

NASA Conference Publication 3315

1995 Science Information Management and Data Compression Workshop

Edited by
James C. Tilton
*Goddard Space Flight Center
Greenbelt, Maryland*

*Proceedings of a workshop sponsored by the
National Aeronautics and Space Administration
in cooperation with the
Washington/Northern Virginia Chapter of the
Institute of Electrical and Electronics Engineers
Geoscience and Remote Sensing Society
and held at NASA Goddard Space Flight Center
Greenbelt, Maryland
October 26 - 27, 1995*



National Aeronautics and
Space Administration

Goddard Space Flight Center
Greenbelt, Maryland

1995

This publication is available from the NASA Center for Aerospace Information,
800 Elkridge Landing Road, Linthicum Heights, MD 21090-2934, (301) 621-0390.

FOREWORD

The Science Information Management and Data Compression Workshop was held on October 26-27, 1995, at the NASA Goddard Space Flight Center, Greenbelt, Maryland. This NASA Conference Publication serves as the proceedings for the workshop. The workshop organized by the Information Sciences Technology Branch, Space Data and Computing Division of the NASA Goddard Space Flight Center, and was supported by the Office of Mission to Planet Earth, NASA Headquarters. The workshop was held in cooperation with the Washington/Northern Virginia Chapter of the Institute of Electrical and Electronics Engineers (IEEE) Geoscience and Remote Sensing Society.

The goal of the Science Information Management and Data Compression Workshop was to explore promising computational approaches for handling the collection, ingestion, archival and retrieval of large quantities of data in future Earth and space science missions. It consisted of fourteen presentations covering a range of information management and data compression approaches that are being or have been integrated into actual or prototypical Earth or space science data information systems, or that hold promise for such an application.

Papers were selected from papers submitted in response to a widely distributed Call for Papers. Fourteen papers were presented in 3 sessions. Discussion was encouraged by scheduling ample time for each paper.

The workshop was organized by James C. Tilton and Robert F. Crompt of the NASA Goddard Space Flight Center.

Workshop Organizers

James C. Tilton
Mail Code 935
NASA GSFC
Greenbelt, MD 20771
phone: (301) 286-9510
FAX: (301) 286-1776
Internet:
tilton@chrpisis.gsfc.nasa.gov

Robert F. Crompt
Mail Code 935
NASA GSFC
Greenbelt, MD 20771
phone: (301) 286-4351
FAX: (301) 286-1776
Internet:
crompt@sauquoit.gsfc.nasa.gov

CONTENTS

A Spatially Adaptive Spectral Re-ordering Technique for Lossless Coding of Hyper-spectral Images	1 - 1
<i>Nasir D. Memon, Northern Illinois University, DeKalb, IL, USA; and Nikolas Galatsanos, Illinois Institute of Technology, Chicago, IL, USA</i>	
A Comparison of Model-Based VQ Compression with other VQ Approaches.	13 - 2
<i>Mareboyana Manohar, Bowie State University, Bowie, MD, USA; and James C. Tilton, NASA Goddard Space Flight Center, Greenbelt, MD, USA</i>	
Remotely Sensed Image Compression Based on Wavelet Transform	23 - 3
<i>Seong W. Kim, Heung K. Lee, Soon D. Choi, Korea Advanced Institute of Science and Technology, Taejon, Korea; and Kyung S. Kim, Korea Institute of Science and Technology, Taejon, Korea</i>	
A Packet Data Compressor.	35 - 4
<i>Mitchell R. Grunes, Allied Signal Technical Services Corp.; and Junho Choi, Naval Research Laboratory, Washington, DC, USA</i>	
Reduction and Coding of Synthetic Aperture Radar Data with Fourier Transforms.	45 - 5
<i>David G. Tilley, Elkridge, MD, USA</i>	
Selecting a General-Purpose Data Compression Algorithm.	55 - 6
<i>Gary Jason Mathews, NASA Goddard Space Flight Center, Greenbelt, MD, USA</i>	
Compression Research on the REINAS Project.	65 - 7
<i>Glen Langdon, Jr., Alex Pang, Craig M. Wittenbrink, Eric Rosen, William Macy, Bruce R. Montague, Carles Pi-Sunyer, Jim Spring, David Kulp, Dean Long, Bryan Mealy and Patrick Mantey, Baskin Center for Computer Engineering & Information Sciences, University of California-Santa Cruz, Santa Cruz, CA, USA</i>	
KRESKA: A Compression System for Small and Very Large Images	75 - 8
<i>Krystyna W. Ohnesorge and Rene Sennhauser, MultiMedia Laboratory, University of Zürich, Zürich, Switzerland</i>	
Alternate Physical Formats for Storing Data in HDF.	91 - 9
<i>Mike Folk and Quincey Koziol, National Center for Supercomputing Applications, University of Illinois, Champaign-Urbana, IL, USA</i>	
Context-Based Retrieval of Remote Sensed Images Using a Feature-Based Approach.	103 - 10
<i>Asha Vellaikal and Son Dao, Information Sciences Lab, Hughes Research Laboratories, Malibu, CA, USA; and C.-C. Jay Kuo, Signal and Image Processing Institute, University of Southern California, Los Angeles, CA, USA</i>	
Machine Learning for a Toolkit for Image Mining	115 - 1
<i>Richard L. Delanoy, Massachusetts Institute of Technology / Lincoln Laboratory, Lexington, MA, USA</i>	

Dissemination of Compressed Satellite Imagery within the
Navy SPAWAR Central Site Product Display Environment 123-12
Oleg Kiselyov and Paul Fisher,
Computer and Information Sciences, Inc., Denton, TX, USA

Data Management and Scientific Integration within the
Atmospheric Radiation Measurement Program 131-13
Deborah K. Gracio, Larry D. Hatfield, Kenneth R. Yates and Jimmy W. Voyles,
Pacific Northwest Laboratory, Richland, WA, USA;
Joyce L. Tichler, Brookhaven National Laboratory, Upton, NY, USA;
Richard T. Cederwall, Mark J. Laufersweiler, Martion J. Leach,
Lawrence Livermore National Laboratory, Livermore CA, USA; and
Paul Singley, Oak Ridge National Laboratory, Oak Ridge, TN, USA

Object-Oriented Structures Supporting Remote Sensing Databases 139-14
Keith Wichmann, Global Science and Technology, Greenbelt, MD, USA; and
Robert F. Crompt, NASA Goddard Space Flight Center, Greenbelt, MD, USA

A Spatially Adaptive Spectral Re-Ordering Technique for Lossless Coding of Hyper-spectral Images

Nasir D. Memon
Department of Computer Science,
Northern Illinois University,
DeKalb, IL 60115
memon@cs.niu.edu

Nikolas Galatsanos
Electrical Engineering
Illinois Institute of Technology
Chicago, IL
npg@ikaria.iit.edu

Abstract

In this paper, we propose a new approach, applicable to lossless compression of hyper-spectral images, that alleviates some limitations of linear prediction as applied to this problem. According to this approach, an adaptive re-ordering of the spectral components of each pixel is performed prior to prediction and encoding. This re-ordering adaptively exploits, on a pixel-by-pixel basis, the presence of inter-band correlations for prediction. Furthermore, the proposed approach takes advantage of spatial correlations, and does not introduce any coding overhead to transmit the order of the spectral bands. This is accomplished by using the assumption that two spatially adjacent pixels are expected to have similar spectral relationships. We thus have a simple technique to exploit spectral and spatial correlations in hyper-spectral data sets, leading to compression performance improvements as compared to our previously reported techniques for lossless compression. We also look at some simple error modeling techniques for further exploiting any structure that remains in the prediction residuals prior to entropy coding.

1 Introduction

Recent years have seen a tremendous increase in the generation, transmission and storage of multi-spectral images. With the advent of high spectral resolution images, also known as *hyper-spectral* images, the need for efficient compression algorithms has become increasingly important. This is due to the high data rates resulting from the large number of spectral bands in such images. For example the High-Resolution Imaging Spectrometer (HIRIS) to be placed on the *Earth Observing System (EOS)* scheduled to be launched by NASA within this decade, is designed to acquire images with 192 spectral bands at 30m spatial resolution with resulting bit-rates of more than 200 Megabits per second.

Although much work has been done towards developing algorithms for compressing image data, sophisticated techniques that exploit the special nature of multi-spectral images have

started emerging only recently [6, 13]. More specifically, as noticed by researchers in other image processing areas, the spectral correlation in such images cannot be assumed to be stationary in nature. In other words, the correlation between bands i and j cannot be assumed the same as that of bands $i+k$ and $j+k$ (see for example [4]). Therefore, techniques based on a stationarity assumption, that have been used for capturing spatial correlations in 2-D images, will yield sub-optimal results when directly extended to the third dimension and applied to hyper-spectral data sets. Very recently, techniques were proposed to capture the non-stationary nature of the spectral correlation of hyper-spectral images for lossy image compression [13, 12, 5]. However, because of the difference in goals, the optimal way of exploiting spatial and spectral correlations for lossy and lossless compression is different. Furthermore, many powerful tools used in lossy compression, such as transform coding and vector quantization, are not applicable to lossless compression.

The two most popular approaches for lossless coding of spatially correlated images are *linear predictive coding* and *context based coding*. For a survey of lossless image compression techniques, see [11] and [8]. Although linear predictive techniques and context-based techniques are known to adequately exploit spatial redundancies, they unfortunately can not be extended in a straight-forward manner to exploit the non-stationary nature of spectral redundancies that are present in hyper-spectral images. More specifically, for context based techniques, the size of the context needed to effectively capture the non-stationary nature of spectral correlations gets to be prohibitively large in terms of the resulting implementation complexity. This is because the number of different contexts grows exponentially in the size of the context. Given the large alphabet size of hyper-spectral image data (10 to 16 bits), such a scheme becomes infeasible for a reasonably sized context needed to effectively capture both spatial and spectral correlations.

Linear predictive techniques also face a similar problem. Due to the non-stationary nature of spectral correlations, it is hard to come up with a predictor for the entire image that can effectively use adjacent bands. Furthermore, it has been observed in [6] that in hyper-spectral data, bands which are spectrally far apart can be highly correlated. Hence the question of *band selection* arises. That is, which band(s) are the best to use for predicting intensity values in a given band. In [14], it was shown that significant compression benefits can be obtained by re-ordering the bands of multi-spectral images, prior to prediction. The problem of computing an optimal ordering was formulated in a graph theoretic setting, admitting an $O(n^2)$ solution for an n -band image.

Although significant improvements were reported, one major limitation of this approach is the fact that it is two-pass. An optimal ordering and corresponding prediction coefficients are first computed by making an entire pass through the data set. Another limitation of the approach is that it re-orders entire bands. That is, it makes the assumption that spectral relationships do not vary spatially. The optimal spectral ordering and prediction co-efficients will change spatially depending on the composition of the objects being imaged. This latter fact is taken into account by [12], Rao et. al. albeit in the context of lossy compression. They too re-order spectral bands in order to optimize inter-band prediction. However, since the optimal prediction co-efficients and spectral ordering changes spatially, they partition the image into blocks, and compute optimal predictors on a block-by-block basis. The predictor coefficients are then transmitted as overhead. Large blocks are required to keep the overhead reasonably small and this leads to poorer prediction performance as compared

to that obtained with a smaller block. Furthermore, they also found that using the previous band for prediction gives results that are very close to those obtained after computing an optimal ordering.

In this paper, we propose a new approach, applicable to lossless compression of hyper-spectral images. According to this approach, an adaptive re-ordering of the spectral components of each pixel is performed prior to prediction and encoding. This re-ordering exploits, on a pixel-by-pixel basis, the presence of inter-band correlations for prediction. Furthermore, the proposed approach takes advantage of spatial correlations, and does not introduce any coding overhead to transmit the order of the spectral bands. This is accomplished by using the assumption that two spatially adjacent pixels are expected to have similar spectral relationships. We thus have a simple technique to exploit spectral and spatial correlations in hyper-spectral data sets, leading to significant performance improvements as compared to our previously reported techniques for lossless compression [10].

2 Spatially Adaptive Spectral Modeling

Given two spatially ‘adjacent’ pixels $X = (x_1, x_2, \dots, x_n)$ and $Y = (y_1, y_2, \dots, y_n)$ of an n band multi-spectral image, we assume that X and Y have similar relationships in the spectral dimension. Hence we scan the image in some order (to be specified below) and estimate the current pixel Y by using optimal ‘model’ computed from one of its spatially adjacent neighbors X that has already been transmitted. Since lossless compression is performed, the prediction error at pixel Y is transmitted, using the true values (y_1, y_2, \dots, y_n) at Y which are found and used to update the model parameters at Y . Continuing in this manner, we have an adaptive technique for encoding a multi-spectral image, which effectively captures both spectral and spatial correlations. Since the technique processes the data pixel by pixel in band-interleaved order, simple realizations of it would be well suited for real-time applications.

The question that arises for each pixel Y , is which of its neighbors X do we obtain parameters from to estimate values at Y ? Furthermore, how does one make this choice such that the adaptive process remains causal when applied over the entire image? An optimal answer to these questions can be obtained by formulating the problem in a graph theoretic setting in a manner similar to [9, 14, 12]. We begin by constructing a weighted graph by assigning a vertex for each pixel in the image. We add two directed edges between the vertices corresponding to spatially ‘adjacent’ pixels X and Y . The notion of adjacency can be based on any model including the commonly employed 4-neighborhood and 8-neighborhood models. The weight on the edge going from X to Y represents the ‘coding cost’ incurred by using optimal parameters derived from X in order to code Y and the weight on the edge going from Y to X represents the vice-versa. Now, it is easy to see that a *minimum weight directed spanning tree* of this weighted graph specifies a minimal cost encoding of the image by using the adaptive technique described above. Specifically, to encode the image we start at the root vertex of this tree and traverse it in depth-first order, encoding each pixel by using the optimal model obtained from its parent. The root vertex is transmitted as is.

It is well known that a minimum weight directed spanning tree can be found in $O(V \log E + E)$ for sparse graphs [3, 7], where V is the number of vertices and E the number of edges.

In our case, $E = 8V$ while using the 8-neighborhood model yielding a computation time essentially of the order $n \log n$ for an n band multi-spectral image.

For lossless compression the cost assigned to the edges of the graph can be the entropy of the prediction residuals. This would represent a bound on the coding cost if the prediction residuals were treated as iid and entropy coded. However, in previous work we have shown that this cost function yields an intractable combinatorial optimization problem [9]. However, we showed that for Laplacian and Gaussian statistics there exist equivalent choice that yield tractable optimizations [9].

Now, although an optimal ordering for processing the pixels can be found by constructing a minimum weight spanning tree, the computational and memory resources needed for doing this may be prohibitive for many applications. In such cases, a simple heuristic that uses a model obtained from one or more neighboring pixels can be used. Nevertheless, computing an optimal ordering would still be of interest as it would give us a performance bound that could be expected of any such simple heuristic.

What we have described up to this point is the general principle of the proposed approach. Different modeling schemes and cost functions for the edges of the graph lead to different implementations. In the next section we describe some simple modeling schemes that we have experimented with thus far in our ongoing investigations. We also give preliminary implementation results that seem to be promising.

3 Modeling by Spectral Re-ordering and Linear Prediction

We now turn to the question of modeling the spectral components of a pixel. Recent studies seem to indicate that correlations in a multi-spectral data set are highest along the spectral dimension. For hyper-spectral data sets it is also known that for homogeneous ground features, intensity values in different bands are related by a multiplicative factor [12] of the form

$$I[i, j, r] = \alpha_{rs} I[i, j, s] + \beta_{rs}$$

where $I[i, j, k]$ is the intensity value at spatial location (i, j) in band k .

However, as stated before, in hyper-spectral data, bands which are spectrally far apart can be highly correlated leading to the question of band selection for prediction. Keeping these two facts in mind we describe below a simple modeling scheme that is based on spectral re-ordering and linear prediction.

In order to model a given pixel X , we first re-order its spectral components into ascending order by intensity values. In other words, given pixel X from an n band image, $X = (x_1, x_2, \dots, x_n)$, let σ be a permutation on n points that sorts the components of X in ascending order and let the resulting vector be $\hat{X} = (\hat{x}_1, \hat{x}_2, \dots, \hat{x}_n)$ where,

$$\hat{x}_1 = x_{\sigma(1)} \leq \hat{x}_2 = x_{\sigma(2)} \leq \dots \leq \hat{x}_n = x_{\sigma(N)}$$

It is clear intuitively that the vector \hat{X} is more amenable to approximation by a linear model as compared to the original vector X . Furthermore, the sorting step brings closer spectral

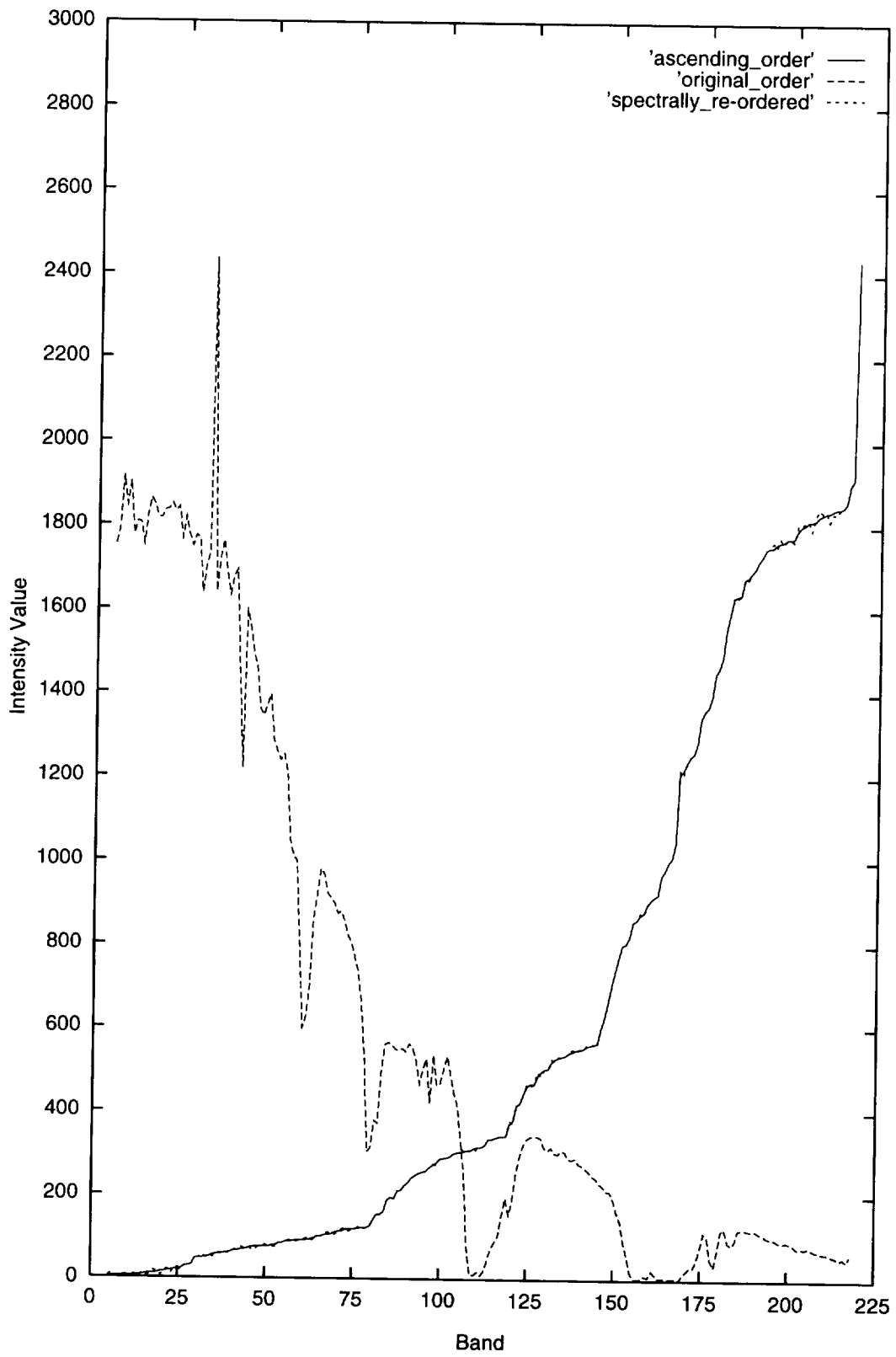


Figure 1: The spectral components of a pixel before and after re-ordering

Prediction Scheme	Residual Entropy
Best JPEG	6.13
Spectral differencing	8.19
Spectral re-ordering and differencing	6.27
Spectral re-ordering, adaptive first order prediction	5.52
Spectral re-ordering, adaptive second order prediction	5.36
Prediction Trees	5.48

Table 1: Entropy of prediction residuals with different schemes.

components that might have been widely separated in the original data although they are highly correlated with one another and have similar intensities.

In figure 1 we plot the spectral components of a random pixel selected from an AVIRIS image. It can be clearly seen from this figure that that the re-ordered vector \hat{X} can be better approximated by a linear model using a given number of parameters as compared to the original vector X . For example, if we wish to approximate \hat{X} with a piecewise linear function, $f(\cdot)$ such that $f(i) = \hat{x}_i \pm \epsilon$ for $1 \leq i \leq n$, we would expect to do so with fewer segments, for a prescribed error tolerance, as compared to the piecewise approximation of the original vector X . In recent work, Bhaskaran et. al. [2] have given an optimal algorithm for waveform approximation with a piecewise linear function at a specified error tolerance ϵ .

Denote the parameters of a piece-wise linear approximation of the vector \hat{X} obtained after re-arranging the spectral components of X into ascending order to be α , Now, if the parameters α and the re-ordering permutation σ are used to model another pixel $Y = (y_1, \dots, y_n)$, then we get $\tilde{Y} = f(\sigma, \alpha)$. The difference between \tilde{Y} and Y , which we denote by E is then the error in our estimate of Y , or in other words, the prediction error. If X and Y are spatially adjacent pixels of a hyper-spectral image, the parameters σ and α derived from X usually provide a good approximation of Y . For example in figure 1 we also show the pixel plot after re-ordering with respect to the sorting permutation σ of a neighboring pixel. The resulting plot of spectral components is almost in ascending order and is difficult to distinguish from the plot of components given in ascending order.

Now, the cost of using the parameters of X to encode Y is simply the cost of encoding E . For lossless compression we are interested in minimizing the number of bits needed to encode E . If we assume that the components of E are zero-mean Laplacian then it is easy to prove that the cost of encoding E is minimized by minimizing the sum of absolute values of the components of E [9]. This gives the necessary cost function for weighting the directed edge going from X to Y .

In table 1 we show some preliminary results obtained with simple approximations of the techniques described above. These were done with the intention of testing the validity of our approach. Implementations with more sophisticated prediction and re-ordering schemes are currently under progress. The results were obtained from a portion of an AVIRIS image of the Cuprite Mining District, Cuprite, Nevada. The image contained 200 samples, 200 lines, and 224 spectral bands. The entry Best JPEG refers to the zero-order entropy of prediction residuals obtained after using the eight JPEG predictors on each band of the image and

selecting the best. Here no spectral information is being taken into account, the prediction is purely spatial.

The entry ‘Spectral Differencing’ gives the zero-order entropy of prediction residuals after taking differences among pixels in adjacent spectral bands but the same spatial location. In other words, a fixed, first-order spectral predictor is employed with coefficient 1. The third entry ‘Spectral Re-ordering and Differencing’ gives the zero-order entropy of prediction errors, after first re-ordering the spectral components by using the permutation that sorts the average of the horizontal, vertical and left-diagonal neighbors and then taking differences as above. It is seen that a considerable reduction in entropy is attained by simply re-ordering the spectral components prior to differencing.

The fourth entry ‘Spectral re-ordering, Adaptive First order prediction’ refers to a technique that re-orders the spectral components as above but uses the value $\alpha_i \cdot \hat{x}_i$ to predict the value of the spectral component x_{i+1} of the re-ordered pixel $\hat{X} = (\hat{x}_1, \dots, \hat{x}_n)$. Here α_i was computed by

$$\alpha_i = \frac{y_{i+1}}{\hat{y}_i}$$

from the vector $Y = (\hat{y}_1, \dots, \hat{y}_n)$ obtained by arranging the average intensity values of the vertical, horizontal and left-diagonal neighbors in ascending order. The entry ‘Spectral re-ordering, Adaptive second order prediction’ refers to a technique identical to the one above but using the previous two components of the re-ordered vector to estimate the value of the next component.

Finally, in order to make comparisons the entry ‘Prediction trees’ lists the zero-order entropies reported in earlier work that used a spatially adaptive prediction scheme called a ‘prediction tree’ obtained from an adjacent spectral band. We see that we can already improve over these results. Further improvements should be achievable by using more sophisticated prediction and/or re-ordering schemes. Note that the current improvements are obtained despite a reduction in computational complexity and memory usage. Also, the new techniques process pixels in band-interleaved order, which is the order in which hyper-spectral images are usually acquired. Hence, such a technique could be used in real-time applications, provided it is simple enough to implement in hardware.

4 Modeling Prediction Errors

If the residual image consisting of prediction errors is treated as an *iid* source, then it can be efficiently coded using any of the standard variable length entropy coding techniques, like Huffman coding or arithmetic coding. Unfortunately, even after applying the most sophisticated prediction techniques, generally the residual image has ample structure which violates the *iid* assumption. Hence, in order to encode the residual image efficiently we need a model that captures the structure that remains after prediction. Most lossless compression techniques that perform error modeling use a *composite source model* to capture the structure in the residual image.

The notion of a composite source model was first described by Berger [1]. In composite source modeling, we model the data as an interleaved sequence generated by multiple sources, called *sub-sources*. Each sub-source has its own model and associated probability

distribution. Each subsequence is treated as an iid sequence generated by the corresponding sub-source and encoded by one of the single source techniques described in the literature. Error modeling schemes that use a composite source model typically consist of the following two components either in an implicit or explicit manner:

1. A *family of probability mass functions*, one for each sub-source, that are used to construct a variable length code.
2. An *switching scheme* which indicates which particular distribution is to be used to encode a particular pixel (or block of pixels).

