measure of information content and therefore obviously correlated with the codeword length, and the two are also be weakly correlated with the "slope" of the coefficient block. The DC coefficients, codeword lengths and coefficient block "heights" are also obviously correlated with their counterparts in adjacent blocks, and the coefficient block "slopes" are also weakly spatially correlated. Finally, all of the 5–dimensional quantities have internal correlation among their scalar components. These correlations are exploited in a DPCM compressor of overhead information. The codeword length is compressed losslessly, and the other quantities are compressed lossily, with the compressor using those values of lossily compressed parameters that the reconstructor will have rather than the original values.

5.0 Performance of the Compressor on Meteorological Images

Astro–Space division maintains ongoing efforts to determine the value to end users of lossily compressed imagery. Studies have included generic quality measures (*e.g.* mean squared and absolute errors, frequency distortion, automated pattern recognition, edge detection and location, difference histograms and images, *etc.*), user–oriented objective quality measures (*e.g.* calculated sea surface temperatures, spatial coherence analyses, cloud type identification, cloud cover calculation, forest classification, crop identification, ice fissure detection, location, and measurement, interplanetary probe landing site hazard identification, surface altitude and grade derivation from stereo pairs, *etc.*), and subjective expert assessments. Algorithms have been evaluated not only for the quality of their reconstructed images but also for hardware requirements at the transmitter, vulnerability to communications errors, compatibility with packeting and encryption, and exploitation of error–correcting codes and retransmission.

Most analyses have indicated a surprising robustness of image–derived information to compression losses, particularly to compression errors without structure, *e.g.* white noise. A battery of meteorological tests on 5–band AVHRR data indicated that with our best compression techniques compression ratios of 50:1 were achievable with borderline unacceptable defects in the images. Images compressed 25:1 were clearly acceptable for all analyses, and images compressed between 5:1 and 10:1 exhibited either no loss (due to rounding to some user–selected relevant resolution) or negligible loss in image–derived quantities.

For example, one study focussed on computing multichannel sea surface temperatures (MCSST) from compressed AVHRR data. Table 2 below summarizes root–mean–square (RMS) errors in degrees Kelvin for a variety of compressors.

Compression Ratio	DCT/Laplacian	JPEG Draft Rev. 8	DPCM	MRVQ
5:1	0.146 °K	0.292 °K	0.388 °K	0.331 °K
10:1	0.289 °K	0.588 °K	0.863 °K	0.560 °K
20:1	0.396 °K	0.811 °K	1.233 °K	0.789 °K

Table 2 : Sea Surface Temperature RMS Errors

The data in table 2 were interpolated for those compressors (all but DCT/Laplacian, which employs rate feedback control) which don't produce an arbitrary compression ratio exactly on demand. The

DCT Laplacian algorithm used involved no spectral compression or spectral resource allocation in order to produce results commensurate with the other (purely spatial) techniques.

The RMS errors of figure 2 were computed on a pixel–by–pixel basis, and would seem (in some cases) to be unacceptable, except that typical sea surface temperature analyses involve computing average temperatures over regions of not less than 256 pixels. Such averaging reduces rms error by factors in excess of 16, making even the worst compressed data look good. Similar averaging effects help in other meteorological analyses, *e.g.* cloud cover calculations, since most analyses focus on the properties of large objects (*e.g.* cloud formations, ocean currents, polar icecaps, *etc.*).

Another interesting result of our user–oriented studies is that scalar error measures are usually entirely adequate for intercomparisons of conceptually dissimilar algorithms. For instance, when we presented a panel of meteorologists with a suite of images compressed to various degrees with various algorithms and asked them to subjectively grade the reconstructions, the rank order in which they placed the reconstructions exactly matched the rank order of their mean squared errors (MSE), without regard to whether the images were compressed using any of a number of DCT algorithms, a DPCM algorithm, or an MRVQ algorithm, all of which produced different characteristic errors. In the spirit of this observation in defense of the much–maligned MSE, figure 5 below presents the MSE performance of the DCT/Laplacian compressor, both with the spectral transform and without, for a particularly challenging scene combining clouds on two levels, land, open water, and polar ice.