We have a number of options for each component. In terms of the probability mass function, we could use the same set of probability mass functions for all images. This would be a static approach. We could adapt the probability mass functions after we encounter each symbol. The same information would also be available to the receiver/decoder so it could be adapted in the same manner. This is referred to as an online or backward adaptive technique. Finally, we could scan the entire image, and construct probability mass functions for each sub-source, which could then be sent to the decoder as side information. This is referred to as an off-line or backward adaptive technique. The development of probability mass functions is often integrated with the encoding function as in the case of adaptive Huffman coding, and adaptive arithmetic coding. Similarly we can make switching decisions based only on what is available to both encoder and decoder (backward adaptive), or we could make switching decisions based on information available only to the encoder (forward adaptive). The switching decisions could then be provided to the decoder as side information. The switching techniques studied in this work were all backward adaptive. The reason for this choice was the wealth of information, in form of previously encoded bands, available to both the encoder and decoder. The results reported here are in terms of the entropy of the composite source.

We constructed a composite source model consisting of eight sources. Each source represents a different level of activity that could be present in a given region of an image. We used four different measures of activity. Recall that in the more active regions of the image the magnitude of the prediction residuals will be large, while in the quasi-constant regions of the image the magnitude of the prediction residuals will be smaller. The four different measures used to check the activity level were:

- PPS: The magnitude of the prediction residual at the spectral component that was encoded just before the current component.
- PC: Magnitude of the prediction residual at same spectral location but at a spatially neighboring pixel.
- COMB: Average of PPS and PC.
- AME: Average magnitude of prediction error over the entire set of spectral components of the pixel being currently encoded. current pixel.

Error Modeling Scheme	Composite Source Entropy
SSE	5.36
PPS	5.17
PC	5.19
COMB	5.18
AME	5.31

Table 2: Composite source entropy of prediction residuals after error modeling.

These measures are compared to a set of eight predetermined thresholds $T_1 < T_2, < \dots, < T_8$. Sub-source i was assumed to be active if the activity measure was less than or equal to T_i but greater than T_{i-1} .

In table 2 we show the composite source entropy achieved with different switching techniques for encoding the prediction residuals. The first entry labeled SSE gives the single source entropy, that is the sum of the zero-order entropy of the prediction residuals after using the ‘Spectral Re-ordering and Second Order Prediction’ scheme described in the previous section. This gain is on top of the gain obtained from the prediction step.

The rest of the entries give results for the composite source model with eight sub-sources, using the four different activity measures listed above to perform the switching. In all cases the thresholds were picked to be 0, 2, 4, 8, 16, 32, 64 respectively. These thresholds were picked rather arbitrarily, and we expect to get better performance with a more intelligent choice of thresholds. In fact, optimal thresholds for a given activity measure can be computed for a given image using dynamic programming techniques. However, we used these thresholds because our intention here is only to demonstrate the improvements that can be obtained by exploiting the remaining structure in the prediction residuals prior to encoding. We see that almost a quarter bit improvement can be obtained by the use of a composite source model as opposed to using a single distribution for modeling the source.

Further, we have used information from only a single previous neighbor in order to gauge the activity level at the current pixel. Measures that take into account the prediction errors incurred in a larger neighborhood (spatial and spectral) will lead to more efficient error modeling. Our intention again was to first investigate whether there is a big difference between using spatially adjacent pixels or spectrally adjacent pixels for estimating the activity at the current pixel. It seems that there is a slight advantage to using information from spectrally adjacent pixels.

5 Conclusions and Future Work

We have presented in this paper a new approach for lossless compression of multi-spectral image data that exploits both spectral and spatial correlations in a simple and adaptive manner. What we have described in this paper is just one choice of predictor, re-ordering and encoding cost estimation. A number of alternatives can be used. Implementation results with a few different choices schemes are currently under investigation and will will be

described in the full version of the paper. Also, we need to make more detailed comparisons of compression performances obtained with other schemes given in the literature [14, 12] that also employ spectral re-ordering prior to prediction.

The error modelling schemes used in our preliminary simulations have been rather ad-hoc. Better schemes for partitioning the prediction residuals need to be devised that will lead to further lowering of final bit rates.

Finally, the modeling paradigm that we have introduced in this paper can also be utilized for lossy coding of multi-spectral images, by quantizing the residuals. Lossy coding techniques that utilize our approach will be investigated in future work.

References

- [1] T. Berger. *Rate Distortion Theory: A Mathematical Basis for Data Compression*. Prentice-Hall, Englewood Cliffs, NJ, 1971.
- [2] V. Bhaskaran, B. K. Natrajan, and K. Konstantinides. Optimal piecewise-linear compression of images. In *Proceedings of the Data Compression Conference*, pages 168–177. IEEE Computer Society Press, 1993.
- [3] P. M. Camerini, L. Fratta, and F. Maffioli. A note on finding optimum branchings. *Networks*, 9:309–312, 1979.
- [4] N. Galastanos and R. Chin. Digital restoration of mult-channel images. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 37(3):415–421, 1989.
- [5] S. Gupta and A. Gersho. Feature predictive vector quantization of multispectral images. *IEEE Transactions on Geoscience and Remote Sensing*, 30:491–501, May 1992.
- [6] R. N. Hoffman and D. W. Johnson. Application of eof's to multispectral imagery: Data compression and noise detection for aviris. *IEEE Transactions on Geosciences and Remote Sensing*, 32(1):25–34, 1994.
- [7] P. A. Humblet. A distributed algorithm for minimum weight directed spanning tree. *IEEE Transactions on Communications*, COM-31(6):756–762, 1983.
- [8] N. D. Memon and K. Sayood. Lossless image compression - a comparative study. In *Still Image Compression*, pages 8–20. SPIE Proceedings Volume 2418, 1995.
- [9] N. D. Memon, K. Sayood, and S. S. Magliveras. Efficient scan patterns for image decorrelation. Proceedings of the Thirty First Annual Allerton Conference on Communications Control and Computing, September 1993.
- [10] N. D. Memon, K. Sayood, and S. S. Magliveras. Lossless compression of multispectral image data. *IEEE Transactions on Geosciences and Remote Sensing*, 32(2):282–289, March 1994.
- [11] Majid Rabbani and Paul W. Jones. *Digital Image Compression Techniques*, volume TT7 of *Tutorial Texts Series*. SPIE Optical Engineering Press, 1991.

- [12] A. K. Rao and S. Bhargava. Multispectral data compression using inter-band prediction. In *Proceedings of the Industry Workshop on DataCompression*. Snowbird, Utah, 1994.
- [13] J. A. Saghri, A. G. Tescher, and J. T. Reagan. Practical transform coding for multispectral imagery. *IEEE Signal processing Magazine*, 12(1):32–43, 1995.
- [14] S. R. Tate. Band ordering in lossless compression of multispectral images. In *Proceedings of the Data Compression Conference*, pages 311–320. IEEE Computer Society, 1994.

A Comparison of Model-Based VQ Compression with other VQ Approaches

0302
P.10

Mareboyana Manohar

Department of Computer Science
Bowie State University
Bowie, MD 20715

James C. Tilton

Information Science and Technology Branch
NASA Goddard Space Flight Center
Greenbelt, MD 20771

Abstract. In our pervious work on Model-Based Vector Quantization (MVQ)[1], we presented some performance comparisons (both rate distortion and decompression time) with VQ and JPEG/DCT. In this paper, we compare the MVQ's rate distortion performance with Mean Removed Vector Quantization (MRVQ) and include our previous comparison with VQ. Both MVQ and MRVQ compute the mean of each vector (raster-scanned image block) and produce mean removed residual vectors by subtracting the block mean from the elements of each vector. In the case of MRVQ, a codebook of residual vectors is generated using a training set. For MVQ, an internal codebook is generated based on the statistical properties of the residual vectors, and upon correlations derived from a Human Visual System (HVS) model. In both MVQ and MRVQ, the block mean and address of the codevector from the codebook that most closely matches each input vector are transmitted to the decoder. In MVQ, a single additional statistical parameter is transmitted to the decoder. For MRVQ, we assume that the codebook of residual vectors is available to the decoder. In our experiments, we found that the rate distortion performance of MVQ is almost always better than VQ, and is comparable to MRVQ. Further, MVQ is much easier to use than either VQ or MRVQ, since the training and management of codebooks is not required.

1. INTRODUCTION

Over the past several years, Vector Quantization (VQ) has been advanced as an image compression approach for large image data archives such as those being developed at various NASA centers [2,8]. NASA's distributed active archive centers, which are being built to handle data from the Earth Observing System, are expected to store some 2 to 3 Terabytes of image data products per day. Image compression will be required to minimize the stored data volumes, and to produce reduced image products for quickly browsing through the image data archive. VQ is attractive for this purpose, because of the low computational load required for decompression, of the high image quality that can be achieved at high compression ratios, and of the ease of imbedding VQ into a progressive compression system that integrates image browse with lossless retrieval of the original image data. However, there are certain difficulties with VQ codebook training and management (described below) that limit the use of VQ in image data archival, retrieval and distribution. We have developed a VQ variant called Model-based Vector Quantization (MVQ) that overcomes these difficulties with codebooks by eliminating the need for explicit VQ codebooks [1].

VQ operates on the principal that sections of image data, called vectors, can be efficiently represented by scalar indices into a list of representative image vectors. This list of representative image vectors is called a "codebook." Compression is achieved when each scalar index consists of fewer bits than the vector it represents. The best image fidelity is achieved with codebooks

containing many optimally selected image vectors. More compression is achieved with larger vector sizes and fewer codebook image vectors.

In most VQ approaches, VQ codebooks are generated through a computationally intensive training process. One of the most popular training algorithms for VQ codebook generation is the Generalized Lloyd Algorithm (GLA) [4]. See [5,6] for other training algorithms. The training process is carried out on a set of image data called the training data set. The codebook thus generated provides optimal rate distortion performance for a given size of the codebook for all the images included in the training set. However, for images outside this training set, the performance is suboptimal. Nevertheless, near optimal performance can be achieved with a carefully selected representative training set. While these issues of training data set selection and codebook generation can be hidden from the data user, they do complicate setting up the data archive.

Large codebooks are required for the best rate distortion performance in VQ compression approaches. While the compression rate (defined as the number of bits in the compressed image) for VQ approaches increases as the logarithm of the codebook size, distortion (normally measured as mean squared error between the input image and reconstructed image) decreases linearly with increasing codebook size [7]. This makes rate distortion proportional to the logarithm of codebook size divided by the codebook size, implying that rate distortion improves (*i.e.*, decreases) with increasing codebook size. Large codebooks, however, pose large computational requirements for codebook training and data encoding, and pose large codebook management requirements. Fortunately, the computational requirements can be met through powerful parallel computers at the archive site. See Manohar and Tilton [8] for a fast parallel full search implementation of VQ training and encoding. However, since the codebooks are large, we cannot transmit the codebook along with the VQ codebook indices. Thus we must manage the codebooks, presumably by labeling each codebook and transmitting them separately to each data user. A codebook label would have to be transmitted along with the VQ codebook indices to let the data user know which codebook to use. Data users would, however, see the requirement of storing several large codebooks as an unwelcome burden. This codebook management problem is a significant detraction from making VQ approaches viable options for data archival, retrieval and distribution.

The codebook management problem is reduced somewhat by the VQ variant Mean Removed Vector Quantization (MRVQ) [7, pp. 435-441]. This is because the codebooks are more generic compared to the codebooks used in standard VQ, and thus fewer codebooks would need to be stored by each data user. In MRVQ, each block of pixels from the input source is vectorized in raster scan order (\mathbf{X}) and decomposed into a block mean value (m) and a residual vector (\mathbf{R}), where $\mathbf{R} = \mathbf{X} - m\mathbf{I}$ (where \mathbf{I} is the unitary vector). In this paper we have compressed the block means using JPEG lossless compression [3] while the residual vectors are compressed using VQ as described above. However, the MRVQ codebook tends to be more consistent from data set to data set, as it has been observed that in most NASA image data sets, the statistics of the residual vectors tend to be more similar from data set to data set than the statistics of the original data. Nonetheless, while there may be fewer distinct codebooks to deal with, MRVQ still requires user involvement in the management of the residual VQ codebooks.

We have developed Model-based VQ (MVQ) [1] as a VQ variant which eliminates the need for codebook training and management. In MVQ, as in MRVQ, the mean value is calculated and subtracted from each VQ vector producing residual vectors. These mean values are losslessly compressed separately from the residual vectors, just like in MRVQ. Unlike in MRVQ, MVQ internally generates its codebook based on the Laplacian error model and imposes suitable correlations on the codebook vectors through a Human Visual System (HVS) model. The Laplacian error model requires as input the standard deviation of the residual vectors, while the HVS model requires no inputs. As far as the user is concerned, MVQ is a codebookless VQ variant.

In the following sections, we describe MVQ (including our use of the Laplacian error model and HVS model), briefly describe our implementation of VQ encoding on a massively parallel processor, and then present a performance comparison of MVQ with VQ and MRVQ.

2. MODEL-BASED VECTOR QUANTIZATION

As in most Vector Quantization (VQ) approaches, the first step in Model-based Vector Quantization (MVQ) is extracting vectors from the source or input image. In the case of two-dimensional image data, the VQ vectors are obtained by systematically extracting non-overlapping rectangular ($rx \times c = k$) blocks from the image. If the image is multispectral, non-overlapping cubes ($rx \times c \times b = k$) may be used, allowing VQ to exploit spectral as well as spatial correlations. The blocks (or cubes) are converted to vectors by performing a raster scan (band by band) of each block.

As in MRVQ, in MVQ the next step is to remove the mean from each image vector. For the i th image vector, the vector mean, m_i , is given by

$$m_i = \frac{1}{k} \sum_j x_{ij} \quad (1)$$

where x_{ij} is j^{th} element of the vector \mathbf{X}_i . These vector means are compressed separately and can be used to construct a first level browse image. In our implementation of MVQ we losslessly compress the mean vectors with the JPEG/DPCM algorithm.

Next we send the residual information in compressed form. The residual vector for i^{th} image vector is given by

$$\mathbf{R}_i = \mathbf{X}_i - m_i \mathbf{U} \quad (2)$$

where \mathbf{U} is the unit vector of same dimension as input vector \mathbf{X}_i . In MRVQ and MVQ, the residual vector, \mathbf{R}_i , is represented by an index I of the codebook (CB) entries:

$$\text{MRVQ and MVQ: } \mathbf{R}_i \rightarrow I_i \quad (3)$$

For MRVQ, the codebook is obtained by training on representative training data. It may be necessary to have more than one codebook for a particular type of data depending on image content to obtain the desired performance. For MVQ, a unique codebook is randomly generated for each image data set based on the Laplacian error model with parameter λ :

$$p(r) = \frac{1}{2\lambda} e^{-\frac{|r|}{\lambda}} \quad (4)$$

To find λ , we find the standard deviation of the residual vector elements, σ_e , and use the formula:

$$\lambda = \frac{\sigma_e}{\sqrt{2}} \quad (5)$$

The MVQ codebook is then generated by first using a random number generator to produce uniformly distributed numbers, u_l , over the range -1.0 to 1.0, which are grouped into vectors of length k . These uniformly distributed numbers, u_l , are cast into random numbers, v_l , following a Laplacian distribution through the formula

$$v_l = -\lambda \log_e (1-u_l) \quad (6)$$

where a + or - sign is randomly given to v_l . The random variables, v_l , are independent and identically distributed (i.i.d.) Laplacian random variables. However, this doesn't accurately model the residual vectors, since the elements of the residual vectors are not statistically independent, but rather have considerable correlations between them.

One method we have found that imposes the appropriate correlations on the i.i.d. Laplacian random variables is to impose the characteristics of the Human Visual System (HVS) on the variables. We use the method employed by the Joint Photographic Experts Group's Discrete Cosine Transform (JPEG/DCT) compression algorithm [3]. In JPEG/DCT, the DCT coefficients are quantized based on their significance to human perception. One way to impose HVS properties on codebook vectors generated by the Laplacian model is to DCT transform the vectors, weight the result with HVS DCT weight matrix, and inverse DCT transform the result. The HVS DCT weight matrix we have used for model generated residual vectors is shown in Figure 1. Note that we have modified the usual JPEG/DCT coefficient weighting matrix as developed by Chitparsert and Rao [9] by making the DC term 0.000 (upper left of matrix) so that it is appropriate for residual vectors with 0 mean value.

0.000	1.000	0.702	0.381	0.186	0.085	0.037	0.016
1.000	0.455	0.308	0.171	0.084	0.039	0.017	0.007
0.702	0.308	0.212	0.124	0.064	0.031	0.014	0.063
0.381	0.171	0.124	0.077	0.042	0.021	0.010	0.004
0.185	0.084	0.064	0.042	0.025	0.013	0.007	0.003
0.084	0.039	0.031	0.021	0.013	0.007	0.004	0.002
0.037	0.017	0.014	0.010	0.006	0.004	0.002	0.001
0.016	0.007	0.006	0.004	0.003	0.002	0.001	0.0006

Figure 1: The Human Visual Systems weight function of DCT coefficients.

The final version of the MVQ codebook is generated as follows. The $\{v_l\}$ from equation (6) are grouped to form set of n_c k -element vectors $\{C_i\}$. Each vector is reformatted back into a rx_c block or rx_cxb cube and then undergoes DCT transformation as given below.

$$C_i^{dct} = \text{DCT}(C_i) \quad (7)$$

For cubes, a separate DCT is performed on each band. The next step is to weight each coefficient of C_i^{dct} and take the inverse DCT transform.

$$C'_i = \text{IDCT}(C_i^{dct} * W) \quad (8)$$

where * stands for element by element product of the two matrices and W is the matrix shown in Figure 1. If the block size of C_i is smaller than 8×8 , the corresponding subset of the W matrix is used in the product. After each block or cube C'_i is reformatted back into vector form, the

used in the product. After each block or cube C'_i is reformatted back into vector form, the resulting $\{C'_i\}$ comprises the HVS weighted MVQ codebook. In the current implementation, the final MVQ codebook is obtained from the HVS weighted MVQ codebook by scaling it so that the absolute maximum vector element value is equal to the absolute maximum residual vector element value. This scale factor, s , is passed along to the MVQ decoder along with the Laplacian parameter, λ . This is done because the HVS weighting affects the scaling of the codebook vectors. However, this scaling effect should be consistent (probably related to the determinant of the HVS matrix). Once this consistency is confirmed, we will modify our encoding and decoding programs to use predetermined scaling factors instead of passing a calculated scaling factor.

The compression ratio (CR) that can be obtained from MVQ (or MRVQ) is given by

$$CR = \frac{k * b}{b + \log_2(n_c)} \quad (9)$$

where b is the number of bits per pixel in the original image data. This is actually a conservative estimate of the compression ratio, for it does not take into account that the mean values are losslessly compressed.

The MVQ decoder takes the Laplacian parameter λ , and the scale factor, s , and regenerates the model codebook. Reconstructing the residual vectors is a simple table look-up process, represented by

$$R'_i = CB(I_i) \quad (10)$$

where, R'_i is an approximation of R_i . The reconstructed vector at the decoding end is given by

$$X'_i = R'_i + m_i U \quad (11)$$

The distortion, D , between input vector, X_i and the reconstructed vector, X'_i , can be expressed in terms of mean squared error (MSE). The MSE for all the vectors drawn from the source image is given by

$$D = \sum_i \sum_j (x_{ij} - x'_{ij})^2 \quad (12)$$

In our tests, the input image was decomposed into square blocks of 4x4, 5x5, etc., depending on the targeted compression ratio. The vector size, $k (=r * c)$, for a targeted compression ratio can be computed from equation (9), where r is the number of rows and c is the number of columns of the block of pixels from the source image data. In all our test we used a codebook of 16,384 codevectors, since this matched the number of processors in the MasPar MP-2 we used for MVQ encoding.

3. IMPLEMENTATION OF MVQ ENCODING ON THE MASPAR

MVQ is an asymmetrical algorithm like all VQ approaches. While decoding is a table look-up process that can be performed quite efficiently on a sequential machine, coding using full search is

computationally very intensive. Others solve VQ coding computational problems by structuring the codebook as tree [6]. However, the price of such structuring is suboptimal performance. We instead solve the computational problem by using an implementation on a massively parallel computer to obtain optimal performance (in the rate-distortion sense) for a given codebook size by doing full search on the codebook for best match of the source residual.

We have implemented MVQ encoding on 16,384 processor MasPar MP-2. Each processing element (PE) has a local memory of 64 Kbytes. We generate 16,384 vectors of using the algorithm given in the preceding section and load each vector in each of the 16,384 PEs. We chose 16,384 vectors because we have found that this size codebook gives good performance, and because this size is most efficient on a 16,384 PE MasPar. The input image is decomposed into blocks of 4x4 or 6x6 or 8x8 pixels depending on the rate-distortion requirements and the block means are computed on the MasPar by propagating each vector from the image to one PE and the mean removed residuals from the input image are computed in parallel. The mean of the vectors are stored in a separate file and compressed using JPEG/DPCM lossless compression technique [3]. The distance between the residual vector from the input source and the codebook entries can be computed and the min distance can be found by using min function of the Maspar. The time taken for the best match search from the codebook takes time proportional to the number of bits in the Euclidean distance norm between source residual vector and codebook entry (32 bits).

If the codebook size is smaller than the PE array size by a factor of 2 or its multiple, simple load balancing techniques can be used to gain speedups for smaller codebooks. For example, if the codebook size is half the array size, then each codebook entry can be split up into two smaller codebook entries each containing half the number of elements and loaded into local memories of the adjacent PEs. The input residual vector from the source is similarly split into two smaller vectors and propagated such that first half of the vector is placed in PEs with even address and second half in those with odd address PEs. The distances are then computed in each PE separately and combined by shift operations to find the closest matching vector. Codebooks larger than PE array size by factor of 2 or its multiples can be handled by using processor virtualization.

4. COMPARISON OF MVQ RESULTS WITH VQ AND MRVQ

VQ and MRVQ require that codebooks be generated through some training process. We used a set of Landsat TM images to generate four different codebooks for different vector sizes (k 's). These codebooks can be used to compress TM image to the desired compression ratio. We assume that all these codebooks are available to the decoder to reconstruct the images. We used Generalized Lloyd Algorithm (GLA) to construct these codebooks. In MVQ, the codebook is generated using a model that needs single parameter (λ) derived from the image and the HVS weight matrix (which is not input image dependent). Thus, we need to send only one parameter (λ) to the decoder, so that the decoder generates the same codebook for reconstructing the image. In Figure 2 we see that MVQ's rate distortion performance (Compression Ratio (CR) vs. Mean Squared Error (MSE)) is better than VQ. That is for a given CR, MVQ has lower distortion (MSE) compared to VQ. However, MRVQ performs marginally better than MVQ. In Figure 3, the test image and the compression results are shown. Through inspecting these image one can see the improvements in the visual quality of the MVQ compared to VQ and that the MVQ reconstructed quality is nearly as good as the result of MRVQ. These visual comparisons are more obvious in Figure 4, where the results of VQ, MRVQ and MVQ are given for a larger compression ratio.

Considering how close MVQ's rate distortion performance is to VQ and MRVQ, the only drawback for using MVQ is the computational burden of computing the codebook entries upon image decoding. However, this computational burden is constant for any image size. For

decoding large images, this constant term becomes negligibly small. The elimination of the problems arising from training and managing codebooks makes MVQ preferable to VQ or MRVQ image archive, retrieval and distribution applications.

ACKNOWLEDGMENT

This research was partially supported by NASA grant, NAG 5-2829. The authors wish to thank Kavitha Havanagi for programming support.

REFERENCES

1. M. Manohar and J. C. Tilton, "Model-Based VQ for Image Data Archival, Retrieval and Distribution," Visual Information Processing IV, Proc of the SPIE International conference, Orlando, FL, April 17-18, 1995, pp. 190-196.
2. J. C. Tilton, M. Manohar, and J. A. Newcomer, "Earth Science Data Compression Issues and Activities," *Remote Sensing Reviews*, Vol 9, 1994, pp. 271-298.
3. W. Pennebaker and J. Mitchell. *JPEG Still Image Data Compression Standard*, pp. 29-64, Van Nostrand Reinhold, 1993.
4. Y. Linde, A. Buzo and R. M. Gray. "An Algorithm for Vector Quantizer Design," *IEEE Trans. on Communications*, Vol. COM-28, pp. 84-95, 1980.
5. T. Kohonen. "The Self-Organizing Map," *Proc. of the IEEE*, Vol. 78, No. 9, pp. 1464-1480, 1990.
6. P. A. Chou, T. Lookabaugh and R. M. Gray. "Optimal Pruning with Application to Tree-Structured Source Coding and Modeling," *IEEE Trans. on Information Theory*, Vol. IT-35, pp. 299-315, 1989.
7. A. Gersho and R. M. Gray. *Vector Quantization and Signal Compression*, pp. 309-406, Kluwer Academic Publishers, 1992.
8. M. Manohar and J. C. Tilton. "Progressive Vector Quantization on a Massively Parallel SIMD Machine with Application to Multispectral Image Data," Accepted for publication in *IEEE Trans. on Image Processing*.
9. B. Chitprasert and K. R. Rao. "Human Visual Weighted Progressive Image Transmission," *IEEE Trans. on Communications*, Vol. 38, No. 7, pp. 1040-1044, July 1990.

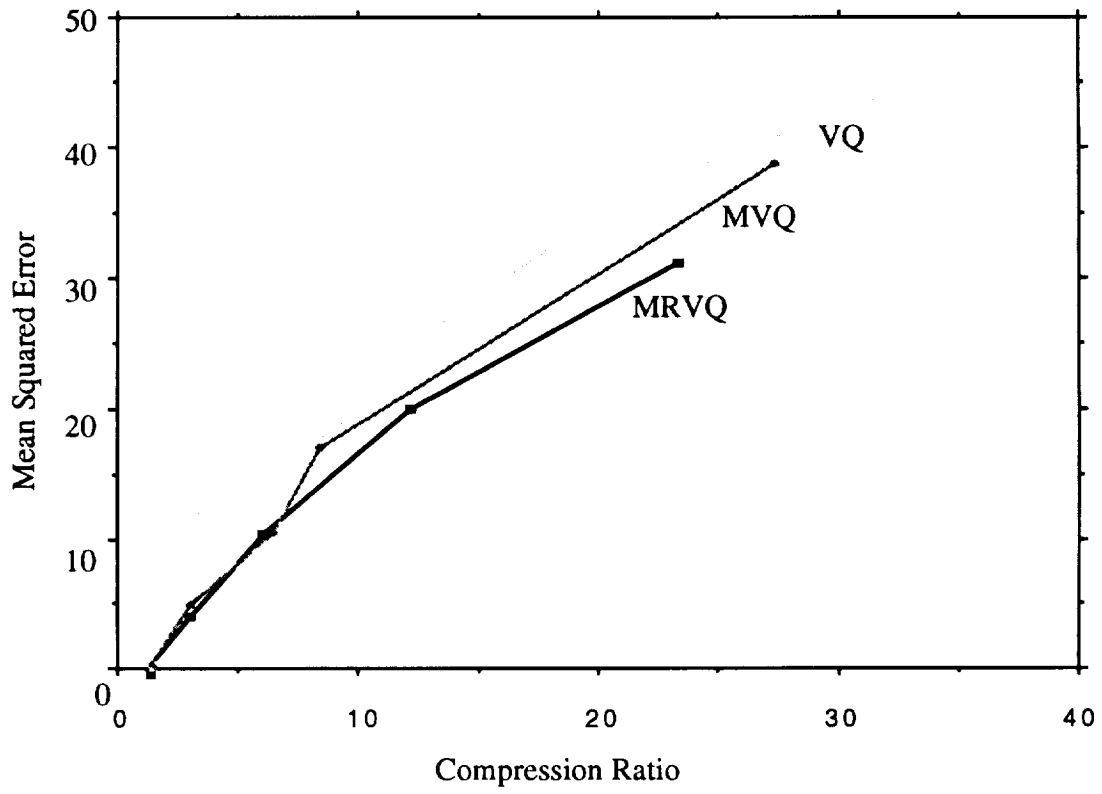
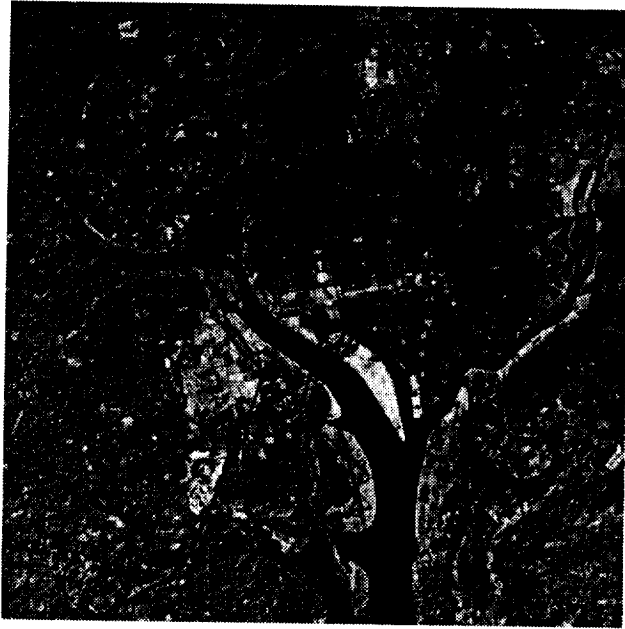
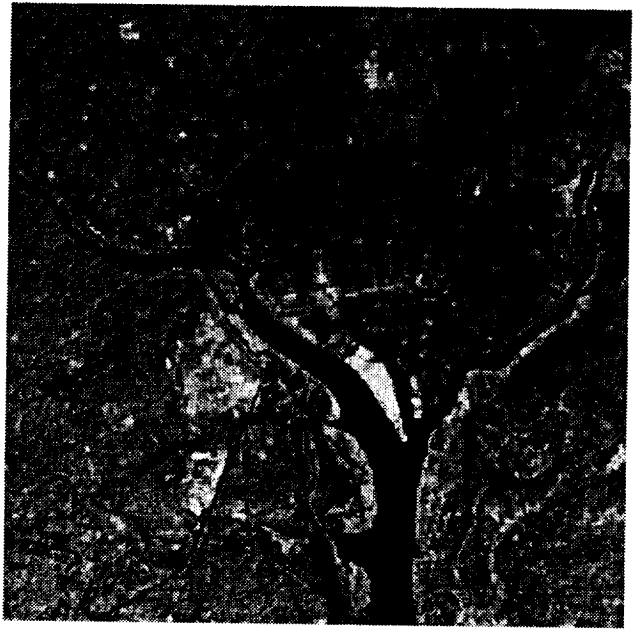


Figure 2. Rate-Distortion Performance of VQ, MVQ, and MRVQ



(a)



(b)



(c)



(d)

Figure 3. Visual quality of the reconstructed images from VQ, MRVQ and MVQ compression approaches. (a) Landsat TM image. (b) VQ: rate = 0.85 bits/pixel, mse = 18.83. (c) MRVQ: rate = 1.32 bits/pixel, mse = 10.42. (d) MVQ: rate = 1.23 bits/pixel, mse = 10.49.

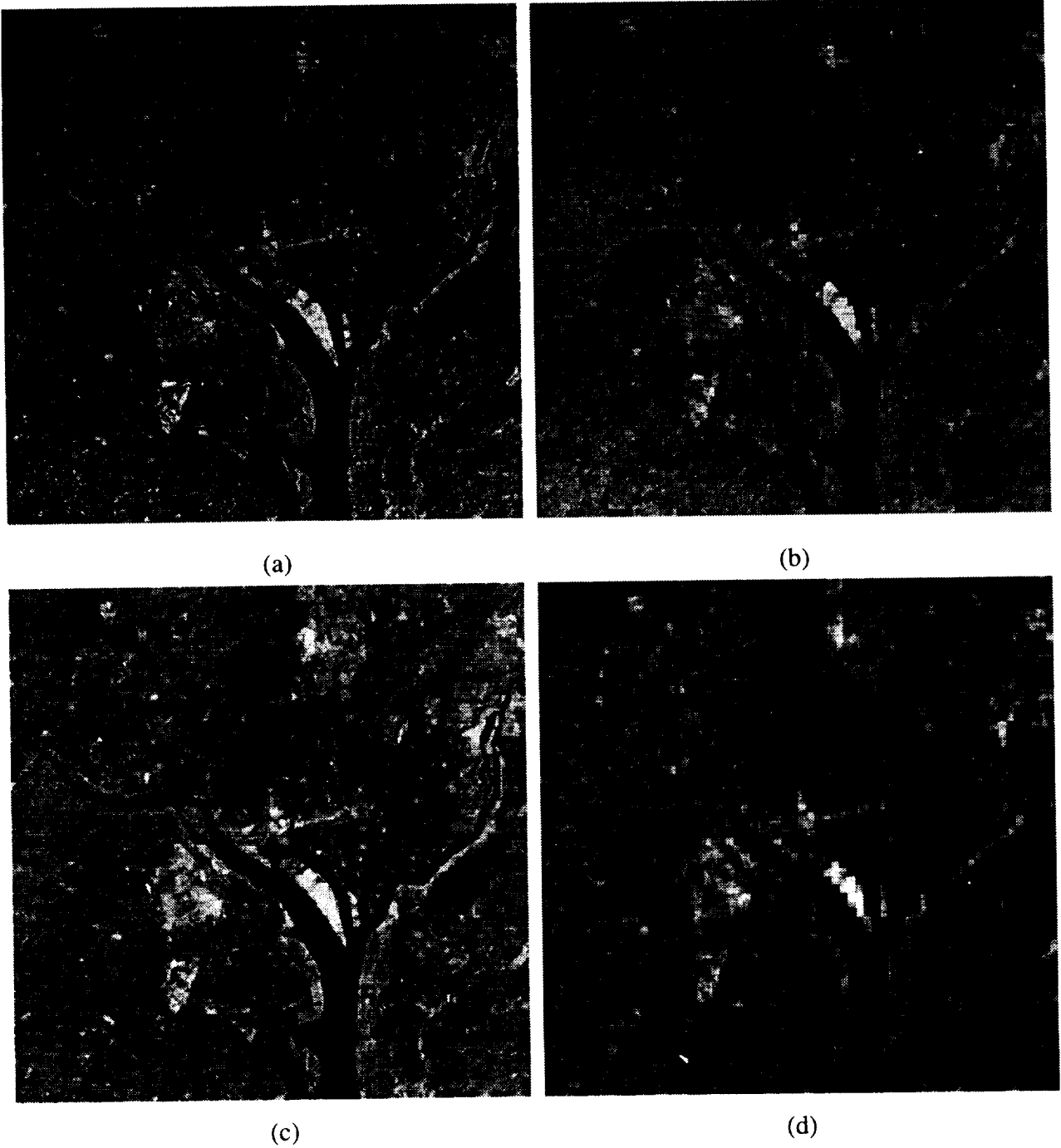


Figure 4. Visual quality of the reconstructed images from VQ, MRVQ, and MVQ compression approaches. (a) Landsat TM image. (b) VQ: rate = 0.25 bits/pixel, mse = 43.65. (c) MRVQ: rate = 0.34 bits/pixel, mse = 31.18. (d) MVQ: rate = 0.30 bits/pixel, mse = 38.75.

Remotely Sensed Image Compression Based on Wavelet Transform

Seong W. Kim*, Heung K. Lee*, Kyung S. Kim**, Soon D. Choi***

Department of Computer Science, KAIST*

Satellite Technology Research Center, KAIST***

Software Engineering Research Institute, KIST**

373-1 Kusong-Dong, Yusong-Ku, Taejon 305-701, Korea

In this paper, we present an image compression algorithm that is capable of significantly reducing the vast amount of information contained in multispectral images. The developed algorithm exploits the spectral and spatial correlations found in multispectral images. The scheme encodes the difference between images after contrast/brightness equalization to remove the spectral redundancy, and utilizes a two-dimensional wavelet transform to remove the spatial redundancy. The transformed images are then encoded by hilbert-curve scanning and run-length-encoding, followed by huffman coding. We also present the performance of the proposed algorithm with the LANDSAT MultiSpectral Scanner data. The loss of information is evaluated by PSNR(peak signal to noise ratio) and classification capability.

1 INTRODUCTION

In the remote sensing field, one way to handle a huge image data is through on-board data compression in the satellite or through archival purpose compression in the ground-station. But, in spite of considerable progress in data compression research field, the image compression of either on-board data handling or archiving has very seldom been used because of risks in reliability and data alterations.

There are mainly two kinds of coding schemes. One is entropy coding which is lossless and the other is transform coding which is lossy. But we can not hardly gain a compression gain with the lossless coding, and thus we have to design a transform coding which retains both good subjective quality and good classification capability. Recently, wavelet transform is being studied for the efficient computation time and the energy preserving capability for image data. In this paper, an image compression system is presented with wavelet transform for the remote sensing. Since the satellite sends many images per second, the compression is indispensable to send the image through microwave channel and to archive the image on the ground station.

In this paper, we present an image compression algorithm that is capable of significantly reducing the vast amount of information contained in multi-spectral images. The developed algorithm exploits the spectral and spatial correlations found in multi-spectral images. The proposed scheme encodes the difference between images after contrast/luminance equalization to remove the spectral redundancy, and utilizes a two-dimensional wavelet transform to remove the spatial redundancy. The transformed images are encoded using Hilbert-curve scanning and run-length-encoding, followed by Huffman coding. We will conclude by presenting the performance of the

proposed algorithm for LANDSAT multi-spectral scanner data. The loss of information is evaluated by PSNR(peak signal to noise ratio) and classification capability.

Section 2 explains the wavelet transforms used in this paper. After a general review of wavelets, we extend one-dimensional construction to a two-dimensional scheme with separable filters. A new coding scheme is then presented in Section 3. Here we focused on contrast/brightness equalization techniques to maximize the correlations between multispectral bands and the Hilbert curve scanning method to maximize the length of zero run after wavelet transform. Experimental results are given in Section 4 for LANDSAT multispectral scanner data and KITSAT-1 sample images. We conclude in Section 5.

2 WAVELET TRANSFORMS

In this section, we introduce basic ideas of wavelet transform and explain the application of wavelet transform to image processing.

2.1 Wavelet analysis

Wavelets are functions generated from one single function ψ by dilations and translations $\psi^{a,b}(t) = \frac{1}{|a|^{1/2}} \psi(\frac{t-b}{a})$, where t is an one dimensional variable. Figure 1 shows the dilated and translated functions. The definition of wavelets as dilates of one function means that high frequency wavelets

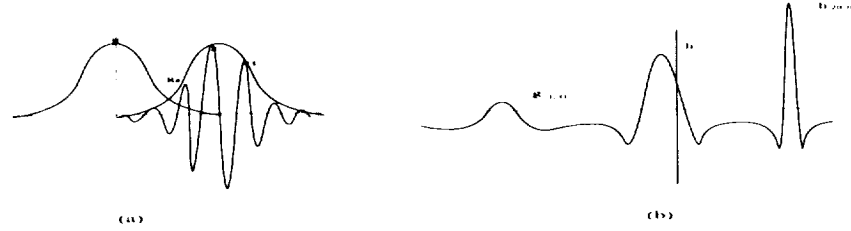


Figure 1: Basis of windowed fourier transform(a) and wavelet transform(b)

correspond to $a < 1$ or narrow width, while low frequency wavelets match $a > 1$ or wider width. If we depict the wavelet basis in time-frequency(x-k) plane, it looks like Figure 2(b). We compared wavelet transform with windowed fourier transform as shown in Figure 2(a). The main difference between the two transforms is the variable sliding window in the wavelet transform. For higher frequencies, the window size in time decreases(better time localization) while its size in the frequency increases(broader frequency localization).

The basic idea of the wavelet transform is to represent any arbitrary function f as a superposition of wavelets. Any such superposition decomposes f into different scale levels, where each level is then further decomposed with a resolution adapted to the level. One way to achieve such a decomposition is that we write f as an integral over a and b of $\psi^{a,b}$ with appropriate weighting coefficients. In practice, one prefers to consider f as a discrete superposition. Therefore, one introduces a discretization, $a=a_0^m, b=nb_0a_0^m$, with $m, n \in \mathbb{Z}$, and $a_0 > 1, b_0 > 0$ fixed. The wavelet decomposition is then $f = \sum c_{m,n}(f)\psi_{m,n}$ with $\psi_{m,n}(t) = \psi^{a_0^m, nb_0a_0^m}(t) = a_0^{-m/2}\psi(a_0^{-m}t - nb_0)$. For $a_0 = 2, b_0 = 1$ there exist very special choices of ψ such that the $\psi_{m,n}$ constitute an orthonormal basis, so that $c_{m,n}(f) = \langle \psi_{m,n}, f \rangle = \int \psi_{m,n}(x)f(x)dx$ in this case.

The construction in Daubechies's works⁴ gives ψ with finite support, and therefore, corresponds

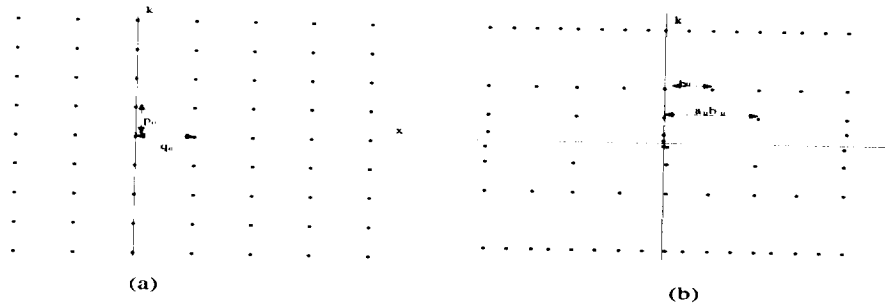


Figure 2: Time-Frequency plane configuration for windowed fourier transform(a) and wavelet transform(b)

to FIR filters. It follows that the orthonormal bases correspond to a subband coding scheme with exact reconstruction property, using the same FIR filters for reconstruction as for decomposition. The extra ingredient in the orthonormal wavelet decomposition is that the signal is to be decomposed as a superposition of reasonably smooth elementary building blocks.⁴

2.2 Biorthogonal wavelet bases

Since images are mostly smooth(except for occasional edges) it seems appropriate that an exact reconstruction subband coding scheme for image analysis should correspond to an orthonormal basis with a reasonably smooth mother wavelet. In order to obtain fast computation, the filter should be short(short filters lead to less smoothness, so they cannot be too short). Unfortunately, there are no nontrivial orthonormal linear phase FIR filters with the exact reconstruction property, regardless of any regularity considerations.

One can preserve linear phase(corresponding to symmetry for the wavelet) by relaxing the orthonormality requirement and using biorthogonal bases. It is then still possible to construct examples where mother wavelets have arbitrarily high regularity. In such a scheme, decomposition is same as orthonormal case, but reconstruction becomes $a_{m-1,l}(f) = \sum_n [\tilde{h}_{2n-l} a_{m,n}(f) + \tilde{g}_{2n-l} c_{m,n}(f)]$, where the filter \tilde{h}, \tilde{g} may be different from h, g .

2.3 Extension to the two-dimensional case: image analysis

There exist various wavelets from one-dimensional wavelet transform to higher dimensions. We use a two-dimensional wavelet transform in which horizontal and vertical orientations are considered preferentially.

In the two-dimensional wavelet analysis, one introduces, like in the one-dimensional case, a scaling function $\phi(x, y)$ such that:

$$\phi(x, y) = \phi(x)\phi(y)$$

where $\phi(x)$ is an one-dimensional scaling function. Let $\psi(x)$ be one-dimensional wavelet associated with the scaling function $\phi(x)$. Then, the following three two-dimensional wavelets are defined as:

$$\begin{aligned} \psi^H(x, y) &= \phi(x) \times \psi(y) \\ \psi^V(x, y) &= \psi(x) \times \phi(y) \\ \psi^D(x, y) &= \psi(x) \times \psi(y) \end{aligned}$$

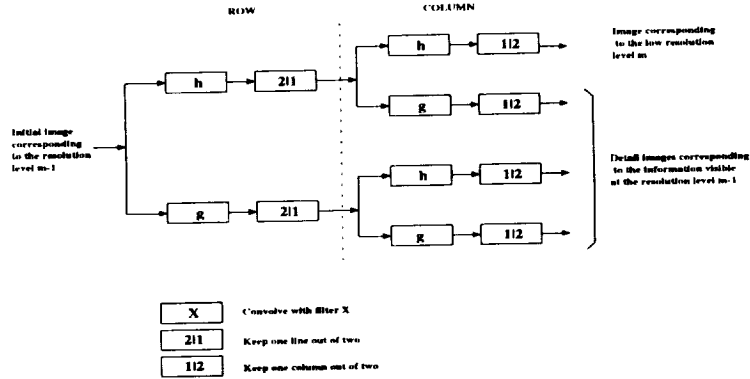


Figure 3: One stage in a multiscale image decomposition¹

Figure 3 represents one stage in a multiscale pyramidal decomposition of an image. The filter h and g are one-dimensional filters. This decomposition provides subimages corresponding to different resolution levels and orientations. The reconstruction scheme of the image can be done reversely.

3 MULTISPECTRAL IMAGE CODING

In this section, we will identify the basic requirements of the satellite multispectral image coding scheme, and present the our proposed scheme to encode the images efficiently.

3.1 Requirements

Because we are considering both on-board image transmission system and ground station archival system, we must consider two requirements. One is processing speed, and another is high compression ratio. Currently we must process 1 image per second for the KITSAT-1 and 2, which are the first and second satellites in KOREA. They were launched in 1992, 1993, respectively. But, In this paper, we will concentrate on compression issues because more powerful processor can be available in the near future.

We designed the multispectral image coding scheme based on wavelet transform. In figure 4, we can see the typical multispectral image from LANDSAT and unispectral image from KITSAT-1. Figure 4(a)(b)(c)(d) shows the multispectral image of Ichon area of Korea and figure 4(e)(f) shows the image of Italy and Korea.

To obtain the coding efficiency, we should exploit the redundancies in the multispectral image and consider the response of HVS(human visual system) to the elimination of redundancies. There are three redundancies in the multispectral images: spatial redundancy, spectral redundancy, and statistical redundancy. In short, spatial redundancy means the similarity between neighboring pixels in an image. Spectral redundancy is said to be the inter-channel similarity. And, statistical redundancy is the inter-symbol similarity in the lossless coding. There are many mechanisms to reduce these redundancies. So the hybrid coding scheme to reduce the redundancy should be used.

We use wavelet transform to reduce the spatial redundancy, contrast and brightness equalization between channels for spectral redundancy, and Hilbert-curve scanning and Huffman coding for statistical redundancy. The block-diagram of the proposed coder block is shown in Figure 5. We will explain each component in detail in the following.

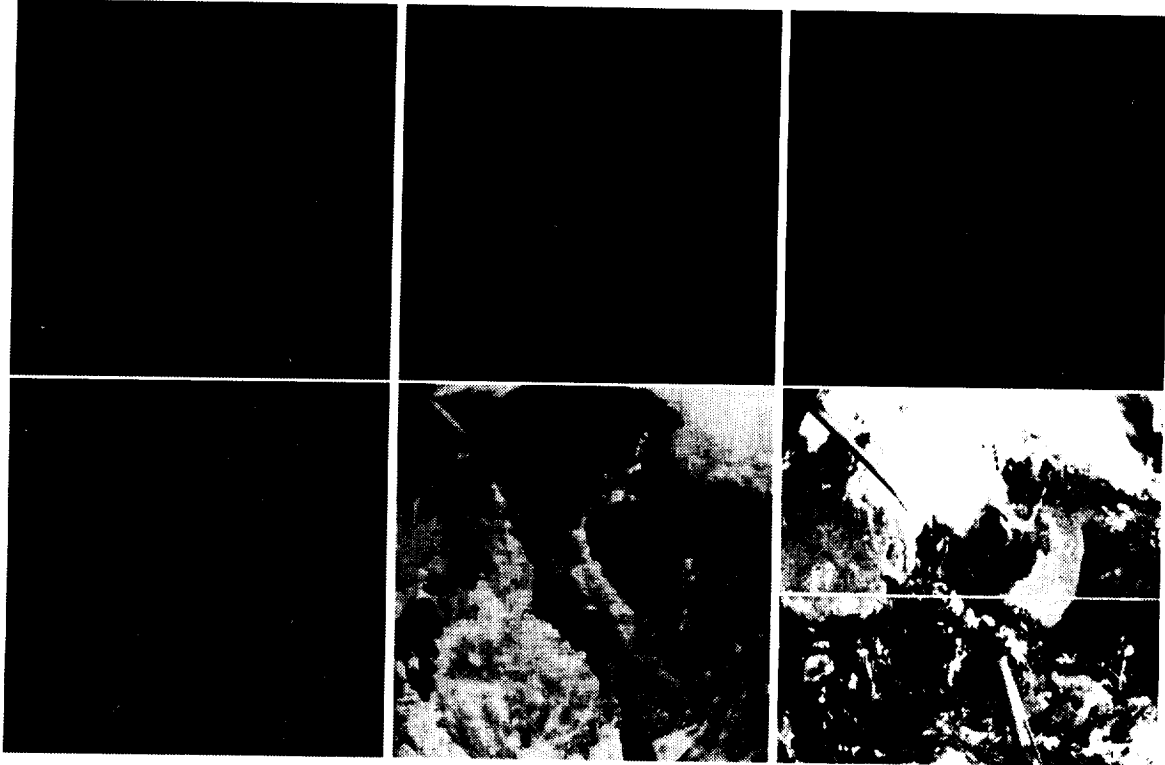


Figure 4: test image: LANDSAT multispectral image: (a) subch1.bin, (b) subch2.bin,(c) subch3.bin,(d) subch4.bin, KITSAT-1 image: (e) KITSAT-1(Italy), (f) KITSAT-1(Korea)

3.2 Wavelet transform and Encoding of wavelet coefficients

In our multi-spectral image coder, we use Daubechies 9-3 filter¹ to transform the frames. Figure 6 shows the scaling function and wavelet for the encoder and decoder, respectively. The filter coefficients are shown in Table 1. After wavelet transform, many coefficients of high frequency

n	0	± 1	± 2	± 3	± 4
$2^{-1/2}h_n$	45/64	19/64	-1/8	-3/64	3/128
$2^{-1/2}\tilde{h}_n$	1/2	1/4	0	0	0

Table 1: Filter coefficients for the Daubechie 9-3 filter

components would be zero. Figure 7(b) shows a sample wavelet transformed image. There is the low resolution level subimage on the top of the image and other high resolution details which shows high frequency components of the image.

Because HVS is less sensitive to the high frequency components, less bits can be assigned to the high frequency bands. Moreover, because the HVS is insensitive to the diagonal frequency components, we can save bits in diagonal frequency bands. Thus we drop off the highest diagonal frequency components. The resulting quantization steps for each frequency band are shown in Figure 7(c). Also, Because the HVS is sensitive to motion and insensitive to the changes in the luminance in the border of images, less bits are assigned to the border.