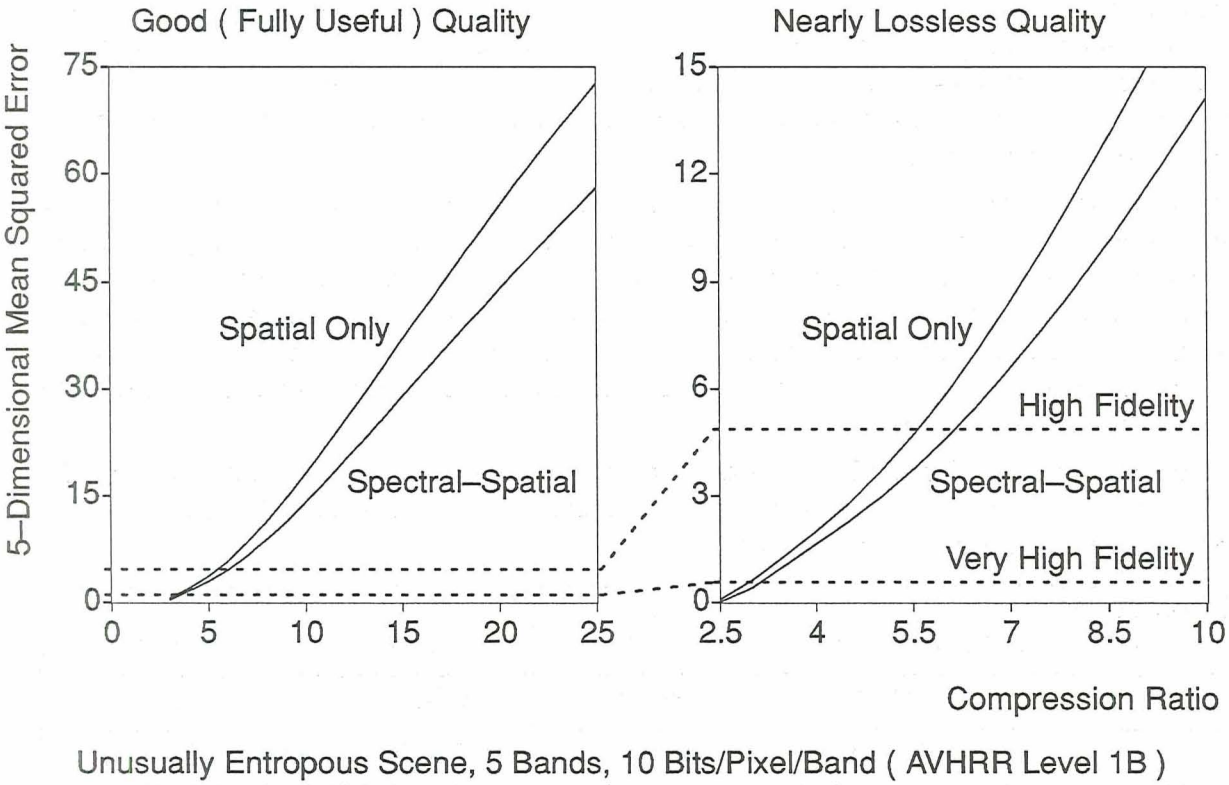

Figure 5 : Bad–Case Compressor Performance

Operating the spectral compressor without the spectral transform but with the same value of W_∞ used for each spectral band causes the same cost constraint R to be used across the whole image, resulting

in bit allocation among the bands as well as among the blocks in such a manner as to reduce overall distortion as much as possible within the tradeoff ratio $-R$ of distortion for output. The difference in performance shown in figure 5 is due entirely to spectral decorrelation, and can be used as a measure of the added benefit to be gained from full spectral/spatial compression over spatial compression with output bits allocated among the spectral bands according to need. The latter approach suggests itself as a way to compartmentalize the effects of single–band sensor hardware failures such as that on the AVHRR sensor aboard TIROS–7.