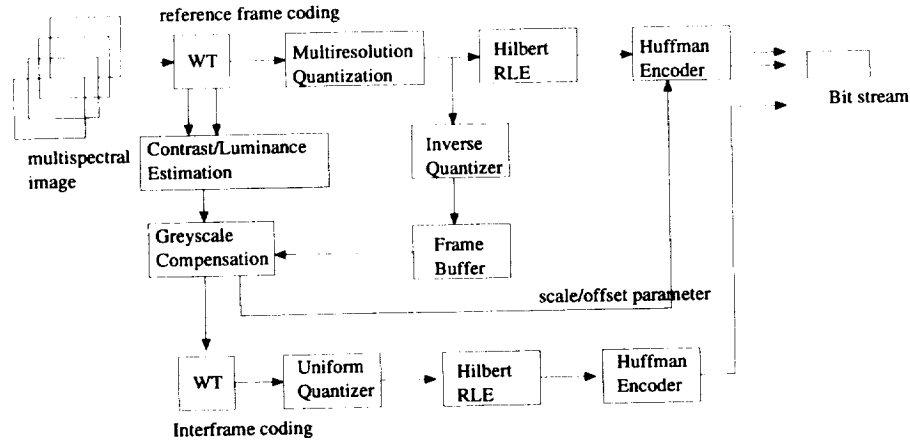


Figure 5: Multi-spectral Image Coder

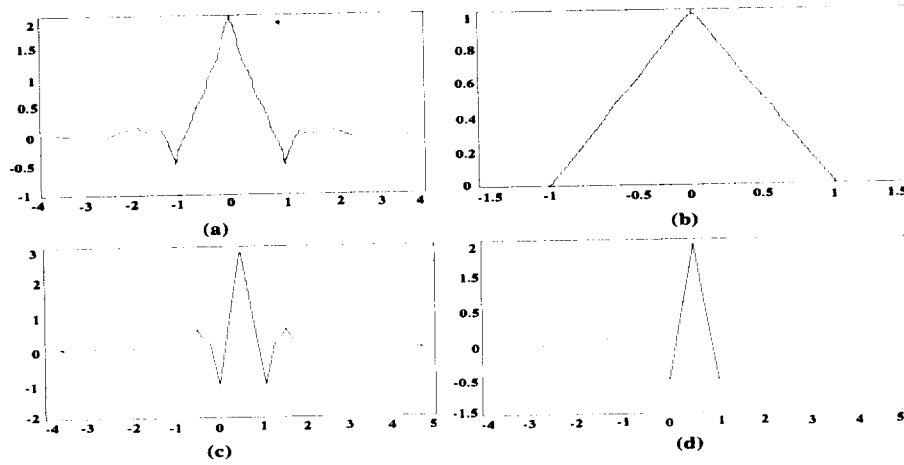


Figure 6: Scaling functions $\phi, \tilde{\phi}$ and wavelets $\psi, \tilde{\psi}$ (a)Scaling functions ϕ ,(b)Scaling functions $\tilde{\phi}$, (c)Wavelet ψ ,(d)Wavelet $\tilde{\psi}$

Wavelet transform decomposes an image into a set of sub-frames with different resolutions corresponding to different frequency bands. This multifrequency nature of frames provides an efficient representation for coding because there are many 0-coefficients in the high frequency bands for natural images. We can compress a frame more compactly if we can maximize the length of 0 run. Hilbert-curve-scanning method is used (Figure 8) instead of line-by-line scanning method to maximize it. To apply for the nonsquare image, we used the modified Hilbert curve scanning.

3.3 Contrast/Brightness Equalization Between Channels

To improve the coding efficiencies, we can use the spectral redundancy between spectral bands in a multi-spectral image as shown in figure 4. Multi-spectral image seems to be very similar to the motion pictures in video coding field. It is noticed that motion estimation for the the moving objects is important in the video coding applications, which is not for the multi-spectral image. Instead, In the multi-spectral image, it is important to equalize the brightness of each channels, so

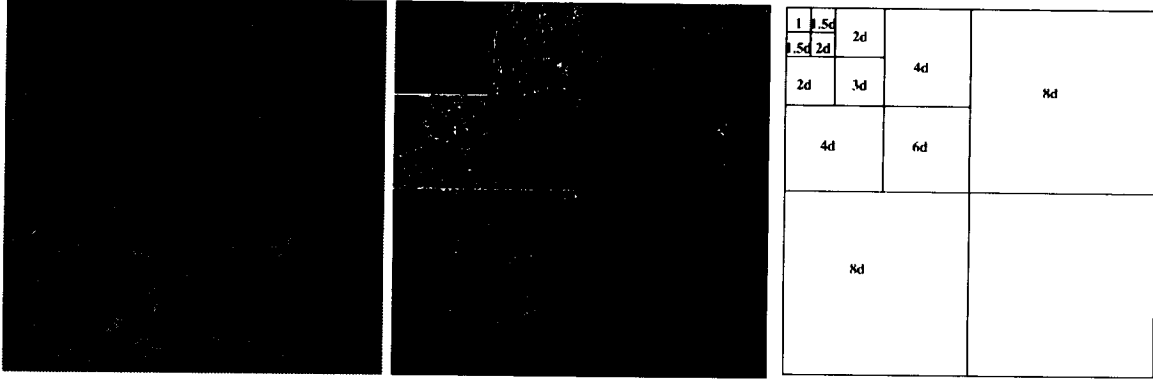


Figure 7: (a) Sample image (b) Wavelet Transform of Sample image (white points means the values above 8) (c) bit allocation table for each frequency bands

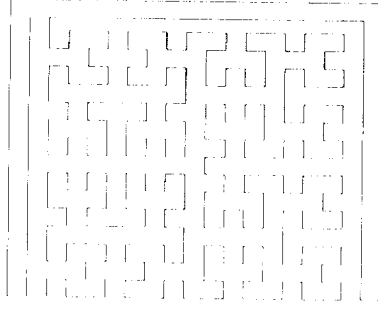


Figure 8: Hilbert curve scanning

that we can take advantage of the minimized difference.

In our scheme, we take the image with the smallest variance in brightness as a reference image. For other images, we equalized the image with the reference image. We used a simple linear model for the brightness, $b_i = sa_i + o$, where b_i is a pixel value of reference image, and a_i is a corresponding pixel value of the matched image. Our problem is to minimize the $R = \sum_{i=1}^n (sa_i + o - b_i)^2$.

We can define the problem formally as follows. Given two squares containing n pixel intensities, a_1, \dots, a_n from the matched image and b_1, \dots, b_n from the reference image. We can seek s and o to minimize the quantity

$$R = \sum_{i=1}^n (sa_i + o - b_i)^2$$

This will give us a contrast and brightness setting that makes the matched block image a_i values have the least square distance from the b_i values. The minimum of R occurs when the partial derivatives with respect to s and o are zero, which occurs when

$$s = \frac{[n^2(\sum_{i=1}^n a_i b_i) - (\sum_{i=1}^n a_i)(\sum_{i=1}^n b_i)]}{[n^2 \sum_{i=1}^n a_i^2 - (\sum_{i=1}^n a_i)^2]}$$

$$o = \frac{[n^2 \sum_{i=1}^n b_i - s \sum_{i=1}^n a_i]}{n^2}$$

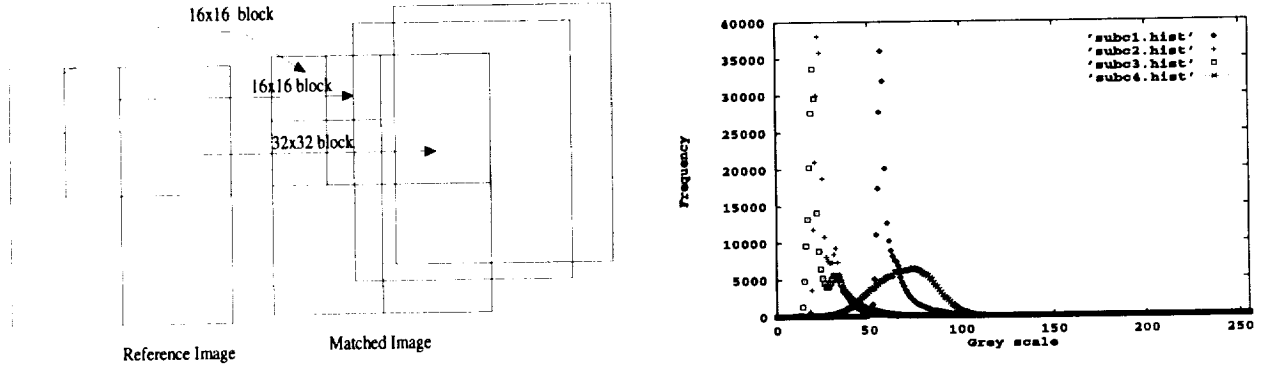


Figure 9: Multi-spectral Channel Equalization Between Spectral Bands (a) scheme (b) pixel value histogram for sample multi-spectral image

In that case,

$$R = \left[\sum_{i=1}^n b_i^2 + s \left(s \sum_{i=1}^n a_i^2 - 2 \left(\sum_{i=1}^n a_i b_i \right) + 2o \sum_{i=1}^n a_i \right) + o \left(on^2 - 2 \sum_{i=1}^n b_i \right) \right] / n^2$$

The s and o estimated for lower frequency band are predicted with more specific details and higher frequency bands with less details as shown in Figure 9. The s and o in the highest diagonal band are not estimated because it is insensitive to HVS.

After s and o estimation, equalized error image is constructed from differencing matched image and decoded reference frame to make sure same operation at the decoder side. Because HVS is less sensitive to the changes in the high frequency area, we give less bits for the high frequency components of equalized error-frame.

3.4 Progressive transmission

The main objective of progressive transmission is to allow the receiver to recognize a picture as quickly as possible at minimum cost, by sending a low resolution level picture first. Then it can be decided to receive further picture details or to abort the transmission. If necessary, further details of the picture are obtained by receiving the encoded wavelet coefficients at different resolution levels and directions. The proposed scheme is suitable for progressive transmission because of its multifrequency nature. Our coding scheme is very well suited for this situation.

4 EXPERIMENTAL RESULTS

We have implemented a multispectral image codec for satellite imaging. LANDSAT multispectral multiscanner image and KITSAT-1 sample image as shown in figure 4 is used for test images. Table 2 shows the specification for our test images.

The multispectral image codec is composed of three components: reference frame coding part, contrast and brightness estimation part, and brightness equalized error frame coding part. Thus, each of quality measure is suggested for individual components and the performance of each measure is given. After that, we will show the coding rate and image quality for each test images.

image	sensitivity(μm)	resolution(m)
LANDSAT(subc1.bin)	0.5-0.6	82
LANDSAT(subc2.bin)	0.6-0.7	82
LANDSAT(subc3.bin)	0.7-0.8	82
LANDSAT(subc4.bin)	0.8-1.1	82
KITSAT(kaiw0008.bin)	wide view	low
KITSAT(kaiw000e.bin)	wide view	low

Table 2: Characteristics of test multispectral images

4.1 Evaluation of reference frame coding

The reference frame coding scheme is evaluated in rate-distortion. bit per pixel(bpp) is used for rate. There are two distortion measures: objective quality such as mean square error between two images and subjective quality concerned with HVS. We will evaluate the objective quality by mean square error and the subjective error by k-means clustering based classification capability. The following measure is used for evaluating objective quality. Here, $f(i,j)$, $r(i,j)$ means the pixel value of original image and reconstructed image at i,j position respectively.

$$PSNR(\text{peak signal to noise ratio}) = 10 \log_{10} \frac{255^2}{(1/255)^2 \sum_{i=0}^{255} \sum_{j=0}^{255} [f(i,j) - r(i,j)]^2}$$

where $f(i,j)$ is original image and $r(i,j)$ is reconstructed image

A test images are compressed with the reference coding scheme, and the results are shown in Figure 10(a) for the LANDSAT multispectral image. The presented scheme has shown good image quality under 0.40 bpp(bit per pel) at average. Figure 10(b) shows the decoded subc2.bin image under 0.25 bpp.

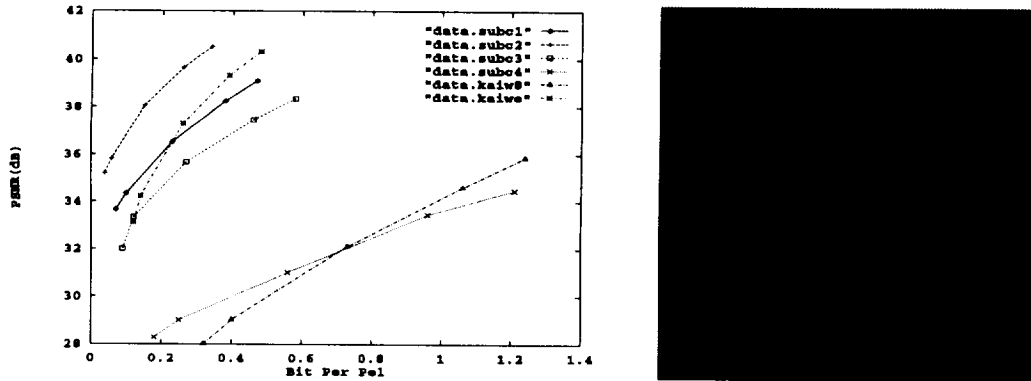


Figure 10: (a) Rate-distortion for LANDSAT image, (b) Decoded subch2.bin image under 0.25 bpp

4.2 Evaluation of Equalized Error Image Coding

We used 3 level wavelet transform to equalize the two images, with the block size 16 by 16, 32 by 32, 64 by 64 for each bands, respectively. We used 2 bytes for representing contrast(s) and brightness(o) estimation factor as an overhead information. After the equalization step, the difference between reference image and matched image has bounded in small dynamic range.

We also performed K-means classification on decoded image, because classification or subpixel analysis is popular techniques in the remote sensing field. We used IRIS PCI/EASI analysis tool for K-means clustering. For details, we used 16 class K-means clustering, and the number of miss classified pels is 37925 out of 512 by 512 resolution images. Figure 11(a) shows the classification result for the original image and Figure 11(b) shows the result for the decoded image.

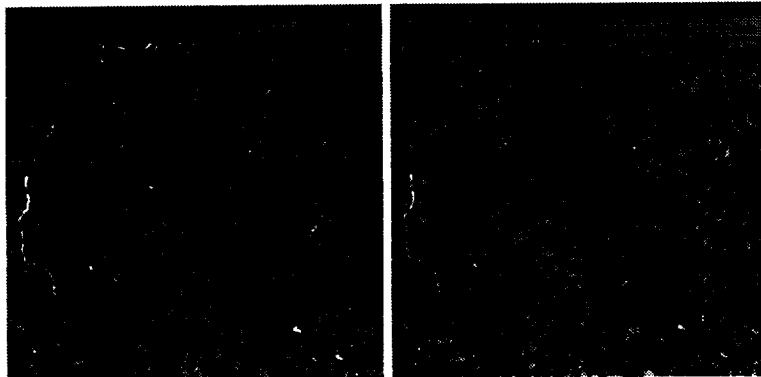


Figure 11: (a) Clustering result for the original image, (b) Clustering result for the decoded image

4.3 Computational complexity

The multispectral image codec have two time consuming jobs: wavelet transform and contrast and brightness estimation step. The time complexity of the wavelet transform is $O(n \log n)$ and time cost of estimation step is negligible.

5 CONCLUSIONS

We have presented multispectral image coding scheme for satellite image transmission system. It is shown that the wavelet transform based scheme is well applied to encode the satellite image and the contrast/luminance equalization between the wavelet transformed images can be used for more effective coding of a multispectral image. Our multispectral image coding scheme is evaluated for LANDSAT multispectral scanner data and KITSAT-1 images.

It is also shown that wavelet transform coding approach has a good performance in terms of the objective quality(PSNR) and subjective quality(classification capability).

6 ACKNOWLEDGEMENTS

It is our pleasure to acknowledge and thank all the people who have influenced us through the course of this research. This work was supported by CRAY Research, Inc. under the University Research & Development Grant Program in 1995.

7 REFERENCES

1. M. Antonini, M. Barlaud, and et.el., "Image Coding Using Wavelet Transform," *IEEE Transactions on Image Processing* 1(2), April 1992.

2. T. Markas and J. Reif, "Multispectral Image Compression Algorithms," *Proc. Data Compression Conference*, Apr.1993, pp. 391-400.
3. J.M. Combes, A. Grossman, and Ph.Tchamitchian(ed), "Wavelets: Time Frequency Methods and Phase Space(2/e)," *Springer-Verlag*, 1989.
4. I. Daubechies, "Othonormal Bases of Compactly Supported Wavelets," *Communications on Pure and Applied Mathematics XLI*, 1988.
5. I. Daubechies, "Time-Frequency Localization Operators: A Geometric Phase Space Approach," *IEEE Trans. on Information Theory 38(4)*, July 1988.
6. I. Daubechies, "The Wavelet Transform, Time-Frequency Localization and Signal Analysis," *IEEE Transactions on Information Theory 36(5)*, Sep. 1990.
7. R.A. Devore, B. Jawerth and B.J. Lucier, "Image Compression Through Wavelet Transform Coding," *IEEE Transactions on Information Theory 38(2)*, March 1992.
8. J.C. Feauveau, P. Mathieu, B.M. Antonini, "Recursive Biorthogonal Wavelet Transform for Image Coding," in *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing, IEEE*, 1991
9. S.G. Mallat, "A Theory for Multiresolution Signal Decomposition: The Wavelet Representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence 11(7)*, July 1989.
10. J. Moik, "Digital Processing of Remotely Sensed Images," *NASA Scientific and Technical Information Branch*, 1980.
11. Lu, Y.C., "Earth Observing System Output Data Products and Input Requirements, Version 2.0," Science Processing Support Office, NASA Goddard Space Flight Center, August 1992.

A Packet Data Compressor

Mitchell R Grunes
AlliedSignal Technical Services Corp.

Junho Choi

Naval Research Laboratory

(E-Mail: grunes@nrlvax.nrl.navy.mil, choi@nrlvax.nrl.navy.mil)

5304
P. 10

ABSTRACT: We are in the preliminary stages of creating an operational system for losslessly compressing packet data streams. The end goal is to reduce costs. Real world constraints include transmission in the presence of error, tradeoffs between the costs of compression and the costs of transmission and storage, and imperfect knowledge of the data streams to be transmitted. The overall method is to bring together packets of similar type, split the data into bit fields, and test a large number of compression algorithms. Preliminary results are very encouraging, typically offering compression factors substantially higher than those obtained with simpler generic byte stream compressors, such as Unix Compress and HA 0.98.

INTRODUCTION

The creation of complex technical systems in the real world environment often involves a major organizational effort to integrate the equipment and data created by diverse organizations. Each organization brings its own interests and prior experience, builds its own instruments, and defines its own data stream content.

An administratively efficient solution to data stream integration is to organize the data into "packets" containing a common set of basic bit fields, and a variable format field. Each organization is free to define the structure and content of the variable format field for each of its packet types. The common elements allow hardware and software, on the platform and on the ground, to deal with all types of data in a similar manner.

A simple case might include:

Field	Bits	Content
1	4	Organization defining this packet particular type.
2	5	Packet type within that organization.
3	3	Telemetry flags indicating system overall equipment status.
4	28	Time tag for packet generation.
5	32	Variable data field.

One packet structure might define the variable format field to contain four 8 bit values, all part of the same data sequence. Another packet structure might contain three fields, with 9, 11 and 12 bits, in distinct data sequences.

The administrative efficiency is obtained at the cost of some increase in data bandwidth. In addition, some telemetry fields change rather slowly. Thus, a data compression system which

has been customized to the specific packet structures can attain relatively high compression factors. To create an optimal system one needs the structure, content and estimated statistics of the data, as well as test data sets with the appropriate statistics.

Data transmission and storage in modern remote sensing systems represents a major component of total cost, through its effects on bus bandwidth, transmit power, cost and launch (or payload) weight of power systems, the cost of transmission lines and channels, and storage requirements, on-board and on the ground. Hence, an efficient compression system can generate substantial cost savings. For some applications inexpensive off-the-shelf compressors provide adequate results. This study is directed towards those applications where the high costs of transmission and storage justify a more elaborate system. This must always be balanced against the cost and power requirements of the data compression system itself.

Our present system can be easily customized for compression and de-compression in specific applications by modifying those portions and tables defining and manipulating packet types, and which input or output data. Flexibility is important, since one may have imperfect knowledge about the format and statistics of the input data stream at the design stage of the communications system, when other design elements are still in flux.

Compressed data is more context-dependent than uncompressed data, so error correction coding (ECC) is strongly recommended for channels that transmit compressed data. One often uses ECC hardware that is off-the-shelf or embedded in the data channel. Typically the compression system can not determine when and where badly transmitted bits have been lost, so some overhead has been included to detect errors and to re-sync the data. Thus, a block of packets can be lost when an uncorrectable error occurs, but the rest of the data stream can still be recovered. This capability is extremely important.

Other complications of the real-world exist. There is a great deal of high quality freely available compression source code, but one must still deal with patents. The present system includes two such algorithms. It required a significant effort to embed stand-alone software inside our application. In an operational environment one must insure that no component can accidentally or deliberately halt or do harm to associated computers, networks and transmission channels. This involves a great deal of work, in terms of comprehending, re-writing, and simplifying the source code. One may also have to add some bounds checking for indices and pointers, and check for other potential problems. Finally, it was considered prudent to always check the de-compressed result before choosing the algorithm as best.

We impose the formal requirement that compression will cause no data to be lost or re-ordered, even if it does not match the assumed format, or is filler data. This is necessary so that the system can be properly debugged if unexpected format data is transmitted.

The runs of each packet type may not be very long, so one can not efficiently start a new run of compression processing each time the packet type changes. Therefore, the data is divided into 512 packet blocks, packets are classified by type, and the less common types are classified together as being of a generic type, which breaks out the common fields, but views the variable

format fields as a sequence of bytes. A byte stream packet type, which views the entire packet as a sequence of bytes, is used when the data does not match the assumed format.

All of the packets of a given packet type from a given block are assembled into one sub-stream, and each of their bit fields is then compressed as an independent compression sequence. In the current version, each bit field sequence is tested against 17 compression algorithms, and the best is chosen. This method introduces a delay of at least 512 packets before information can be transmitted. To solve that problem on temporarily quiescent data streams, a time-out occurs if the 512 packet buffer does not fill quickly enough, and the block is terminated after less than 512 packets.

The output compressed data packet output from each block of input packets includes:

- Sync code
- Number of original packets (usually 512)
- Compression algorithm # for packet type sequence
- Compressed packet type sequence
- For each field in the packet structure:
 - Compression algorithm # for field
 - Compressed stream of field values
- Check Sum

On reception an incorrect sync code or check sum indicates that a compressed packet has been transmitted incorrectly. If either is wrong, or another error occurs during de-compression, the block is discarded, and the receiver scans for the next sync code after the start of the bad block. Good blocks are de-compressed, and the packets are placed back into the original order.

COMPRESSION ALGORITHMS

No attempt has been made to develop radically new compression algorithms, but some improvements have been made to published algorithms, and some algorithms have been combined into hybrids.

Current algorithms include:

1. Constant Coding--If all of the values are of one constant value, that value is sent only once, and no other algorithms need be tested.
2. Constant Bit Removal--If only some of the bits are constant, a mask is sent specifying which bits are constant, and those bits are sent only once. The remaining bits are transmitted unchanged. *Constant bit removal is always applied before the remaining algorithms.*
3. Run Length Encoding--Each value is transmitted with its repeat count. As an improvement to the algorithm, the number of bits needed to code the largest repeat count is transmitted before transmitting the run-length pairs.

4. Rice Coding--This algorithm, based on references [2] and [3], allows for very rapid adaptivity, because it transmits one adaptive parameter--the number of least significant bits of the remapped differences that may optimally be transmitted unchanged--for each Rice block (currently 32 values; could be varied). The most significant bits are re-mapped and transmitted as a terminated unary code (0 as 1, 1 as 01, 2 as 001..., with no termination required for the largest detected value). The differencing and remapping algorithms are discussed in the next section.

A fairly large amount of work went into making improvements to this algorithm. For example, two special adaptive parameter values handle the low entropy cases: one indicates that all of the differences are zero, and the other adds one value to represent four zero difference values. The adaptive codes are themselves Rice coded within each block of packets.

5. LZ77--This was based on reference [4]. It is a "dictionary search" algorithm intended to be used data having repeated strings of values. A window of prior values is searched for the longest string of values matching the current and future values. The backwards distance of the best match (or a flag value for new values not matching prior values) is transmitted, as is the number of matching values in the string. Flag values are prefixed before transmittal of values not previously sent.

Improvements were made to the algorithm: an elaborate adaptive technique is used to determine the adaptive window size, based on prior matches, with periodically increased sizes, and to determine the maximum string length. These sizes are rounded up to fit into an integral number of bits, and each value is transmitted in the minimum number of bits. Linked lists are used to speed up processing.

6. LZ77 Applied to Differences--The same algorithm is applied to the remapped differences.
7. LZRW3A--This dictionary search algorithm, based on reference [6], is a LZW-family software package that was obtained freely. The hash table depth was increased from 3 to 6. No problems were encountered embedding it into our system. It was designed to deal with byte streams, so the field is copied into a byte stream: fields with 1-8 (after constant bit removal) are copied to 1 byte/value; values with 2-16 bits are copied to 2 bytes/value, etc.
8. LZRW3A Applied to Differences--The same algorithm is applied to the remapped differences.
9. LZ78--Another LZW dictionary search algorithm, based on reference [5], searches for groups of past strings. Linked lists are again used to speed processing.
10. LZ78 Applied to Differences--The same algorithm is applied to the remapped differences.
11. HA 0.98--This is the freely available file archiver that was rated by [1] as providing the highest compression factors on generic data sets, using an "improved" PPMC (Prediction

by Partial Matching-C) algorithm with 4th order Markov modeling. The modeling is used to form probability weights for arithmetic coding. As with LZRW3A, the data is placed into a byte stream. HA 0.98 actually includes two somewhat different algorithms--"ASC" and "HSC". Both are tested.

The HA 0.98 software was originally designed as a very complex stand-alone program to perform actions in response to command strings, and to interact with the operating system via elaborate system calls. Safely embedding this complex stand-alone program into our application consumed a great deal time and effort, but it does yield excellent compression factors on many fields, in many cases.

12. HA 0.98 Applied to Differences--The same algorithm is applied to the remapped differences.
13. Radix Coding--The minimum value is found and subtracted from all data values. The maximum value (M) is then found, and the sequence of reduced values are interpreted as the base M representation of a single number.

We tried to use mixed radix (M) coding in hybrid with other algorithms (e.g., to code a fractional number of bits in a modified Rice algorithm), but those hybrids were not found to represent significant improvements and were dropped.

14. Radix Coding Applied to Differences--The same algorithm is applied to the remapped differences.
15. Arithmetic Coding--This algorithm is based on reference [7]. It is based on a simple zero-order incrementally adaptive probability model.
16. Arithmetic Coding Applied to Differences--The same algorithm is applied to the remapped differences.
17. Run Length Encoding + Rice--The run length pairs are derived as for algorithm 3. The values are then differenced, remapped and Rice coded. Remapping is modified to take advantage of the fact that values are never repeated. The run lengths are also Rice coded.

Our algorithm is a combination of all of the above algorithms. As discussed earlier, we break the data up into fields, combine them into sequences, and use the algorithm that produces the highest compression factor. In the event that no algorithm leads to an improvement, the field is sent uncompressed.

ADAPTIVE DIFFERENCING AND REMAPPING

Many of the algorithms rely on differencing of values from their predicts to reduce the typical size of the numbers to be coded. A variety of methods is tried.

In the simplest methods, the predict for any value is simply the prior value. The differences of the current value from its predicts are easily remapped into a sequence of non-negative numbers. The optimal way to perform this remapping depends on the proper interpretation of the bit field. All of the following interpretations are tested, to minimize the sum of the remapped values:

- (A) No remapping (or differencing).
- (B) Field is the least significant portion of a larger value; positive differences more likely.
- (C) Same as B, negative differences more likely.
- (D) Field is unsigned.
- (E) Same as D, negative differences more likely.
- (F) Field is signed, two's complement.
- (G) Same as F, negative differences more likely.

The greatest common factor is removed at two points in the process, and minimum values are found and subtracted. If no negative or no positive differences occur and/or there is a minimum absolute difference, those facts are also used. The largest values are found at two points in the algorithm, and are used to determine the number of bits needed to transmit the remapped values.

Finally, a least squares fit is tried in which the current difference is predicted to be $\alpha + \beta * (\text{previous difference})$

The fit is only used when it improves matters. It often does not, because residuals have fewer removable systematic patterns than the differences themselves, and because of overhead. (A two dimensional predictor would be needed to deal with images.)

ANALYSIS OF RESULTS

1. How does our system perform, in terms of speed and complexity?

The tests of multiple compression and differencing algorithms comes at a considerable cost in speed and complexity. In real-world applications, one must limit the algorithm search, based on the test data sets.

2. How does our system compare to generic byte stream compressors, in terms of compression factor?

Table 1 compares the compression factors (C.F.) obtained on 8 packet data streams to those obtained by Unix compress and HA 0.98.

Lossless compression factors vary with the input data, and can not be relied upon completely. Data must be buffered to smooth out compression factor variation, and the system must be able to continue after data loss due to inadequate compression. For example, no compression system can compress random data.

Data set 4 was of an unexpected format. The best our system could do was to apply HA 0.98 to each block of packets. It did slightly worse than Unix compress, probably because Unix

Table 1 Comparison of our compression factors with results from Unix Compress and HA 0.98, as applied to 8 data sets.

Data Set	Description	# of Packets	Our C.F.	Unix Compress	HA 0.98
1	Short runs of different packet types.	5000	3.43	2.04	3.22
2	Mostly of expected data format.	44160	3.42	1.96	3.16
3	Mostly of expected data format.	5000	3.27	1.67	2.49
		125000	3.19	1.68	2.48
4	Unexpected format: header data.	5000	3.15	1.67	2.48
		19200	3.28	1.73	2.62
5	Not of specific expected format, but of expected generic type.	5000	15.23	9.28	15.23
		125000	15.39	15.76	15.39
6	Not of specific expected format, but of expected generic type.	5000	3.01	1.76	2.04
		125000	3.13	1.85	2.13
7	Not of specific expected format, but of expected generic type.	5000	3.13	1.72	2.27
		125000	3.58	2.09	2.16
		375000	3.24	1.93	2.00
8	Expected format.	5000	3.33	1.85	2.75
		125000	3.97	2.07	3.25
		2841	3.54	1.71	2.44

compress was applied to the data set as a whole, rather than to 512 packet blocks, as was done for our system and for HA 0.98 alone.

In every other case our system outperformed both of the generic byte stream compressors.

3. Which compression algorithms perform best, in terms of compression factor?

In our tests, algorithms 6, 7 and 8 were never or almost never picked as best, but the rest were at least occasionally picked. The most commonly picked algorithms are 1, 4, 11, 12, 13, 15 and 16.

As an example, consider a 14 bit field, from data set 8 (the selected algorithm is underlined):

Algorithms:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Compression Factors (n/a means could not be applied, or was worse than uncompressed):																
n/a	1.99	1.68	2.41	2.16	2.16	n/a	n/a	2.27	2.25	<u>2.72</u>	2.70	2.29	2.28	2.65	2.64	2.35
n/a	1.99	1.84	2.24	2.43	2.42	n/a	n/a	2.27	2.26	<u>2.78</u>	2.76	2.17	2.16	2.46	2.45	2.23
n/a	1.74	1.54	2.09	2.30	1.49	1.77	n/a	2.32	1.54	<u>2.88</u>	2.01	1.96	1.94	2.39	1.92	2.06
n/a	1.74	1.27	2.23	2.39	1.62	1.77	n/a	2.30	1.67	<u>2.89</u>	2.18	1.96	1.97	2.36	2.10	2.17
n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a

The first 5 intervals had 512 input packets, the last had 281. In the last two intervals, the field was too noisy for any compression.

The variation in the algorithm 2 compression factor indicates that the number of constant bits varied from interval to interval. Some methods involving first differencing performed well, indicating it is partially valid to model the values as a smooth curve. However, some dictionary search methods also performed well, indicating that the data has a tendency to repeat itself.

Consider also a 4 bit field, from the same data set, on which dictionary search algorithms and HA 0.98 both perform poorly:

Algorithm:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Compression Factors (n/a means could not be applied, or was worse than original):																
n/a	1.99	1.08	2.32	1.70	1.69	n/a	n/a	1.80	1.79	n/a	n/a	<u>2.49</u>	2.46	2.42	2.39	2.14
n/a	1.99	1.10	2.40	1.66	1.65	n/a	n/a	1.84	1.78	n/a	n/a	<u>2.49</u>	2.46	2.43	2.40	2.15
n/a	1.99	1.11	<u>2.49</u>	1.78	1.75	n/a	n/a	1.81	1.77	n/a	n/a	2.49	2.46	2.47	2.44	2.15
n/a	1.99	n/a	2.48	1.75	1.75	n/a	n/a	1.85	1.80	n/a	n/a	<u>2.49</u>	2.46	2.47	2.44	2.20
n/a	1.99	1.16	2.41	1.73	1.72	n/a	n/a	1.78	1.83	n/a	n/a	<u>2.49</u>	2.46	2.43	2.40	2.23
n/a	1.98	1.06	2.32	1.54	1.52	n/a	n/a	1.76	1.71	1.68	1.65	<u>2.46</u>	2.42	2.34	2.30	2.03

Algorithm 13 (Radix coding) generally performs best, indicating that the contents are essentially random, but are restricted to some range.

Finally, consider a 16 bit field, from the same data set:

Algorithm:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Compression Factors (n/a means could not be applied, or was worse than original):																
n/a	1.59	2.46	9.15	2.32	6.39	1.07	3.88	1.45	7.46	2.91	8.12	1.92	5.09	1.66	<u>9.48</u>	8.87
n/a	1.33	1.75	<u>7.09</u>	1.68	4.80	n/a	n/a	1.10	5.35	2.07	5.76	1.75	4.42	n/a	<u>6.99</u>	7.05
n/a	1.59	1.94	7.34	1.88	5.25	n/a	n/a	1.19	5.36	2.03	6.19	1.76	5.41	1.61	<u>7.49</u>	7.15
n/a	1.45	1.94	7.45	1.83	5.04	n/a	n/a	1.19	5.56	2.16	5.78	1.77	5.08	n/a	<u>7.49</u>	7.24
n/a	1.59	2.23	<u>7.83</u>	2.05	5.15	n/a	n/a	1.29	5.70	2.33	5.95	1.79	4.82	1.63	<u>7.45</u>	7.59
n/a	1.76	2.15	7.59	2.04	5.63	n/a	n/a	1.35	5.88	2.19	5.94	1.99	5.22	1.78	<u>7.74</u>	7.24

This is a clear case where testing multiple algorithms can substantially improve performance. The best algorithms were 4 and 16. Since both involve coding the remapped first differences, it is probable that this field follows a fairly smooth curve. Intervals in which method 4 (Rice) performed better presumably have regions of different activity levels, and so benefitted from its small scale adaptivity.

CONCLUSION

This methodology can produce markedly higher compression factors on packet data streams than simpler generic compression techniques, but at a substantial cost in complexity and speed.

REFERENCES

- [1] Gailly, Jean-loup, "Comp.Compression Frequently Asked Questions", pasted to Usenet news group comp.compression, 27 July, 1995.
- [2] Rice, R.F., "Some Practical Universal Noiseless Coding Techniques", JPL Publication #79-22, NASA, JPL/CIT Pasadena, CA, March 1979.
- [3] Rice, "Some Practical Noiseless Coding Techniques, Part III, Module PSI14,K+", JPL Publication #91-3, NASA, JPL/CIT, Pasadena, CA, November 1991.
- [4] Williams, Adaptive Data Compression, Kluwer Academic Publishers, Boston, MA, 1991, pp 35.
- [5] Williams, *ibid*, pp 37.
- [6] Williams, "An extremely fast ZIV-Lempel Data Compression Algorithm", Proc. of Data Compression Conference, IEEE TH0373-1/91, pp 362-371. Actually describes algorithm LZRW1. The LZRW3A algorithm was described in a computer archive (<ftp://ftp.adelaide.edu.au/pub/compression>).
- [7] Bell, T.C., Cleary, J.G. and Witten, I.H., Text compression, Prentice Hall, Englewood Cliffs, NJ, 1990.

Reduction and Coding of Synthetic Aperture Radar Data with Fourier Transforms

David G. Tilley
6612 Hunter Road
Elkridge, MD 21227
dgtilley@aol.com

55-61

5305

1-10

Abstract

Recently, aboard the Space Radar Laboratory (SRL), the two roles of Fourier transforms for ocean image synthesis and surface wave analysis have been implemented with a dedicated radar processor to significantly reduce SAR ocean data before transmission to the ground. The object was to archive the SAR image spectrum, rather than the SAR image itself, to reduce data volume and capture the essential descriptors of the surface wave field. SAR signal data are usually sampled and coded in the time domain for transmission to the ground where Fourier transforms are applied both to individual radar pulses and to long sequences of radar pulses to form two-dimensional images. High resolution images of the ocean often contain no striking features and subtle image modulations by wind generated surface waves are only apparent when large ocean regions are studied, with Fourier transforms, to reveal periodic patterns created by wind stress over the surface wave field. Major ocean currents and atmospheric instability in coastal environments are apparent as large scale modulations of SAR imagery. This paper explores the possibility of computing complex Fourier spectrum codes representing SAR images, transmitting the coded spectra to earth for data archives and creating scenes of surface wave signatures and air-sea interactions via inverse Fourier transformation with ground station processors.

Introduction

Synthetic aperture radar (SAR) images of the earth and oceans are computed using Fourier transform methods applied to temporal records of microwave radar signals received along an orbit in space. The Applied Physics Laboratory of Johns Hopkins University (JHU/APL) developed its ocean SAR data processor for the Space Radar Laboratory's C-band radar system to produce Fourier power spectra of SAR images. The full swath width was not required and two imageries were computed and their average spectrum was transmitted every 20 kilometers along the shuttle track to significantly reduce the data volume. The Fourier power spectra were computed from 7.68 x 7.68 kilometer square SAR imageries, 256 x 256 picture elements (pixels) each, which were pulse compressed and Doppler synthesized by the on-board processor. During image synthesis, radar pulses were pre-summed on input and image pixels were post-summed on output to obtain 30 meter resolution. During SAR analysis, each SAR image spectrum was defined as 8-bit Fourier power amplitudes over an array of 64 x 64 wave numbers, after block averaging with factors 4 x 4. The 64 coordinate values along track were equally spaced spanning wave numbers from -0.105 to 0.105 (i.e., $\pi/30$) radian/meter. Only the 32 positive wave numbers in the cross-track coordinate were transmitted to the ground because the SAR is subject to a 180° directional ambiguity and the negative half of the spectrum contains redundant information. Block averaging from 256 to 64 samples degrades spectral resolution from 0.0004 to 0.0016 radian/meter. Therefore, archived SAR power spectra represent image features as small as 30 meters but only as large as 1.92 kilometers. If two complex components of the Fourier power spectrum could be archived in the future, then SAR images could be reconstructed on the ground. Furthermore, the full 7.68 kilometer extent of the SAR imageries could be retained if the complex spectra were not block averaged, but reduced by discarding the Fourier components with the lowest power.

The two flights of the Space Radar Laboratory (SRL) produced over 100,000 SAR ocean power spectra which have been further reduced on the ground to surface wave number vectors along the tracks of the space shuttle through the oceans of the southern hemisphere. The SRL

surface wave data are still being assessed and it is too soon to draw conclusions or report the results of surface wave imaging experiments in the southern oceans. Comparisons of the C-band SAR sensors aboard the ERS-1 satellite and Canadian CV-580 aircraft (Tilley and Beal, 1994) indicate that ocean wave resolution for waves travelling along the radar track conforms to velocity bunching theories that have been based on L-band SIR-B and Seasat SAR observations. However, coherent interactions between tilt modulation and position bunching of hydrodynamic facets over the long wave profile are possibly more restrictive for C-band imaging of waves travelling across the radar track. This does not impact the resolution of ocean wavelength but is rather a problem in measuring the direction of propagation and angular spread of the wave field. Furthermore, this problem is most severe in low sea states when the onset of wind roughening produces the most hydrodynamic contrast over the wave cycle. Hence, a high sea state SRL scene imaged at C-band is expected to be similar to the L-band SIR-B scene of ocean waves spawned by hurricane Josephine (Tilley and Sarma, 1993).

The ocean wave spectrum depicted in Fig. 1 was characteristic of the SIR-B hurricane Josephine data set. This was truly an extreme sea state with winds reported approaching 30 meters per second at the time of the overpass of the space shuttle Challenger approximately 90 kilometers northeast of the storm center. These SAR ocean image data, acquired in 1984 two years before the Challenger accident, surprised many remote sensing scientists who expected that wave imaging would be severely compromised in high sea states. It seems that just the opposite was true; the SAR performed most reliably in situations when we needed the most accurate information on storms that threatened our coastlines with wind damage, high water and beach erosion.

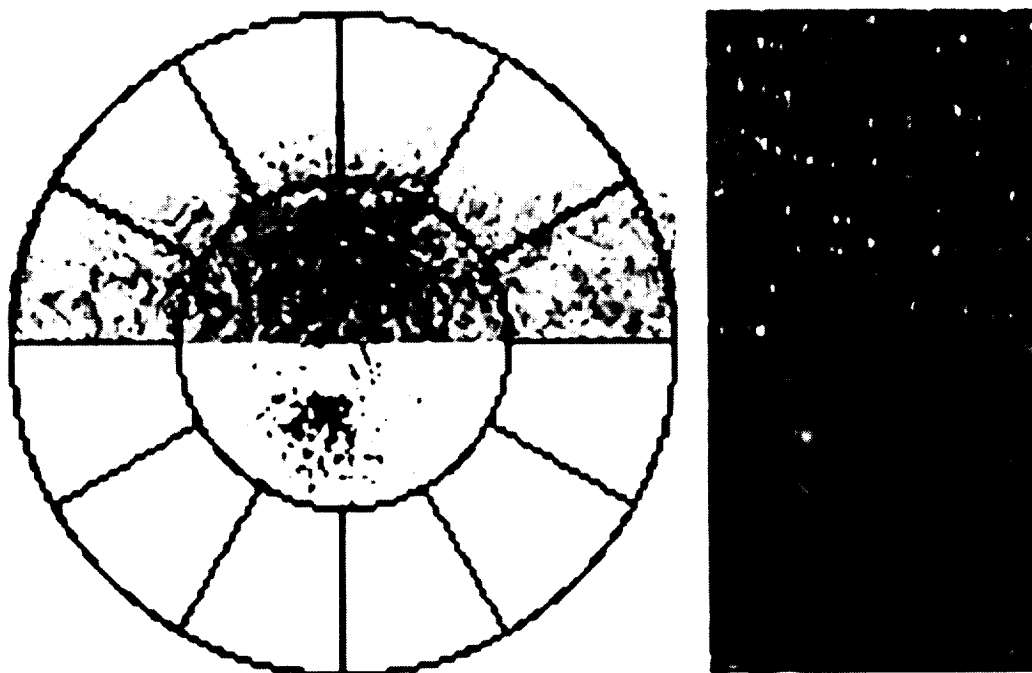


Figure 1. The top half of the Fourier wave number spectrum on the left depicts the power signature of both the wave signal and speckle noise in the SAR image, top right. The inner and outer circles represent wave number magnitudes of 0.63 and .125 rad/m. (i.e. 100 and 50 meter wavelengths, respectively). The angular coordinate represents the direction of wave propagation subject to a 180° ambiguity. For example, the unpartitioned spectrum at the top indicates linear SAR features moving toward the northeast. The partitioned signal spectrum at the bottom indicates waves travelling toward the southwest. It is not possible to make this distinction in the SAR image or its spectrum compressed and reconstructed wave scene, lower right. This stationery aspect of the image is the source of the ambiguity.

In this case, the spectrum indicated waves travelling toward the NNE (parallel to and several hundred kilometers off the North Carolina shore) with a wavelength of about 200 meters. The spectrum was computed from the SAR image at the top of the panel on the right side of Figure 1. Note that waves in the SAR image appeared as linear rows of points (i.e., spikes in radar cross section) thought to be produced by the sporadic generation of radar normal facets on the windward side of surface swell waves. This discrete nature of the SAR image resulted in a noise-like cloud of low power wave numbers distributed broadly in the Fourier spectrum. The linear patterns in the SAR image resulted in a cluster of high power wave numbers representing the surface wave signal in the Fourier spectrum. It was found that a power threshold one standard deviation above the mean wave number power can be used to effectively partition the spectrum into noise and signal components. Furthermore, the signal typically consisted of about 1% of the Fourier wave number components in the full spectrum. Inverse Fourier transformation of the signal spectrum partition resulted in the low resolution image in the lower portion of the right hand panel in Figure 1. This lower resolution image was adequate for ocean wave imaging suggesting that Fourier domain coding of SAR imagery could be employed to reduce data volume by a factor of 99%. Indeed, the reconstructed image more visually represented the ocean wave height topography. (Tilley and Sarma, 1993). Independent ship sightings reported that these waves were about 8 meters high with a wavelength of 200 meters from crest to crest.

A SAR Data Compression Method for Coastal Oceanography

Coastal oceanography often involves monitoring geostrophic currents and air-sea interaction processes that are manifested over large spatial regions. Ocean swell does not necessarily play a major role in basin scale ocean processes. However, the behavior of surface gravity waves under the influence of strong currents in the ocean and atmosphere often reveals much about the coastal environment. For example, surface waves crossing the Gulf Stream were observed by the SAR system aboard the ERS-1 satellite and spectral measurements of wave refraction and dispersion have been used to estimate the current velocity profile (Tilley and Beal, 1992). Hence it is important to resolve swell wavelengths of 100 meters typically and to observe oceanographic features over littoral regions of 100 kilometers and more. Hence, megabytes of image pixels are required to archive single scenes and as satellite observations are repeated with each orbit, many gigabytes of SAR data will be required at a single site to monitor global change over several years. It is clear that data compression methods are required that satisfy the specific scientific requirements of oceanographic remote sensors and other individual earth observing systems that will be used to address global change research in the new millennium.

The German X-band SAR system designed for the Space Radar Laboratory (SRL) returned image data that are well suited for testing SAR data compression techniques. The Fourier filtering method applied to the SIR-B data was designed to produce 100 meter image pixels, an 8x reduction from the original 12.5 meter grid. This earlier SAR system design was more than adequate for ocean imaging and the Fourier filtering technique severely compromised fine scale features. However, the X-band SAR processor produced images with 50 meter image pixels, from a radar system with nominal resolution of 75 meters, so that Fourier filtering to a 100 meter grid is quite appropriate.

SAR signatures of littoral land masses may be of interest. Hence, the Fourier coding of such scenes should be investigated to determine what savings in data volume can be achieved while preserving adequate scene detail. The East Coast of Australia is depicted in the X-Band SLR scene in Fig. 2a. This scene was used to define a Fourier coding algorithm. A two dimensional Fourier transform of the image was computed with MATLAB software yielding a complex spectrum. Only wave numbers less than $\pi/100 = 0.063$ radian/meter in both range and azimuth (i.e., the inner quarter of the square wave number domain) were used during an inverse Fourier transformation to produce a filtered image with 75% fewer pixels, each representing a 100 meter square. Further

reduction of the spectral data can also be simulated by setting complex values to zero when their power intensity (modulo 2 amplitude) is less than the mean spectrum power as shown in Fig. 2b.

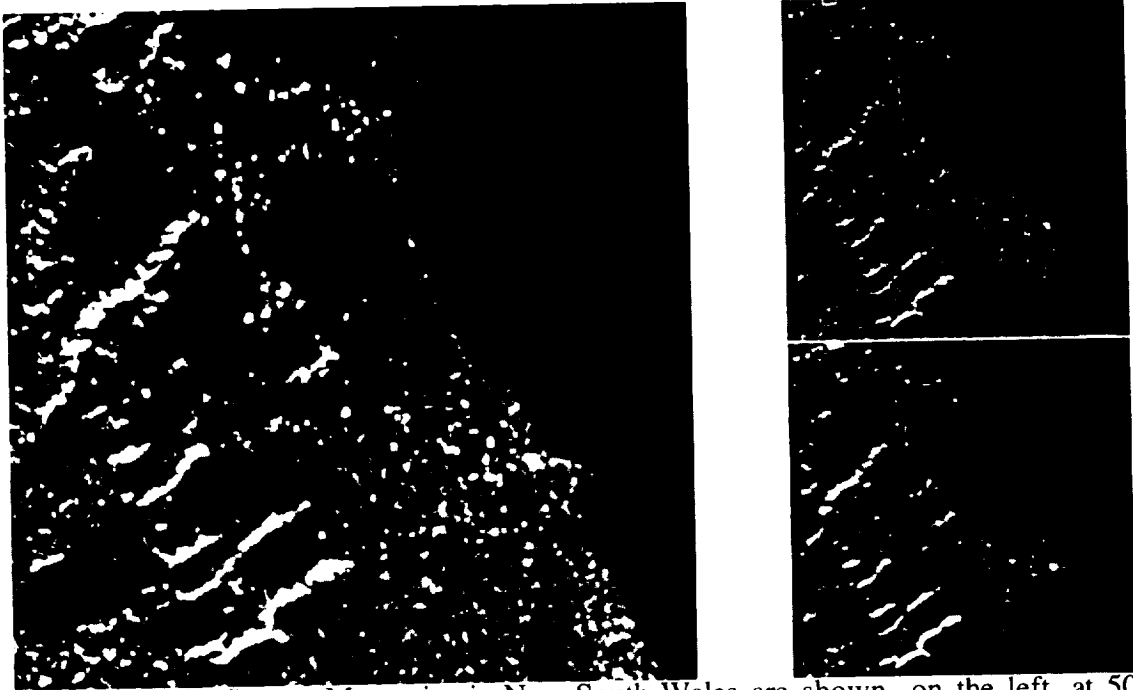


Figure 2a. The Snowy Mountains in New South Wales are shown, on the left, at 50 meter resolution in this 256 x 256 pixel segment of a scene created by a German X-band processor. Data compression via spectrum filtering, between forward and inverse Fourier transforms, yields scenes with 100 meter resolution in arrays of 128 x 128 pixels, right.

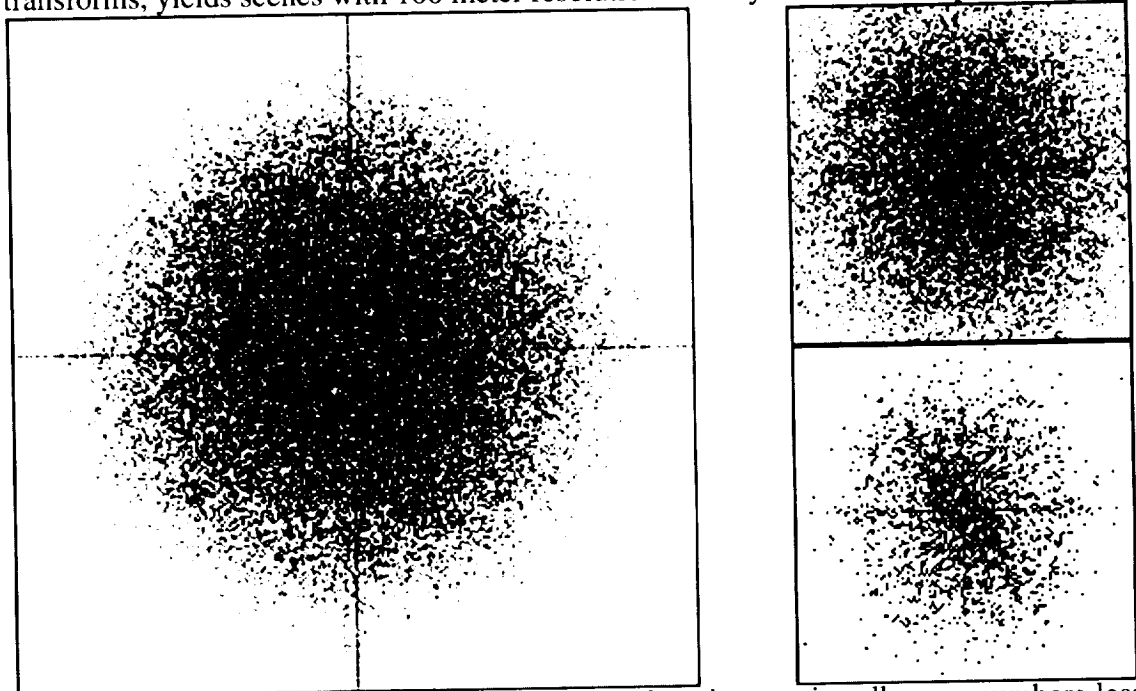


Figure 2b. The initial Fourier power spectrum domain contains all wave numbers less than $0.125 (\pi/50)$ radian/meter, left. The filtered spectral databases include all wave numbers less than $0.063 (\pi/100)$ radian/meter, top right, and only those power spectral intensities above the mean, bottom right. Note the low frequency alias along each axis.

The East Australian Current is quite apparent in a Fourier filtered scene consisting 384 x 512 pixels at 100 meter resolution, as shown in Fig. 3. This scene is the result of 25% spectral data reduction applied to an X-band SRL scene segment originally consisting of 768 x 1024 pixels, which was divided into 12 subscenes, each 256 x 256 pixels. Each subscene (i.e. A1, B1, C1, A2, B2, etc.)