Figure 5 shows two statistical classifications of reconstruction quality in use at Astro–Space Division: High Fidelity reconstruction is defined as reconstruction with an MSE of one squared quantum per scalar dimension (chosen for parsimony and uniphilia), and Very High Fidelity reconstruction is defined as reconstruction with an MSE of $1/12$ of one squared quantum per scalar dimension (chosen to match quantization error for dense quantizers). Together with Lossless and Lossy these provide a range of qualitative descriptors of reconstructions.

6.0 Choosing Spectral Bands for a Sensor

Existing sensors designed without spectral data compression in mind (such as AVHRR) tend to have spectral bands which are relatively uncorrelated. This is only natural, since without spectral compression the cost (in storage space and transmission bandwidth) of adding a new band to a sensor is independent of the added nonredundant information in the band, so bands are chosen to provide as much nonredundant information as possible. Intuitively, when a sensor is designed without spectral compression in mind, the optimal sensor design produces spectrally irreducible data, thereby packing in the most information per band, and justifying the omission of spectral compression after the fact. This leads to spectral compressor performance such as that shown in figure 5, where exploiting spectral redundancies provides only limited performance improvements over well–designed spatial compression.

Adding spectral compression to a sensor changes the cost of adding a new band by exploiting spectral redundancies so that a system designer has only to pay (in bytes, baud, and bucks) for the nonredundant information in a band, rather than paying a fixed price for all the information in a band. Since the cost and added value of a new band will vary together given spectral compression, there will no longer be an obvious incentive to choose highly uncorrelated bands for a sensor. Future sensors with spectral compression will therefore tend to have more spectral bands, and more correlation among bands, than current sensors designed for the same purposes.

7.0 Choosing Sensor Output Quantization Resolution

In parallel with our user–oriented studies Astro–Space division also conducts systems studies to assess the impact of data compression on sensor design, backplane capacity, buffer memory requirements, and downlink encryption and error correction. In this context, one of the most common customer requirements is for lossless data compression regardless of other systemic, dynamic, and random sources of loss. Astro–Space Division develops lossless compressors for various uses, but future sensors ought to be designed for lossy compression, since lossy data compression applied to higher–resolution data always produces better radiometric accuracy for a given bitrate than lossless data compression applied to lower–resolution data.

This is intuitively a consequence of the fact that the underlying "exact" data values are real numbers, and the difference between higher–resolution sensor data and lower–resolution sensor data is simply the degree of truncation, with the lower–resolution data always worse than, and reproducible from,

the higher–resolution data (higher–resolution data is produced by a quantizer with bin boundaries at all of the boundaries of the lower–resolution quantizer, plus other intermediate boundaries as well). Since the lower–resolution data is reproducible from the higher–resolution data, the choice between lossless compression of the lower–resolution data and lossy compression of the higher–resolution data becomes a choice between two lossy approaches to compressing the higher–resolution data: truncation followed by lossless compression or direct lossless compression. Direct lossless compression has flexibility to incur losses in some optimal manner, and therefore should always outperform truncation followed by lossless compression, in which the losses are fixed in an arbitrary manner.

Retaining higher–resolution data for lossy compression is rarely a matter of buying a more precise (and hence more expensive) analog–to–digital converter (ADC), since most sensor data needs to be calibrated, corrected for nonlinear sensor response, spatially resampled, *etc.* The calibrated/corrected/resampled/processed data will be real numbers even though the ADC produces only integers, so as long as some of these steps (and preferably all of them, which is feasible with current spaceborne processors) occur before compression, the data will have to be truncated once more (or many times more) before compression, and these later truncations will determine the resolution of the data being compressed.

The effect of original data resolution on reconstructed data quality can be assessed using the previously presented formula for MSE as a function of bits truncated,

$$MSE = \frac{2^{2b} - 1}{12} .$$

This formula can be inverted to get effective resolution lost as a consequence of error, which, combined with the original resolution, yields effective remaining resolution:

$$R_e \equiv R_0 - \log_4(12MSE + 1) .$$

Effective remaining resolutions can be compared for reconstructions independently of the original data resolution, so one way to assess the value of extra bits of resolution is to measure how many bits of effective resolution are gained by adding a few bits of resolution to the original data, given that in either case the data have to be compressed to the same fixed bitrate before reconstruction.