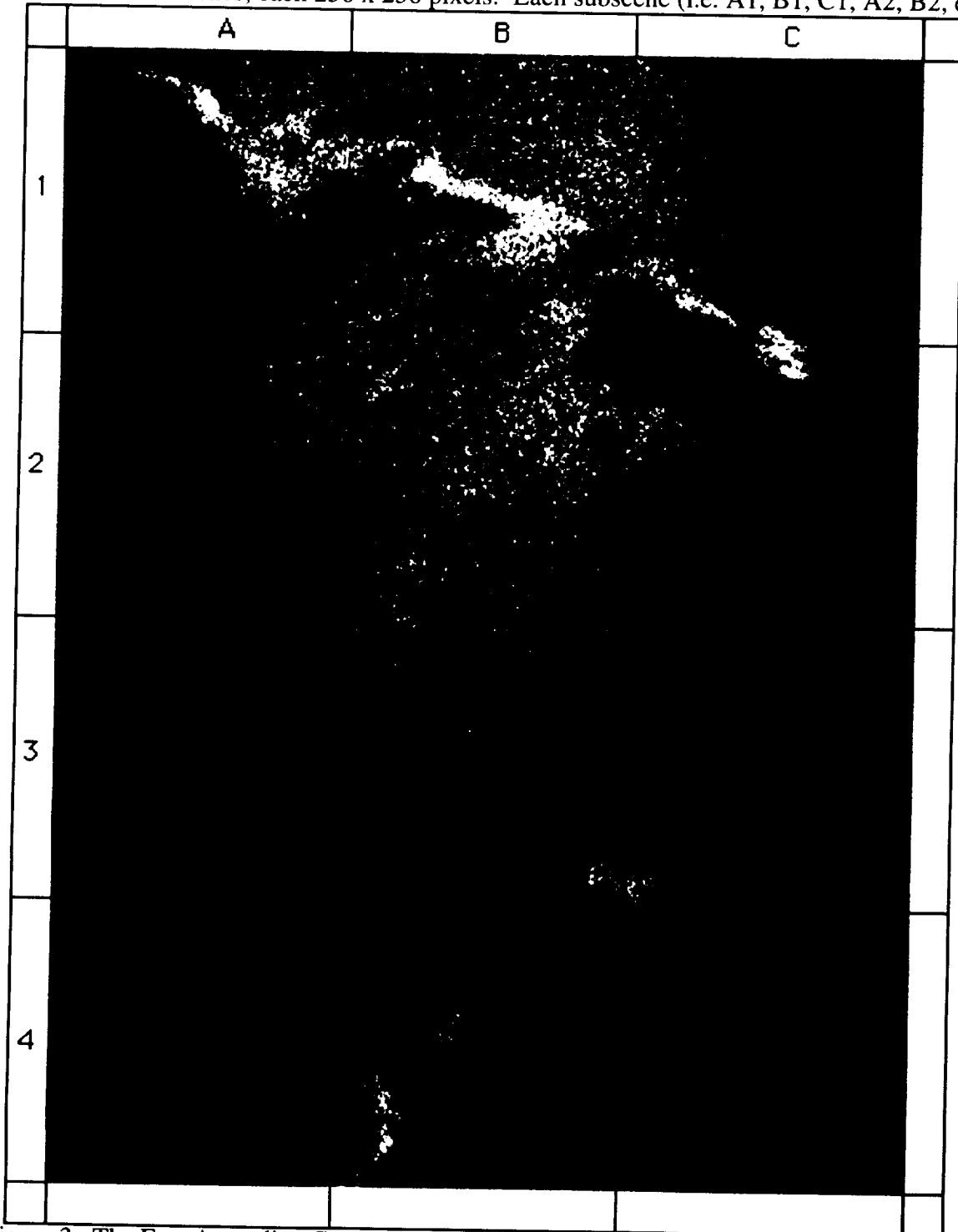


Figure 3. The East Australian Current is shown in the X-band SAR image acquired by the NASA Space Radar Laboratory on April 11, 1994. Image reduction via Fourier filtering preserves the image mean with a slight reduction in signal variance. The low frequency alias introduced by edge effects in the Fourier transform is barely visible along boundaries.

was processed separately without applying the mean spectral threshold. The input image mean was computed and after Fourier filtering the output image was scaled to have the same mean value. Hence, subscene boundaries were nearly indiscernible when the 12 subscenes were reassembled. The edge effects that are visible are introduced by the forward Fourier transform. Even though the mean is removed from the image prior to the transform, its variance is discontinuous at the four edges. The high frequency components associated with these edges are folded in upon the spectrum axes as a low frequency alias of those Fourier amplitudes. Much of this alias is removed when Fourier components are trimmed from the spectral domain at wave numbers exceeding $\pi/100$ radians/meter in either the range or cross range (i.e., azimuth) coordinate. However, after the inverse Fourier transform, a small alien variance of the zero mean image is apparent along its boundary.

Although, a portion of the variance observed in SAR imagery can be attributed to speckle noise and other coherent aspects of the remote sensing technology, a good portion is signal modulation descriptive of ocean features. Therefore, data compression via Fourier filtering should preserve image variance to the greatest possible degree. The significant parameter often used to evaluate speckle noise content in SAR imagery is the variance-to-squared mean ratio. Although, it is beyond the scope of this paper to determine signal-to-noise levels for the German X-band sensor, the variance-to-squared mean ratios have been computed for the twelve subscenes that encompass the East Australian Current in Figure 3. Table 1 lists these ratios for the SAR subscenes both before and after data compression from 50 to 100 meter pixels. This study represents the most elementary application of the Fourier filtering technique over a typical SAR scene and the values listed in Table 1 represent a normal control on more severe lossy compressions that further reduce the spectral database.

Table 1. Mean Normalized Variance of SAR Scenes Before (X) and After (Y) Compression

Scene #	A1	A2	A3	A4	B1	B2	B3	B4	C1	C2	C3	C4
Mean	91.4	82.3	57.1	39.0	122.	115.	85.7	77.4	86.6	59.2	59.8	64.4
Var.X	.157	.082	.092	.204	.061	.046	.079	.112	.124	.282	.128	.102
Var.Y	.149	.075	.084	.194	.054	.039	.072	.104	.117	.274	.120	.095

The edge effects discussed above could be avoided by applying a much larger Fourier transform to the entire X-band scene. This could be accomplished in future SAR system designs during wave domain compression (Tilley and Yemc, 1994) of complex signal frequencies in the process of forming images from raw radar data. Such image processing procedures for aperture synthesis include window functions (e.g., the Hanning window) in fast Fourier transform (FFT) computations that taper the complex signal data to prevent the introduction of edge effects. This would prevent the introduction of the low frequency alias into the SAR spectrum and subsequent partitioning of the spectrum into signal and noise components could proceed without emphasizing edge effects. A simple cosine taper has been applied to the X-band SAR data appearing in Fig. 4a representing a surface wave field approximately 100 kilometers off the East Australian Coast on April 11, 1994. The effect of that cosine tape is apparent in Fig. 4b which implements the Fourier filtering method to discard 41% of the spectral components using a mean spectral threshold to identify the noise components. This operation was applied to all the components of the spectrum resulting in an image with 256 x 256 pixels with 50 meter spacing after inverse Fourier transformation. It is possible to discard 75% of spectral components without applying the mean spectral threshold by trimming of the edges of the spectrum to yields a 128 x 128 database. When an inverse Fourier transform is applied to the reduced Fourier spectrum, the image with 100 meter pixels, shown in Fig. 4c, is also 75% smaller. The surface wave field depicted in these images has

a wavelength of 200 meters approximately, with crests separated from troughs by 100 meters. Hence, these bright and dark features appear at the resolution limit of the 128 x 128 pixel database. A good compromise is to trim the spectral database to 192 x 192 spectral components and to apply the mean spectral threshold, discarding a total of 75% of the Fourier amplitudes. Inverse Fourier transformation results in an image, shown in Fig. 4d, with 192 x 192 pixels spaced at 66.7 meters.

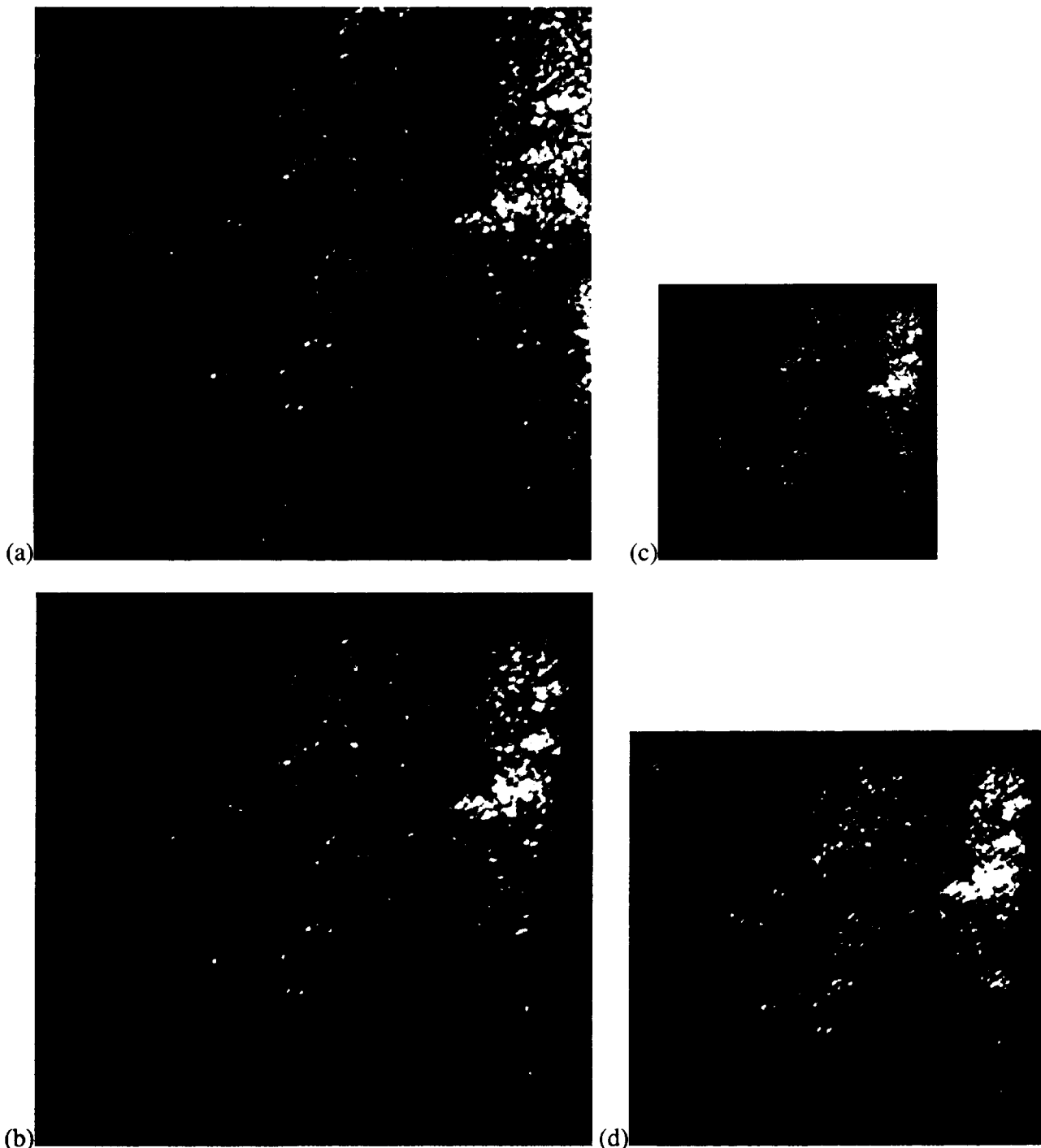


Figure 4. A wave field (4a) was observed in X-band SAR imagery and its tapered database (4b). Fourier component reduction by 75% can be achieved by spectrum trimming and partitioning resulting in images with 100 meter pixels (4c) and 67 meter pixels (4d), respectively.

The full Fourier spectrum of the ocean wave field, as shown in Fig. 5, is descriptive of three air-sea-radar interactions. A circular pattern in the broadband distribution of spectral components, falling in amplitude with distance from the center, departs from the flat 'white noise' response expected for coherent speckle noise. This speckle noise spectrum is descriptive of the finite resolution response of the radar to a uniform distribution of point scattering facets raised by the wind over the sea surface. The concentration of high amplitude components at the center of the spectrum is descriptive of large scale hydrodynamic variations in wind friction over the scene shown in Fig. 4. Note particularly the wind sea spikes in the upper right that appear correlated with wave field crests. The wave field itself is more uniform throughout the scene and velocity bunching and tilt modulations (Tilley and Beal, 1994) of the surface scattering facets appears as clusters of Fourier power components near the inner circle in the spectrum shown in Fig. 5a. The wind friction and wave field signals are isolated by using a mean spectral threshold on the 128 x 128 component database shown in Fig. 5b and the discarded broadband speckle noise cloud is shown in the 192 x 192 component database in the background.

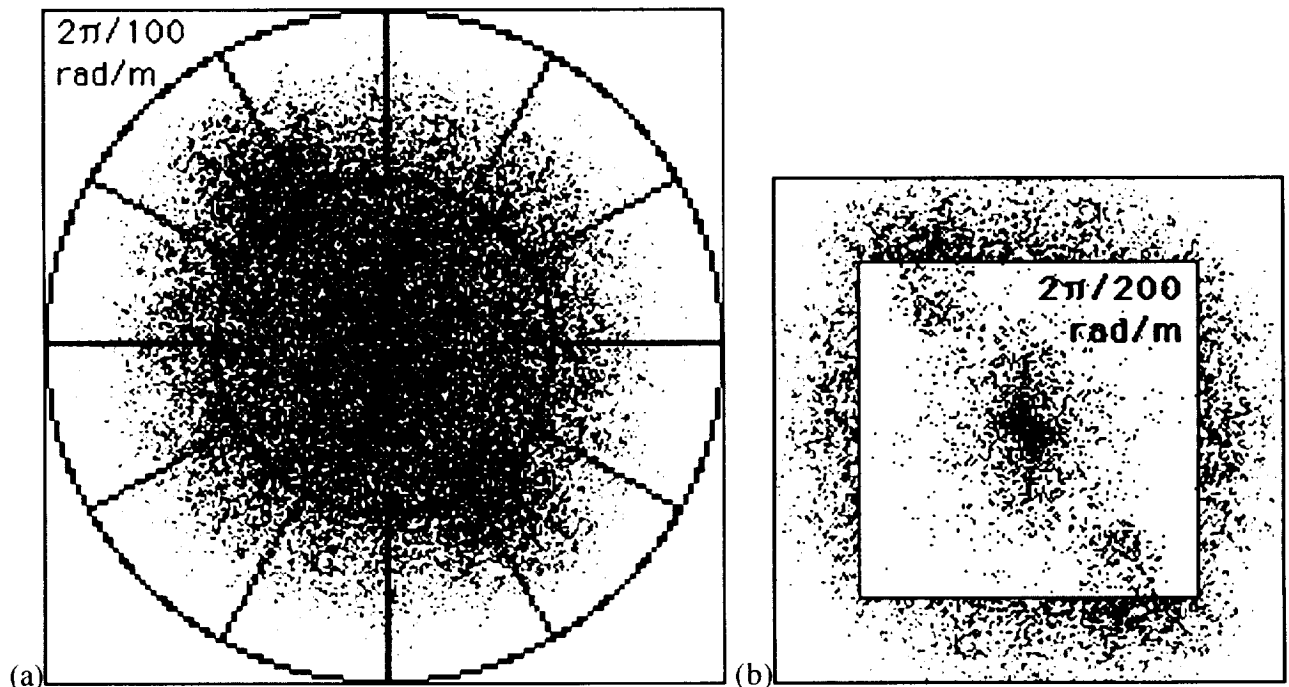


Figure 5. The square boundaries represent radar range and azimuth spatial frequency limits at $2\pi/100$ (0.125) radians/meter which is also the ocean wave number depicted by the radius of the outer circle (a). The inner circle at $2\pi/200$ (0.063) radians/meter coincides with the clusters of spectral components associated with the wave signal bounded by the spatial frequency limits of the inner square (b) of the spectrum that has been partitioned with the mean threshold. Note that no low frequency alias exists along the range and azimuth axes due to the FFT cosine taper.

It is possible to reduce speckle noise while retaining the wind and wave signals, with no low frequency alias to interfere with the calculation of the mean spectral threshold. As the boundary of the spectral partition moves toward lower wave numbers, low amplitude Fourier components are removed from the spectrum and the mean threshold occurs at relatively higher amplitudes. Hence, more of the broadband speckle noise cloud is eliminated. The percentage of the original 256 x 256 Fourier component amplitudes that are discarded in reconstructing images from the complex spectra varies from 0 to 97 to demonstrate the Fourier filtering technique that is proposed for the archival of the SAR data. The reconstructed image mean, its variance-to-squared mean ratio and the percentage of spectral components discarded are listed in Table 2. Note that image variance decreases and the image mean increases with progressive spectral reduction.

Table 2. Mean Normalized Variance for Fourier Filtering of the Surface Wave Image

Figure #	4a	4b	4c	4d	6a	6b
Image Mean	60.63	60.55	61.20	67.99	74.77	85.45
Var-to-SqMean	.2234	.1834	.1842	.1548	.1493	.1231
Reduction %	0	41	75	75	97	97

In practice, the spectral data compression factors than can be achieved will be somewhat less than the reduction percentages listed above. For example, the elimination of low power spectral components could be run length encoded to specify the positions of zero power components that must be used in the image reconstruction. This would require the injection of spectral masking characters representing as much as 99% of the wave number domain, as in the case of hurricane Josephine discussed in Fig. 1. Alternatively, the location in the wave number domain of the Fourier amplitudes that are retained could be specified as 8-bit wave numbers in both range and azimuth. Furthermore, retaining the complex spectrum also requires two 8-bit amplitudes rather than one image pixel or one spectral power amplitude. Hence, the 97% reduction ratios listed above may only amount to 88% (i.e., $100 - 4 \times 3 = 88$) and the 75% reduction may actually not help at all (i.e., $100 - 4 \times 25 = 0$). Clearly, more work is needed to develop an efficient coding algorithm for specifying the wave numbers associated with the complex Fourier amplitudes.

The surface wave field images depicted above were all calculated in MATLAB using full precision floating arithmetic. It may be interesting to simulate archival of the spectral data as two 8-bit complex Fourier amplitudes and observe the difference in reconstructed image statistics and appearance. Image reconstruction by Fourier inversion of the spectrum shown in Fig. 5b. produces the visual comparison shown in Fig. 6. The image reconstructed using full MATLAB precision is shown in Fig. 6a and. Fig 6b depicts image reconstruction when complex spectral amplitudes are scaled from 0 to 255 prior to Fourier inversion. Amplitude statistics for the square pixels, each 100 meters x 100 meters, appearing in these two SAR images are listed in Table 2.

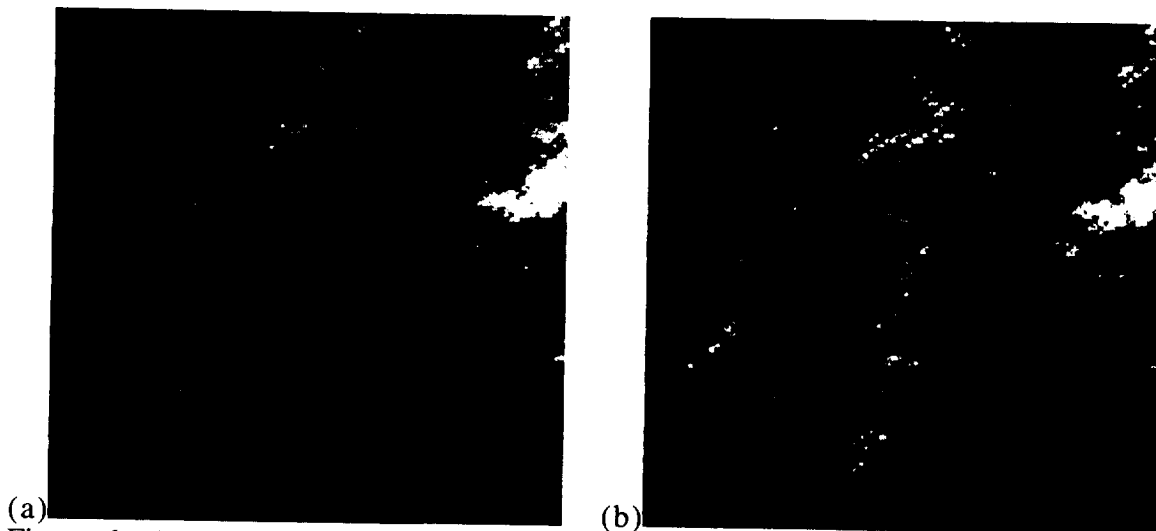


Figure 6. These two SAR images represent a surface wave field within a square 10 kilometer scene using 100 meters pixels. Speckle noise has been suppressed by Fourier filtering out 97% of the complex spectrum amplitudes. The remaining signal components were scaled as 64-bit (a) and 8-bit (b) numbers prior to Fourier inversion of the spectrum.

Recommendation

A small satellite SAR system is now under consideration by NASA with improvements in data processing and reduction. A 50 kilometer SAR swath imaged with 50 meter pixels would enable observation of coastal ocean processes, as demonstrated with the X-band SRL system. Absolute cross section statistics should be computed in the time domain of the raw radar signal and forward Fourier transforms can be applied on the long and short time scales associated with the SAR along track and downrange coordinates. SAR compression algorithms should be applied in the two-dimensional wave number domain (Tilley and Yemc, 1994) to create complex spectral databases that can be transmitted to the ground. These complex spectral databases could be subjected to inverse Fourier transformation with ground station processors to produce full resolution imagery. However, they could also be filtered in the wave number domain to further compress the spectral database. Algorithms for partitioning ocean wave spectra are now being studied and may be developed to reduce the SAR spectrum to a handful of surface wave system vectors. Localized sea spikes raised by the wind could be described by a two parameter Poisson model (Tilley and Sarma, 1993) associated with the discarded broadband SAR spectrum. These two wave numbers, describing the sea and swell, and the archived cross section data might be sufficient for estimates of wind friction velocity. Only the most significant wave numbers need be archived for well defined oceanographic applications.

Conclusion

Ground processing of raw SAR signal data has historically been accomplished with Fourier transforms. First, SAR image synthesis with the range-Doppler algorithm involves forward and inverse Fourier transform pairs applied to both the downrange and along track coordinates. Second, surface wave analysis involves two-dimensional Fourier transformation of SAR imagery to create backscatter power spectra that approximate the ocean wave spectra used by oceanographers. Therefore, the Fourier transform is the appropriate mathematical code for representing ocean features in SAR imagery. This study has shown that SAR image data can be well represented by complex Fourier spectra that reduce data volume by at least a factor of 2. Furthermore, the raw signal data which are usually transmitted and archived exceed the SAR image data volume by a factor of 8. Hence, a data reduction factor of 16 may be anticipated for a SAR system designed for dedicated oceanographic applications. This study also indicates that coastal land features may also be imaged with 100 meter resolution using complex spectral archives. Therefore, it is recommended that the next generation SAR system include a Fourier wave domain processor to produce complex spectral data representing littoral scenes.

References

- Tilley, D. G., and Yemc, D. J., "Wave domain processing of synthetic aperture radar signals", Johns Hopkins APL Technical Digest 15(3), Sept. 1994.
- Tilley D. G. and Beal, R. C., "ERS-1 and Almaz estimates of directional ocean wave spectra conditioned by simultaneous aircraft SAR and buoy measurements", Atmosphere-Ocean 32(1), 113-142, 1994.
- Tilley, D. G. and Y.V. Sarma, "Comparison of synthetic aperture radars applied for satellite remote sensing of the ocean surface" Trends in Geophys. Res., Chap. 2, Research Trends, Council of Scientific Research Integration, Kerala, India, 1993.
- Tilley D. G. and Beal, R. C., "ERS-1 and Almaz SAR Ocean Wave Imaging over the Gulf Stream and Grand Banks", Proceedings First ERS-1 Symposium: Space at the Service of our Environment, ESA SP-359, 729-734, Cannes, France, 1992.

Selecting a General-Purpose Data Compression Algorithm

Gary Jason Mathews

NASA/Goddard Space Flight Center
Greenbelt, Maryland 20771
(301) 286-6879

mathews@nssdc.gsfc.nasa.gov

56-61

5306

p-10

Abstract

The National Space Science Data Center's Common Data Format (CDF) is capable of storing many types of data such as scalar data items, vectors, and multidimensional arrays of bytes, integers, or floating point values. However, regardless of the dimensionality and data type, the data break down into a sequence of bytes that can be fed into a data compression function to reduce the amount of data without losing data integrity and thus remaining fully reconstructible. Because of the diversity of data types and high performance speed requirements, a general-purpose, fast, simple data compression algorithm is required to incorporate data compression into CDF. The questions to ask are how to evaluate and compare compression algorithms, and what compression algorithm meets all requirements. The object of this paper is to address these questions and determine the most appropriate compression algorithm to use within the CDF data management package that would be applicable to other software packages with similar data compression needs.

Keywords: Data compression; Algorithm comparison; CDF

Introduction

Because of the large amounts of data collected by the National Aeronautics and Space Administration (NASA) and the scientific communities, data compression is an important consideration for archival and data management systems. The National Space Science Data Center (NSSDC) is evaluating various compression techniques for incorporation into the Common Data Format (CDF)[1], which is a data management package for storing, manipulating, and accessing multidimensional data sets.

Obviously, the best data compression algorithm would compress all data with the highest compression rate in the least amount of time, but no such algorithm exists. There is no best compression algorithm for everyone since the criteria for measurement are both data- and application-dependent. One user might want a high data compression ratio regardless of time, another fast compression with some sacrifice to the compression ratio, and others something in-between. There are trade-offs among time to compress and decompress the data, compression rates, memory requirements (hash tables, frequency counts, temporary buffers, etc.), and algorithmic complexity, so application developers must let their requirements determine what trade-offs are more or less important.

CDF provides a programming interface for applications whose goal is to provide fast access to the data; therefore, speed is a high priority for the desired compression algorithm. There is not much need for compression if most data cannot be compressed, so users need a general-purpose

compression algorithm that compresses most data with a good compression ratio. The compression ratio is defined as the percentage of reduction achieved or as $(1 - \text{compressed data size} / \text{uncompressed data size}) \times 100$. A maximum of 100% compression is not possible unless the compressed size is zero, a situation only possible if the uncompressed data size is zero as well. Therefore, the upper limit is 100%, and the ratios will approach this limit. A negative compression ratio will result if the compressed data size is larger than the uncompressed data size, that is, the data have expanded with negative compression. There is no lower limit, since data can expand infinitely by recursively adding a single byte to the previous data ad infinitum. For all intensive purposes the lower limit can be set to zero, since if we fail to compress something, then we can leave it uncompressed with a compression ratio of zero. A good compression ratio is defined as a significant data reduction for all test data as a whole for a particular compression algorithm. Furthermore, CDF provides a library that is linked into all CDF applications, so also needed is a compression algorithm that does not dramatically increase the application size. Portability is another important aspect since both the CDF software and data sets are portable to DEC Alpha, DECstation, HP 9000, IBM PC, IBM RS-6000, Macintosh, NeXT, SGI, Sun, and VAX platforms. The hallmark of the CDF concept is its data set independence, so a compression implementation with minimum architecture- and machine-level dependencies is preferable to simplify porting to other platforms. In the context of the requirements classification, portability will be an implied requirement for a simple algorithm. Therefore, the following three factors will be used to compare the different compression algorithms and decide what algorithm is best for the users' needs:

- A fast algorithm that minimizes compression/decompression time.
- A good algorithm with a high compression rate.
- A simple algorithm with a small code size.

Data Characteristics

CDF can handle data organized conceptually as scalar and multidimensional with arrays of up to ten dimensions. The supported data types are those consistent with types available with C and FORTRAN compilers on most systems, which include 1-, 2-, and 4-byte signed and unsigned integers, 4-byte single-precision floating-point, 8-byte double-precision floating point, and 1-byte signed and unsigned character types. For example, there may be data defined as a single scalar 1-byte integer and another as a $512 \times 512 \times 16$ array of double-precision floating point numbers. Although it is not very promising to compress a single scalar byte, compressing a large array or stream of bytes has vastly different results. Similarly, any data value with 2 or 3 bytes cannot be reduced because no reduction is possible. However, a 4-byte quantity can be reduced to possibly 3 bytes, so the minimum file size of 4 bytes is selected. Since most data stored in CDFs are typically under three dimensions, such as scalar data types, three-element vectors (e.g., x, y, z components), and two-dimensional images (e.g., 320×240), the size of the sample data files is arbitrarily chosen to range from 4 bytes to 144 Kbytes. Although larger files achieve higher compression ratios, the smaller data files better represent the typical size of data stored as components within a CDF.

To evaluate a general purpose compression algorithm, a representative collection of sample data to test the algorithms must be defined. As used by the *Dr. Dobb's Journal* (DDJ) Data Compression Contest[2], data samples were selected from graphics, text, executable, and sound

categories. Although it is not very likely that MS-DOS executable programs will be stored in CDFs, these files have their own structure, and the goal is to find a general purpose compressor for all types of data, so all of the files used by the DDJ tests are included in the tests. In addition to these common file types, a subset of scientific data from the International Solar-Terrestrial Physics (ISTP) Key Parameter Data CD-ROM[3] was included since it represents the typical values and types of data that CDF actually stores, such as ISTP spacecraft orbit, attitude, magnetic field and particle measurements, and image data.

The compression programs evaluated are implementations for variations of the LZ77, LZ78, LZW, Huffman, run-length encoding, and arithmetic compression algorithms. Table 1 shows the programs that were included for the MS-DOS compression tests.

Program	Description
1 ahuff*	Adaptive Huffman coding
2 ar*	Haruhiki Okumura's archiver
3 arith*	Arithmetic coding
4 arith1*	Order-1 arithmetic coding
5 arjt	Robert Jung's ARJ, v2.41a
6 arth1e*	Order-1 arithmetic coding
7 cmp*	Linear conversion scheme for sound
8 compress†	UNIX compress uses LZW variant
9 cmp40-14†	Compress w/COMP40 option + 12-bit
10 comp40†	Compress with COMP40 option
11 comprs12†	Compress 12-bit
12 comprs14†	Compress 14-bit
13 comp_opt	Modified/optimized compress
14 copy	MS-DOS COPY command
15 dct*	Discrete cosine transformation
16 dogzip†	gzip-like LZ77 variant
17 doz†	UNIX-like LZW
18 gzip†	Gnu ZIP uses LZ77 variant[7]
19 huff†	Huffman coding
20 lha†	Harayasu Yoshizahi's LHA v2.13
21 lzari†	LZSS with adaptive arithmetic coding
22 lzhuff†	LZSS with adaptive Huffman coding
23 lzrw1	Ross Williams[5] LZ variant
24 lzrw3a	Ross Williams LZ variant
25 lzss†	Storer-Szymanski modified LZ77
26 lzss*	LZ77 with 12-bit sliding window
27 lzw12*	12-bit LZW compression
28 lzwcom†	Kent Williams LZW variant
29 pkarc†	PKARC v3.6
30 pkzip (default)†	PKZIP v2.04g LZ77 variant[7]
31 pkzip -ext†	PKZIP using maximal compression
32 rdc	Ross Data Compression uses run-length encoding[4]
33 snd*	Silence compression coding
34 zoo†	Rahul Dhesi's Zoo archiver, v2.1

* Source available on disks provided with the *Data Compression Book*[6].

† Source and/or executables available on SimTel anonymous ftp sites (e.g., oak.oakland.edu).

Table 1: Data Compression Programs Tested with CHURN

A compression utility program called CHURN, which accompanies Nelson's *Data Compression Book*[6], is capable of running compression and decompression programs on all data files in a specified disk volume for MS-DOS, measuring the elapsed compression and decompression times and verifying that the contents of the decompressed files match the original data files. CHURN is called with three arguments. The first argument is the drive letter and path name to recursively search for files to compress. The second parameter on the command line is the compression command. CHURN needs to compress the input file to a file called "TEST.CMP". The compression command tells CHURN how to do this. CHURN will execute the compression command by passing the command line to DOS using the system() function call. It inserts the file name into the compression command by calling sprintf(), with the file name as an argument. This means that if the compression command has a %s symbol anywhere in it, the name of the input file should be substituted for it. Finally, the third argument on the command line should be the command CHURN needs to spawn to decompress TEST.CMP to TEST.OUT. An example of how this works would look like this:

```
CHURN D:\DATA\ "LZSS-C %s test.cmp" "LZSS-D test.cmp test.out"
```

The double % symbols are present to defeat variable substitution under some command line interpreters such as 4DOS. A more complicated example testing PKZIP might look like this:

```
CHURN D:\DATA\ "PKZIP.BAT %s" "PKUNZIP TEST.CMP"
```

where PKZIP.BAT has two lines that look like this:

```
COPY %1 TEST.OUT  
PKZIP -M TEST.CMP TEST.OUT
```

CHURN creates a summary of compression results in a file called "CHURN.LOG". This file can be used for further analysis by other programs. This file is reformatted and used to generate the graphic output in Figure 1, where the numbers correspond to those in Table 1. CHURN and the corresponding compression programs were tested on an IBM compatible 80486-33 with MS-DOS 6.20.

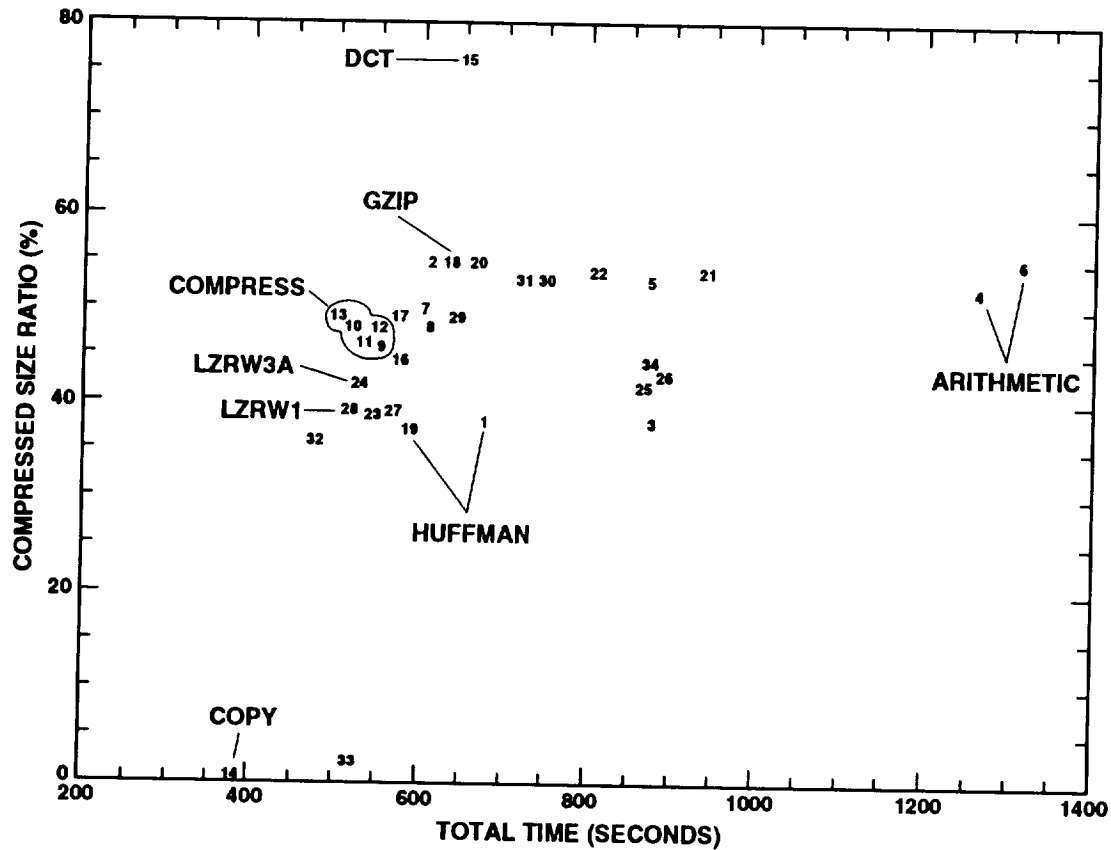


Figure 1. CHURN Compression Results for 30 Programs

The DOS COPY command, listed in Table 1 as item 14, measures the base-line time to copy a file byte-by-byte with no compression. No compression program should be able to read the entire input file, analyze the data, and compress the results in less time than a straightforward copy, but some compression programs actually approach this time with significant compression. Therefore, the overhead for compressing data is not going to impact performance if the faster compression algorithms are used. The DCT program appears to have the highest compression rate, but this is a lossy algorithm included in the test suite only for comparison since lossy algorithms do not meet the requirement for full reconstruction of the data. However, the COMPRESS, GZIP, LZRW1, and LZRW3A programs all achieve high compression rates with fast execution times.

Another test was conducted on six algorithms, listed below in Table 2, using the Computerized Reduction Using Selected Heuristics (CRUSH) data compression program. CRUSH is a portable, multi-method data compression utility that incorporates different algorithms into a common interface for VMS and UNIX systems. CRUSH takes a different approach than CHURN by linking different compression and decompression algorithms into one program rather than by calling external programs. CRUSH is a very flexible utility because users can either compress files using a specific compression algorithm or select automatic mode, which tries all algorithms and selects the method with the best compression. The automatic compression mode ensures that the data have the maximum compression given the available

algorithms but at a high cost of time. The algorithms included in CRUSH were developed by Ian H. Witten, Radford M. Neal, and John G. Cleary (Department of Computer Science, University of Calgary); David M. Abrahamson (Trinity College, Dublin, Ireland); Ross N. Williams (Renaissance Software, Adelaide, Australia); Robert F. Rice (Jet Propulsion Laboratory); and Pen-Shu Yeh and Warner Miller (Goddard Space Flight Center). CRUSH is public domain, and the source code for CRUSH is available via anonymous ftp from [dfnrc.gsfc.nasa.gov](ftp://dfnrc.gsfc.nasa.gov) (128.183.115.71) in the software/unix/crushv3 directory.

Method Name	Algorithm Description
ADAP	Adaptive dependency WNC method
LZC	Lempel/Ziv/Welch, ala UNIX compress
LZRW1	Fast Lempel-Ziv method[5]
LZRW3A	Fast Lempel-Ziv method
RICE	Rice machine
WNC	Witten/Neal/Cleary Arithmetic code

Table 2. Data Compression Algorithms Tested with CRUSH

These tests were run on a Sun 4 machine under SunOS 4.1. This machine, however, is not a dedicated machine and the elapsed times may have been slightly affected by other activity on the system, but the tests were run at off-hour times. CRUSH compresses each data file using each of the six algorithms. For tests of an individual data file, the best compression algorithm varied, but the overall results can be viewed in Figure 2. LZC has the highest compression ratio and one of the lowest times. The next best class of algorithms with a high best compression/time ratio is LZRW3A and LZRW1. WNC and ADAP both offer comparable compression ratios, and ADAP is even slightly better than LZC, but both algorithms take a much longer time. Overall, the RICE algorithm does not work well as a general purpose compressor, but it does work best on several individual files.

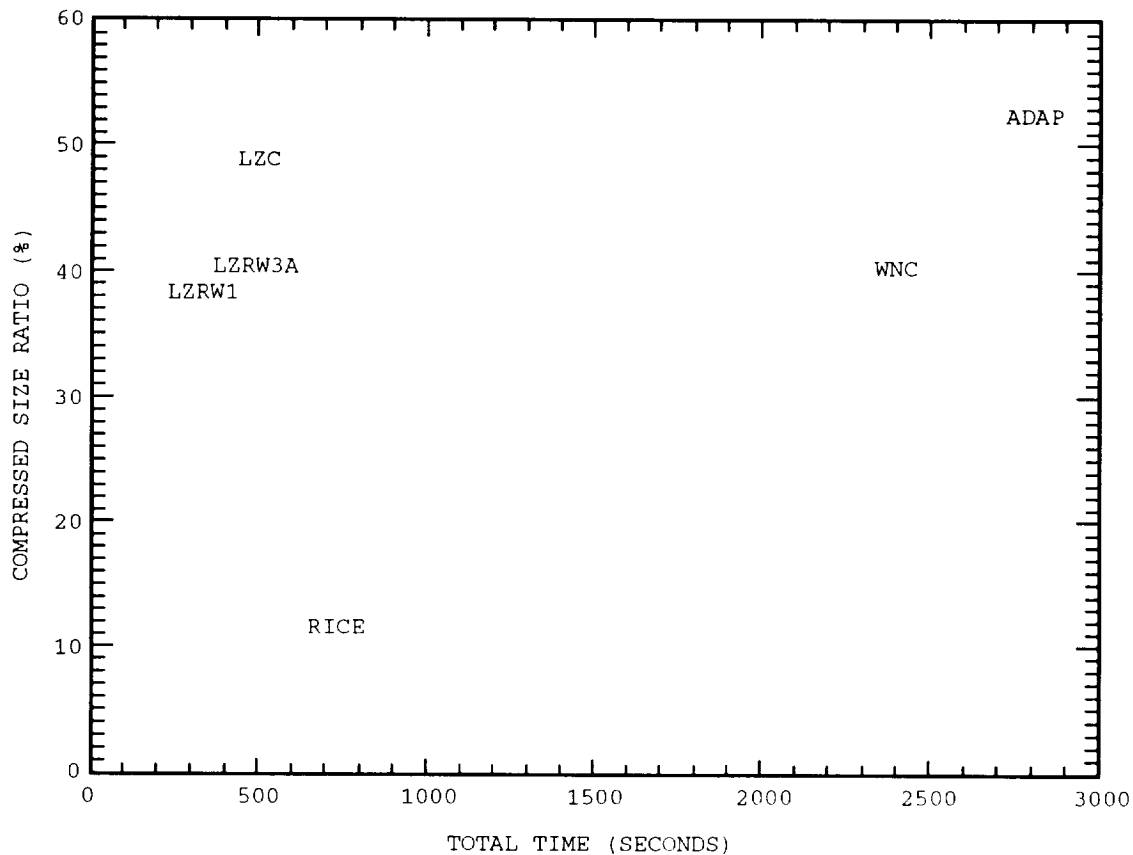


Figure 2. CRUSH Compression Results

The CRUSH program has a single interface with each compression algorithm having equivalent functionality, so the code complexity measurement can be estimated using the size of the corresponding object code files. The object code size is a rough approximation of complexity, but it gives some idea of how much is involved in the algorithm along with the other measurements. Since the compression programs tested on the IBM PC each have different functionality, error handling, and other bells and whistles, the object size and executable size are not very reliable measures of the algorithm's complexity, except where all algorithms are implemented with the same interface and equivalent error handling. For this reason, all complexity estimates will include only the algorithms available through the CRUSH interface, which has the equivalent interface and functionality among the six available algorithms.

The following table (Table 3) ranks each of the three measurements (compression ratio, speed, and complexity) from 1 to 4 (1 = least desirable and worst in its category; 4 = most desirable and best in its category). The compression ratio and speed (total elapsed time) ranks are extracted from the results of running CHURN with the test data and scaled appropriately. The overall score is a weighted sum (10 x compression rank + 10 x time rank + 5 x complexity rank) with a maximum of 100. The complexity weight is lower than the other two because it is the least reliable measurement. The weights can be adjusted appropriately if there is more or less priority on any of the three measurements.

Method	Compression Ratio	Speed	Complexity (Obj. Code Size)	Overall Score (Max=100)
LZC	4	3	2	80
LZRW1	2	4	4	80
LZRW3A	3	3	3	75
ADAP	4	1	3	65
WNC	2	1	2	40
RICE	1	2	2	40

Table 3. Ranking of Six Algorithms with CRUSH

The approaches taken by CRUSH and CHURN each have their own advantages and disadvantages. CRUSH compares compression algorithms and is itself a compression program in that it compresses and decompresses data files, but it is rather cumbersome to integrate many compression algorithms into CRUSH. CHURN is strictly a utility for testing compression programs, and it allows any compression and decompression programs to be called via command-line arguments. CHURN compresses any given input file to an arbitrary file named "TEST.CMP" and decompresses it to a file called "TEST.OUT". Some programs like PKZIP keep the original name of the input file, which require the overhead of copying the input file to a file called "TEST.OUT" as expected by CHURN and, thus, interfere with the overall compression times.

Another study that compares 47 compression programs over various large collections of data was conducted by Greg Flint of Purdue University. The raw test results are available via anonymous ftp from any SimTel archive site as arctst03.zip, such as at oak.oakland.edu (141.210.10.117) in the SimTel/msdos/info directory. Some preliminary examination of these results shows them to be consistent with those presented in this paper.

Results

The best class of programs is based on the Lempel-Ziv adaptive dictionary-based algorithm. Arithmetic compression is good in terms of compression ratio, but worst in terms of speed. The overall best are GZIP and LZC (UNIX Compress), which achieve the highest compression ratios and lowest elapsed time. As external stand-alone compression programs GZIP and LZC are the best all around in addition to being portable on MS-DOS, UNIX, VMS, and Macintosh platforms. The third comparison criterion requires small and simple functions to be incorporated into a much larger data management package. The two algorithms that meet all three criteria are the LZRW1 and LZRW3A algorithms, which are fast, good, simple algorithms.

Unfortunately the LZC implementation, based on the LZW algorithm, is covered by patents (US 4,464,650 and 4,558,302). The terms are defined under which Unisys will license the algorithm to modem manufacturers, but it has not stated how it applies to other types of products and this is one of the reasons for the CompuServe GIF/Unisys controversy. The LZRW algorithms are also covered by one or more patents. After speaking to the author of these two algorithms, it appears that the patents are thoroughly discouraging the implementation and use of many data compression techniques. The GZIP algorithm, however, is a variation of LZ77, free software, and free of patents. GZIP meets the first two criteria for a fast and good compression algorithm, but the code is neither simple nor straightforward for incorporating into a large data management package.

CDF is distributed worldwide as public domain software and part of several commercial data analysis and visualization packages, such as RSI's Interactive Data Language (IDL) and IBM's Visualization Data Explorer, so there may be some legal complications, but these legal issues are beyond the scope of this paper. Aside from these legal issues, these four LZ-based algorithms are the recommendations for good general-purpose data compression algorithms.

Future Plans

Up to now, data compression has been conducted on a collection of data files. The future plans for CDF are to incorporate the LZC and possibly another data compression algorithm into a test version of the CDF data management package, where compression and decompression will be transparent to CDF applications. Various techniques will be tried to determine what to compress for efficient accessing and minimizing the impact on performance.

Acknowledgments

Many thanks to Dr. Jim Thieman and Greg Goucher for their insight and guidance, Dr. Mona Kessel and Dr. Bill Mish for the ISTEP data, Mark Nelson for developing CHURN, Ed Seiler for developing CRUSH, Miranda Robinson for editing this paper.

Author

Mr. Gary Jason Mathews is a computer engineer at the NSSDC Interoperable Systems Office of the Goddard Space Flight Center (GSFC). He has earned a B.S. in computer science from Columbia University and an M.S. in computer science from the George Washington University, and has been at GSFC for 5 years. The focus of his research is the development of portable and reusable software tools. In addition to data compression research, he develops advanced scientific data analysis systems, WWW-based data systems, data management tools, and electronic information management systems that support NASA missions.

References

- [1] Goucher, G. W., and Mathews, G. J., *A Comprehensive Look at CDF*, NSSDC/WDC-A-R&S 94-07, August 1994. (See URL: http://nssdc.gsfc.nasa.gov/cdf/cdf_home.html)
- [2] Nelson, M., "DDJ Data Compression Contest Results," *Dr. Dobb's Journal*, November 1991, pp. 62-64.
- [3] Teague, M., "ISTP Key Parameter Available on CD-ROM," *Solar-Terrestrial Energy Program (STEP) International*, Vol. 4, No. 7, July 1994, pp. 4-5.
- [4] Ross, E., "A Simple Data Compression Technique," *The C User's Journal*, October 1992, pp. 113-120.
- [5] Williams, R. N., "An Extremely Fast ZIV-Lempel Data Compression Algorithm," *Proceedings Data Compression Conference '91*, pp. 362-371.
- [6] Nelson, M., *The Data Compression Book*, M&T Books, San Mateo, CA, 1992.

[7] Ziv, J., and Lempel, A., "A Universal Algorithm for Sequential Data Compression," *IEEE Transactions on Information Theory*, Vol. 23, No. 3, May 1977, pp. 337-343.

Compression Research on the REINAS Project*

Glen Langdon, Jr., Alex Pang, Craig M. Wittenbrink, Eric Rosen, William Macy, Bruce R. Montague, Carles Pi-Sunyer, Jim Spring, David Kulp, Dean Long, Bryan Mealy and Patrick Mantey

Baskin Center for Computer Engineering & Information Sciences
University of California, Santa Cruz
Santa Cruz, CA 95064

Abstract

We present approaches to integrating data compression technology into a database system designed to support research of air, sea, and land phenomena of interest to meteorology, oceanography, and earth science. A key element of the REINAS system is the real-time component: to provide data as soon as acquired. Compression approaches being considered for REINAS include compression of raw data on the way into the database, compression of data produced by scientific visualization on the way out of the database, compression of modeling results, and compression of database query results. These compression needs are being incorporated through client-server, API, utility, and application code development.

1 Introduction

The Real-Time Environmental Information Network and Analysis System (REINAS) [6, 7, 12] is a continuing computer science and engineering research and development program with the goal of designing, developing and testing an operational prototype system for data acquisition, data management, and visualization. The ultimate purpose of REINAS is to further science and the study of the environment with the availability and accessibility of a database for environmental data. The database content should aid meteorological and oceanographical studies in discovering new relationships, and in modeling and predicting weather and ocean phenomena. The Monterey Bay area is an example of coastal water, with land-sea interactions.

The project deals with and motivates many important and challenging problems for compression. Similar to global environmental research and earth mapping projects [2, 13], the large scale of data transfers between networks, data repositories, compute servers, and visualization systems can benefit from compression work. An additional

*This project is supported by ONR grant N00014-92-J-1807

challenge in the REINAS project is the need for real-time acquisition/access/display of environmental data by some of our users.

REINAS will support the real-time utilization of advanced instrumentation in environmental science. Advances in continuous time measurements and improved spatial resolution allow the monitoring and understanding of environmental phenomena in much greater detail than has previously been possible. The REINAS design also supports the retrospective use of integrated environmental data sets.

This paper presents some of the problems, studies, alternative solutions, and work in progress related to applying and integrating compression technology to REINAS.

2 The REINAS approach

The three major components of REINAS are: data acquisition, data management, and data visualization.

2.1 Data Acquisition

Data acquisition includes all the *plumbing* necessary to deliver data from various sources (in-situ instrumentations, remote sensors, observations, numerical simulations, etc.) to the REINAS system. The difficulties in meeting real-time requirements result from constraints of instruments' sampling frequencies, battery life which limits the frequency of transmission, and the quantity of data. A data acquisition concept (a software component) for placing data into the data base is called a "loadpath". Each type of data source requires a unique loadpath.

The data management component of REINAS provides APIs (application programming interface) or service calls to the system, to the data acquisition component. The loadpath employs API calls to specify the actions needed to place various types of data into the database.

2.2 Database Management

The second major component of REINAS is an environmental database management system that supports queries for low volume, real-time data feeds as well as high volume, retrospective data for analysis. The method for interfacing the user requests into the database system includes a step of formulating and presenting to the database a sequence of queries in SQL (Structured Query Language). Thus any database with an SQL interpreter could serve as a lower level REINAS component. Other interfaces include a library API interface and a variety of applications to support non-SQL users.

REINAS employs a commercial database system rather than reinventing one. The data management component (in fact REINAS itself) can be considered as *middleware* since REINAS provides a high-level interface to its users, and REINAS also sits on top of (and interfaces to) a commercial database of choice. At UCSC, the prototype employs an Oracle(R) database. At MBARI (Monterey Bay Aquarium Research Institute), the developers (Bruce Gritton) run the REINAS prototype on top of a Sybase(R) database system. REINAS developers provide the software drivers to access the commercial system.