Such a study is presented in figure 6 below: 10–bit AVHRR data and 8–bit truncations of the same data were compressed to a number of bitrates and reconstructed, and effective resolutions compared. The improvement in effective resolution from having two extra bits in the original data is plotted below as a function of the effective resolution lost in compressing the 8–bit data, so the benefit of extra resolution is expressed as a function of how much of the lesser resolution data are being lost to compression. Clearly, as less of the information in the lower–resolution data is being retained (*i.e.* lossier compression is being performed), the benefits of having even more resolution in the original data evaporate.

Figure 6 shows that substantial gains can be made by increasing original data resolution until the resolution of the data being compressed is such that at least two bits of effective resolution are lost in the compression. Another inference from the figure above is that lossless compression of an image (compressing a lower–resolution image without losing any effective resolution) is about one bit

worse in effective resolution than lossier compression of a higher–resolution version of the same image.

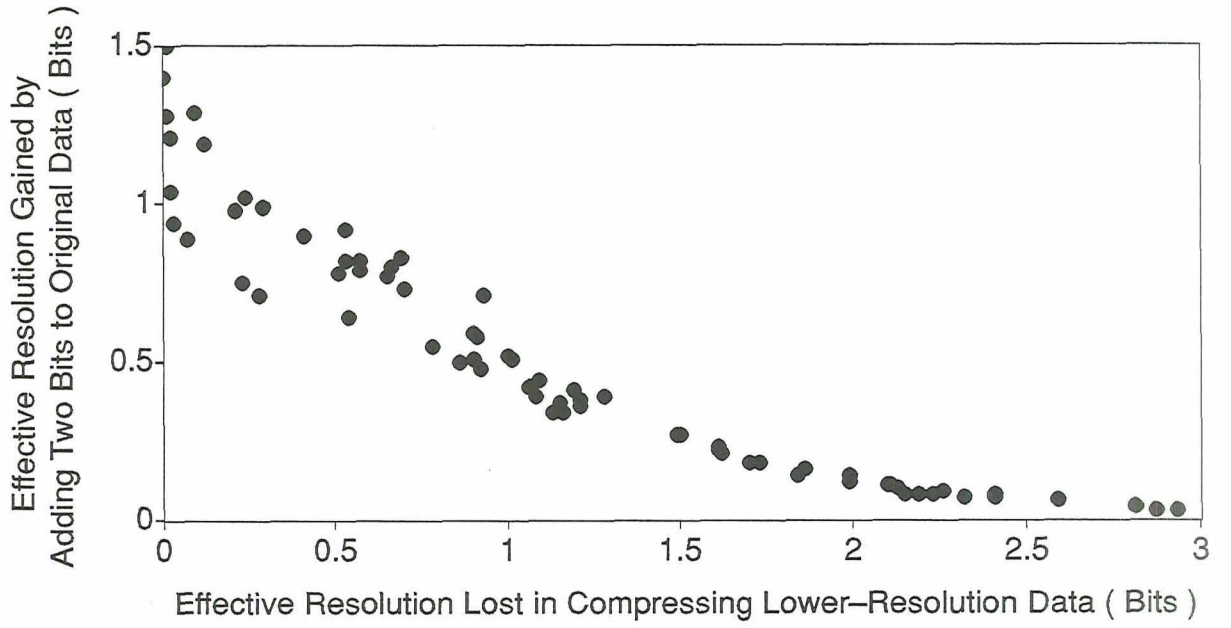

Figure 6 : Benefits of Increasing Original Data Resolution

8.0 Conclusion

Astro–Space Division continues to investigate, develop, and build compression systems for high–volume spectral images. Existing techniques compress data at ratios from 3:1 with pristine reconstruction to 25:1 with acceptable reconstruction quality. Data compression is being designed into future sensors, and future sensors are being designed differently because of data compression. In particular, sensors designed with lossy data compression in mind produce higher resolution calibrated, corrected, and resampled data, and spectral sensors designed with spectral compression in mind can afford more spectral bands since the cost (in storage volume and downlink bandwidth) of adding a new band depends only on the amount of new information in the band.