The data management portion uses *meta-data* (such as pointers) in a heavy-weight directory structure on top of the commercial database. This component ingests

data into the database with the appropriate meta-data, and services temporal/spatial queries of environmental data. A key concept is the generic *science container*: which could be a time-series of measurements from a particular instrument. The data items can be scalar or vector. Data is tagged by latitude, longitude, elevation, and time. A container header provides information common to all the data items.

2.3 Visualization Front-end

The third major component of REINAS is a visualization front-end for monitoring current conditions, generating forecast products, and data analysis. It also supports collaboration among geographically distributed scientists to access and analyze distributed data. A visualization approach or tool called *Spray* [10] permits data to be colored and traced via a conceptual spray can under control of the user. For collaborative visualization, the tool is called *CSpray* [11].

3 Areas for Compression Support

In dealing with data compression, the typical scenario is that the compression algorithm is given a data file to compress, and the algorithm delivers a smaller output file containing the same information, or if not the same, a reasonable approximation to the original information. The REINAS project is a prototype, and relative to data compression, the scenario is not always as simple. Part of the learning experience concerns how to integrate compression into the database, which requires an understanding of the uses of the data. Current compression products for personal computers, for example, transparently compress and decompress data respectively into and out of the hard disk: *Stacker*(R), *DoubleSpace*(R), etc.

Some of our exploratory work has converted algorithms such as LZSS that compress an input file to an output file (FTF) into a compression algorithm employing an MTM (memory-to-memory) paradigm. The MTM paradigm can be used within an executable, where the compression part is an object module.

Another thrust being considered for the GOES feed, which uses features of the Unix environment, is a compression algorithm that accepts data from *standard input* (stdin) and delivers data to *standard output* (stdout).

Many objects in the database are *blobs* (binary large objects). Sometimes the motive for compression is to reduce the bandwidth required over communications lines, and other times the motivation is to save storage space. The objects subject to compression are varied, so no single compression algorithm can be considered to be the best.

The goal of compression is efficient end-to-end storage and transfer of information among different REINAS components and users. Tradeoffs to consider are the added time for doing compression and decompression, potential space savings, and resulting savings on transmission of compressed information. The common design goal in employing compression in REINAS is to make it transparent to the users. That is, compression is performed automatically if it is beneficial. There are several areas where compression can potentially benefit the REINAS system. These are discussed in the following sections.

3.1 Environmental Data Archive

REINAS scientists not only access data, but also the *meta-data* that determines data lineage and quality. In data assimilation research, the output from weather models are validated against measured data and are also used to drive the model. Data come from various sources including in-situ instruments, remote sensors, numerical model outputs, etc. Instruments come in a wide variety including simple meteorological (Met) stations which measure wind vector, temperature, rainfall, etc.; wind profilers which measure wind vectors at different heights; CODARs which measure ocean surface (1 meter deep) current vectors; ADCPs which measure ocean current, salinity, pressure, etc, at different depths; and AVHRR and GOES satellite images that provide periodic snapshots of a larger region. Available computer models of interest to REINAS scientists include the NORAPS [3] model for weather and the Mellor model for ocean.

With all the variety of data sources and their corresponding meta-data, the volume of information that REINAS deals with quickly becomes large. The plan for distributing met data is in “time-series containers” of about 600 KB each. Researchers can request data from particular instruments at a given time granularity. The intent is to send this information in compressed form. The approach for time series is predictive coding. One approach to lossless compression [5] has its genesis with one of the authors (Langdon) while on a NASA Summer Faculty Fellowship. The approach is applicable to most one-dimensional digitized waveforms, as well as to two-dimensional waveforms. Moreover, nothing in the approach precludes its future employment at the met stations themselves.

3.2 Instrument Networks

REINAS supports a wide range of instruments. Their characteristics range from continuous sampling of meteorological data to periodic snapshots. The transmission of data from instruments to the REINAS load paths can also range from continuous to periodic, and in small chunks or large chunks. A specific instance includes instruments mounted on buoys that continually log data to local disk, but can only afford to do transmission at fixed time intervals or on demand due to considerations for battery supply. New routing protocols are being developed for use in the REINAS distributed instrument network. The use of compression in networked communication is obvious, and especially important because of the low bandwidth available to the packet radio modem technology. For additional information on ongoing networking research see the REINAS homepage [12].

3.3 Multispectral Images from Satellites

Our current plan is to store GOES satellite images covering Monterey Bay in the REINAS project. Since the data is so voluminous, instead of storing the images themselves in the data base, the plan is to store the names and pointers identifying the location of the compressed images.

The source of the GOES images are geo-stationary satellites located above the United States. We are using primarily GOES-West. These images have five bands of interest. From the data source, the GOES images are passed through an archive path

that compresses the GOES image in its raw state, losslessly. The current algorithm is *gzip*. However, to be useful, the images must be registered to the latitude-longitude-time scheme used by the data from other sensors feeding the database. Thus, a second part of the task is to “warp” the raw data to the grid, and store the result in HDF format [8]. Converting the GOES images for REINAS use is presently in the implementation prototype stage.

For satellite images that are multispectral, Hotelling’s method of principal components, or EOFs (elementary orthogonal functions), or discrete Karhunen-Loeve transform, is being investigated (G. Ubierno).

The compression approach for registered images can have some loss since registration is already lossy. However, the original data is archived. The current idea is to segment the multispectral image, and identify the region boundaries. The boundary information may indicate the cloud cover, via segments that hide natural boundaries. We also plan to employ the transform on the regions themselves, and have basis matrices on a region basis.

We plan to use knowledge of the terrain as well. For example, knowing the location of a region, and estimating the cloud cover, it may be possible to develop a library of basis matrices. This would save the need to store them with the image itself. An alternative being considered is gathering expected values for Monterey Bay for a predictive coding approach.

3.4 Video Camera

Another type of instrument being employed is the video camera. Monterey Bay has available video capture from an underwater submarine. There is also a video camera being installed on the roof of the Dream Inn (a beach front hotel in Santa Cruz, CA), to provide a panoramic view of the Monterey Bay. Since the robot-controlled video camera captures weather phenomena as it happens. The idea applies to watching weather phenomena as it develops, and observe the values of the instruments at the same time, whether the observer is watching it live, or watching a coordinated and realistic “playback” ten years hence.

The idea of a permanent location for the camera suggests building a large image of what the camera is expected to see in a certain direction under various weather conditions. Thus the compression could include a predictive part.

3.5 Instruments in remote locations

Several of the instruments are located where only packet radio is available for the “first hop” on the way to the REINAS data base. Compression algorithms are being investigated for this application. However, the met data itself, on two minute averages, does not seem to present a bandwidth problem at this time.

3.6 Model Output

This class of data are more predictable and can benefit greatly from compression. Forecast models are typically run daily and produce predictions for different time periods for the following day. The output formats and statistical properties of the data are usually stable. For instance, the NORAPS model runs on a supercomputer and has its output available daily. The model output consists of predictions for the

temperature, pressure, wind, and other fields every several hours for the next 24 hours. Currently, files containing the model output are transferred to the REINAS system manually (compress + ftp). Ideally, the feed from the model output into the REINAS database should be automatic and transparent to the user as well as to the person maintaining REINAS data. Currently, at UCSC, the Mellor oceanographic model is being applied to Monterey Bay by H. Kolsky.

3.7 Accessing REINAS over the Internet

With commercial (for profit) companies being the providers of communications services for the Internet, clearly each byte transmitted over the net has a cost that ultimately is borne by the user. Thus, compressed data is a desirable option for all users. In addition to reducing the connect fees, effective compression can also reduce latency for low bandwidth clients. In the remainder of this section, we detail several applications of compression for accessing REINAS data over the Internet.

Collaborative Visualization

The visualization component allows multiple users to access data stored within the databases, allowing simultaneous collaboration among several geographically distributed scientists. Collaboration over the Internet can happen at several levels depending on the willingness of the collaborators to share data, on the processing powers of the workstations, and on the available network bandwidth.

The scenario has data located a sufficient distance from the scientist (or scientists in case of collaborative viewing) so that communication bandwidth is needed. Compression at the host offers bandwidth savings, assuming each workstation can compress and decompress. Moreover, if the screen is broadcast to several collaborating users, the bandwidth savings are multiplied.

Visualization options considered are:

1. Transfer the scientific data to be rendered to each graphics workstation, assuming each has a renderer.
2. Transfer the graphics primitives, or visualization primitives, to the users and allow them to render the image.
3. Transfer a copy of the rendered image.

In the first scenario, changes in the viewing position may involve very little Internet traffic, during a collaboration. However, broadcasting the raw data can be overly expensive.

In the second scenario, collaborators that have a rendering engine but cannot directly access the raw data on their own machines, can still make requests to the owner's machine to generate visualization primitives for display on their own machines.

In the third scenario, an observer may only have an image decompression program. Here, a simple and quick decompression algorithm reduces the time from the arrival of the data to the time it is viewed. One of the simplest and fastest compression algorithms for images, in terms of the decoder, is Vector Quantization (VQ). An investigation into the theory and practice of VQ was done, and an algorithm was obtained and modified for experimental use in REINAS. Other decompressors studied

include JPEG, a simple lossless algorithm called FELICS of Howard and Vitter [4], and a combination VQ-based and BTC algorithm called VPIC (Visual Pattern Image Coding) of Alan Bovik and his students [1].

With a greater demand on the CPU power of the workstation (inverse Discrete Cosine Transform calculations), then another popular and widely available image compression technique is a possible choice. An emerging ISO/ITU standard called JPEG has free software available (which increases its popularity), and manufacturers are also providing hardware cards. We have obtained the free source code, and experimented with it. The approach is to compare the characteristics of JPEG with VQ to determine advantages and disadvantages.

To the VQ or JPEG compression scheme however, a slight rotation of a 3-D image involves the compression and decompression of a completely new image. If we know that a set of data are to be scientifically visualized in collaboration, then perhaps we should consider the first scenario to compress the data set and broadcast the data to all workstations. Then a second compression algorithm applicable to higher-level commands to render the data in certain ways can be devised.

Clearly, three opportunities for compression to aid collaborative visualization need to be investigated: compression of raw data, compression of geometric primitives describing the image to a renderer, and compression of the rendered images themselves.

As a default compression algorithm, the gzip program has been tried. To simplify implementation, we have investigated using shared memory as input and output buffers for the compressor.

Where collaborators are remotely located at rendering workstations, we have sample data streams of the graphics primitives for test purposes. In a test involving two algorithms, FELICS and gzip, some interesting results were obtained. For ordinary images, both FELICS and gzip compressing the DPCM version of the image did better than compressing the pixel values themselves. The opposite was true of the collaborative image: both gzip and FELICS did better using the original values instead of the differences.

Experiments with visualization primitives (Mealy) resulted in conclusions that the buffer sizes used for socket implementations can significantly impact performance. Visualization primitives are stored in non-contiguous memory, because they are generated on the fly and asynchronously. Because of this, for full compression, a compaction into contiguous memory is done first followed by *gzip* compression in memory.

Accessing Images from REINAS

We are developing part of the database as an image library. Users can obtain a small picture of the image before acquiring all of the data using progressive transmission. We have an experimental client/server program under Xwindows that first transmits the mean value of each 16x16 block of the image, and progresses from there. If the user decides not to continue, a mouse click terminates the transmission.

Accessing time-series containers and other database queries

A study was performed on the feasibility of compressing the outcomes of database queries (Pi-Sunyer). The SQL returns a record at a time, so to get statistical correlations, the study adapted the LZSS algorithm provided by Mark Nelson [9]. The

algorithm is an LZ77 class algorithm, and maintains a rotating history buffer of the previous bytes sent to the client. Since the history buffer may be larger than each record, both the encoder and decoder remember the state from the previous records sent and received. Thus, a session consisting of a sequence of queries between the same client and server maintains a state that persists between queries. In the simple experiment, the compressed data transferred amounted to about 38% of that which would have been transferred over the network in the absence of compression.

4 Summary and Conclusions

The general type of data found in the REINAS database is described, along with possible applications.

The integration of data compression into the database is progressing. Several key areas have been identified for further work. Some popular algorithms have been explored. Xwindow client-server compression has been explored for accessing compressed images from REINAS.

Several alternatives for the visualization scenario have been discussed; compressing graphics primitives, transmitting the data to all collaborators, communicating a 2D image representing a rendered screen. This work is on-going.

The GOES images are multispectral, and represent a considerable amount of data. Several alternatives are under consideration for compressing the registered data.

The compression of data leaving the database to the site of a user over the network in response to user requests has been studied and seems viable.

Acknowledgments

Additional students who have participated in compression research on REINAS include Tom Goodman, Rob Antonucci and Devdutt Sheth. We thank the many colleagues involved in this collaborative effort, especially our science colleagues Professor Wendell Nuss and Jeff Paduan of NPS, Francesco Chavez of MBARI, and Dan Fernandez of UCSC. Led by Professor Darrell Long, the systems group has been instrumental in providing the foundation from which to develop a real application test bed. Many of the key properties and features of REINAS in the early years were motivated by Bruce Gritton of MBARI. The visualization team has provided much of the coding, and development for applications on which we have based our studies including Naim Alper, Jeff Furman, Elijah Saxon, and Michael Clifton.

References

- [1] D. Chen and A. Bovik. Hierarchical visual pattern image coding. *IEEE Transactions on Communications*, 40(4):671–675, Apr. 1992.
- [2] N. D. Gershon and C. G. Miller. Dealing with the data deluge. Special report: Environment, part 2. *IEEE Spectrum*, 30(7):28–32, July 1993.

- [3] R. M. Hodur. Evaluation of a regional model with an update cycle. *Monthly Weather Review*, 115(11):2707–2718, Nov. 1987.
- [4] P. Howard and J. Vitter. Fast and efficient lossless image compression. In J. Storer and M. Cohn, editors, *DCC '93. Data Compression Conference*, pages 351–360. IEEE Computer Society, 1993.
- [5] G. Langdon Jr. and C. Haidinyak. Experiments with lossless and virtually lossless image compression algorithms. In M. Rabbani et al., editors, *Proceedings SPIE, Vol. 2418, Still Image Compression*, pages 21–27. SPIE, 1995.
- [6] D. Long, P. Mantey, C. M. Wittenbrink, T. Haining, and B. Montague. REINAS the real-time environmental information network and analysis system. In *Proceedings of COMPCON*, pages 482–487, San Francisco, CA, Mar. 1995. IEEE.
- [7] P. Mantey et al. REINAS: Real-time environmental information network and analysis system: Phase IV - experimentation. Technical report, CIS Board, University of California, Santa Cruz, 1994. UCSC-CRL-94-43.
- [8] NCSA. Hierarchical data format (hdf) specification. <ftp.ncsa.uiuc.edu>, 1995.
- [9] M. Nelson. *The data compression book : featuring fast, efficient data compression techniques in C*. M&T Books, Redwood City, CA, 1991.
- [10] A. Pang. Spray rendering. *IEEE Computer Graphics and Applications*, 14(5):57 – 63, 1994.
- [11] A. Pang, C. M. Wittenbrink, and T. Goodman. CSpray: A collaborative scientific visualization application. In *Proceedings SPIE IS & T's Conference Proceedings on Electronic Imaging: Multimedia Computing and Networking*, volume 2417, pages 317–326, Feb. 1995.
- [12] E. Rosen. REINAS: Real-time environmental information network and analysis system: Home page. World Wide Web URL: <http://csl.cse.ucsc.edu/reinas.html>, 1995.
- [13] M. Stonebraker. Sequoia 2000: A reflection on the first three years. *IEEE Computational Science and Engineering*, 1(4):63–72, Winter 1994.

KRESKA: A COMPRESSION SYSTEM FOR SMALL AND VERY LARGE IMAGES

Krystyna W. Ohnesorge and René Sennhauser

*MultiMedia Laboratory, Department of Computer Science, University of Zürich,
Winterthurerstrasse 190, CH-8057 Zürich, Switzerland
email: {ohnesorge, sennhaus}@ifi.unizh.ch*

58-61
5308
p-15

ABSTRACT

An effective lossless compression system for grayscale images is presented using finite context variable order Markov models. A new method to accurately estimate the probability of the escape symbol is proposed. The choice of the best model order and rules for selecting context pixels are discussed. Two context precision and two symbol precision techniques to handle noisy image data with Markov models are introduced. Results indicate that finite context variable order Markov models lead to effective lossless compression systems for small and very large images. The system achieves higher compression ratios than some of the better known image compression techniques such as lossless JPEG, JBIG, or FELICS.

1 INTRODUCTION

Digitized images require huge amounts of computer storage and large transmission bandwidth since they contain redundant information and are therefore much larger than the information contained therein. A more compact representation of images can be achieved if either finer image details are not stored or redundant information is removed from the image. If some of the original image's details are not kept, the image is not reproducible exactly (*lossless compression*), although often the difference to the original image is not visible to the human eye (*visual lossless compression*). By removing redundant information from an image, the image can be reproduced without loss, i.e. the decompressed image is an exact bit-per-bit copy of the original image (*lossless compression*). Although the achievable compression ratio is substantially higher for lossy or visual lossless compression, there are applications where it may be not acceptable to tolerate discrepancies from the original image. These applications require an exact reproduction of the image. For example, auto-

matic evaluation of digitized images yields different interpretations in dependence of the used lossy compression scheme since the decompressed image contains subtle modifications. Digital image operations therefore produce different results for different decompressed images. As an other example, the long term archival of digitized images requires the images to be stored in their original form since future applications cannot be foreseen. Hence, these images should be archived in their original form or compressed losslessly.

Currently used image sizes differ from about 256×256 up to 2048×2048 . From a data compression point of view, image sizes can coarsely be classified in small, medium, large and very large images. Nowadays, images of 256×256 pixels are considered to be small. Images with 512×512 pixels can be regarded as medium sized, and images with 1024×1024 pixels are large. Images larger than 2048×2048 pixels are not yet used very often since their computer processing requires extensive computing resources. Therefore, they are considered to be very large.

2 LOSSLESS IMAGE COMPRESSION

General data compression techniques such LZW schemes [Ziv et Lempel 78] do not achieve optimal data reduction for continuous-tone images since these methods do not exploit image specific characteristics. To obtain higher compression ratios, data compression techniques should be adapted to the characteristics of natural images.

For lossless image compression, several approaches have been proposed and theoretically as well as experimentally investigated [Arps et Truong 94]. Along with the standardized lossless JPEG [Pennebaker et Mitchell 93] and JBIG [CCITT 93] compression systems, one of the better known image compression systems is the FELICS system [Howard et Vitter 93]. Lossless JPEG uses one of seven predictors to predict the next pixel and encodes the difference between the pixel and its prediction using either Huffman or arithmetic coding. JBIG operates for grayscale images on Gray coded images and encodes each bit plane separately using a scheme developed specifically for bilevel images. For grayscale images, the performance of JBIG is optimal if the image has low grayscale resolution. FELICS uses a sophisticated error modelling technique and encodes each pixel using either a Rice code or quasi-arithmetic coding. FELICS has been reported to compress slightly better than the lossless modes of JPEG. However, the intricacies of the three systems sketched above are not easily accessible. Lossless JPEG requires manual selection of the best predictor for optimal compression and relies on a non intuitive coding scheme, JBIG uses a series of highly optimized processing steps, and FELICS incorporates an effective parameter selection technique for Rice codes as well as a sophisticated Laplacian error modelling method.

3 HIGHER ORDER MARKOV MODELS FOR IMAGE COMPRESSION

Markov Models have proven to achieve good compression ratios for text. Prediction by Partial Matching (PPM) uses finite context Markov models for characters. The SAKDC (Swiss Army Knife Data Compression) text compression system is a representative of PPM and uses a set of about 25 parameters to control the compression process [Williams 91], [Witten 94]. Since most of the parameters directly refer to the model and because of the large number of parameters, SAKDC may be unsuitable for general use.

Because of their success for text compression, Markov models should be applied to image compression [Ohnesorge et al 93]. Unfortunately, implementation of finite context Markov models for lossless image compression involves some subtle points to yield optimal compression ratios. In this paper, four points are discussed that substantially improve the compression achieved for images using finite context Markov models. First, a new escape probability estimation called KRESC is proposed that is better adapted to image compression and yields higher data reduction. Since fixed order Markov models require the probability distributions for each context to be initialized to a start model such as the equal probability distribution or a precomputed probability distribution, the achievable compression ratios are generally lower than for variable order Markov models. With variable order Markov models, the amount of compression achieved is affected by the method used to estimate the probability of a novel event. Hence, an accurate estimation of the escape probability plays a central role. Second, higher order Markov models obtain better compression ratios for certain model orders only. Increasing the model order does not necessarily yield improved compression. This paper investigates different model orders for variable order Markov models and attempts to find an optimal model order. Third, finite context Markov models for lossless image compression can use different already coded pixels. The nearest already encoded neighbours of the pixel to be encoded usually yield the best compression. This paper proposes a ranking of context pixels to maximize the compression ratio. Fourth, inevitable noise in digitized image data distorts the probability distribution in such a way that the achievable compression ratio is lower compared to noiseless images. Traditionally, noise in images has been reduced using appropriate image filtering methods. Lossless image compression is then achieved by separately encoding the filtered image data and the difference to the original data. In this paper, two novel techniques to compress noisy images are proposed that achieve substantially improved compression. The best parameter settings always obtain improved performance compared to other image compression techniques such as JPEG or FELICS. Although the proposed system is very flexible and highly adaptable to images of different sizes, it is extremely user friendly since there are generally only two parameters to be adjusted which are derivable from a visual inspection of the image.

4 THE KRESKA COMPRESSION SYSTEM

The Kreska system for lossless image compression is a straightforward implementation of finite context variable order Markov models. It uses a new escape probability estimation technique developed specifically for images and applies novel techniques to handle noise in image data [Ohnesorge 95].

4.1 Escape Technique

Finite context variable order Markov models use a fixed set of already encoded symbols as context to predict the next symbol, and combine multiple model orders together [Bell et al 90]. The model estimates the probability distributions of symbols conditioned on fixed-length contexts. Any model order encompasses all symbol probabilities conditioned on contexts of the same length. In adaptive modelling, encoding a symbol starts at the highest model order. If the symbol has already been seen under the current context, it is encoded using the probability distribution of this context. Otherwise, a novel symbol is encountered. An escape symbol has to be encoded before an attempt is made to encode the symbol using the probability distribution of the next lower model order. The probability of encountering a previously unseen symbol is called the escape probability [Witten et al 87], [Witten et Bell 91].

For text compression some escape probability estimation methods have been established to achieve high compression ratios. PPMA is one of the simplest methods to estimate the probability of the escape symbol. It sets the frequency of occurrence of the escape symbol to 1 [Witten et al 87]. PPMC is a better approach which has proven to be superior to PPMA. It estimates the probability as $d/(n+d)$ where d is the number of distinct symbols seen so far and n is the total number of symbols [Moffat 90]. There are other variants of PPMx escape probability estimation techniques such as PPMD which are not yet as widely used [Howard 93]. A finer escape probability estimation is based on a combination of three factors which influence the probability of the escape symbol [Nelson 91]. For the following discussion, an alphabet of 256 different symbols is used.

- 1) *The number of distinct symbols.* The more distinct symbols occur in a probability distribution, the lower the probability for a novel symbol will be. On the other hand, if there are a lot of distinct symbols in a probability distribution, the probability of other not yet seen symbols should be high. The first factor is given by

$$Factor_1 = (256 - NumOfDistinctSymbols) \cdot NumOfDistinctSymbols. \quad (1)$$

- 2) *A measure of randomness of the probability distribution.* The more random a probability distribution is, the higher the escape probability should be. The ran-

domness of a probability distribution can be measured by the proportion of the maximum frequency to the average frequency of the symbols. The second factor is given by

$$\begin{aligned} Factor_2 &= \frac{1}{\frac{MaximumFrequency}{AverageFrequency}} \\ &= \frac{1}{\frac{\sum_{i=0}^{255} Frequency(s_i)}{256 \cdot MaximumFrequency}} \end{aligned} \quad (2)$$

3) *The total number of symbols.* The more symbols have already been encoded using a probability distribution, the more relevant these probabilities will be, and the lower the escape probability should be. The third factor is given by

$$Factor_3 = \frac{1}{\sum_{i=0}^{255} Frequency(s_i)} \quad (3)$$

By combining these three factors, the frequency of occurrence of the escape symbol in a probability distribution is computed as

$$freq(ESC) = \frac{(256 - NumOfDistinctSymbols) \cdot NumOfDistinctSymbols}{256 \cdot MaximumFrequency} \quad (4)$$

Then, the escape probability is given by

$$p(ESC) = \frac{freq(ESC)}{TotalNumOfSymbols + freq(ESC)} \quad (5)$$

Experiments for a large set of grayscale images with this formula have shown that the escape probability can advantageously be increased by a constant factor. In the experiments, factor 3 has emerged to yield the best compression results. This formula is further referred to as NELSON3. Although computationally more expensive, all better optimized escape probability estimations attempt to consider a set of different factors. The formula should fulfil the two requirements so that 1) the escape symbol is encoded effectively if it is used and 2) the probability distribution is distorted only marginally if it is not used. A new escape probability estimation termed KRESC is even better suited to the compression of natural images. Additionally to NELSON3, KRESC includes the minimum frequency of a single symbol and the proportion of symbols in the current order of the Markov model to the number of symbols one order below. The frequency of occurrence of the escape symbol in a probability distribution is then computed by

$$\begin{aligned}
freq_{KRESC}(ESC) = & \frac{NumOfDistinctSymbols_{NextLowerOrder}}{NumOfDistinctSymbols_{CurrentOrder}} \\
& \times \frac{(256 - NumOfDistinctSymbols)}{256} \\
& \times \frac{NumOfDistinctSymbols \cdot MinimumFrequency}{MaximumFrequency + MinimumFrequency}.
\end{aligned} \tag{6}$$

If the current order contains just a few symbols and the next lower order many, the probability will be high that an escape symbol has to be encoded. If the minimum and maximum frequency differ greatly, the randomness of the probability distribution is low and the escape probability will be low, too. If, on the other hand, minimum and maximum frequency are about the same, the randomness of the probability distribution is high, and the escape probability will also be high. Therefore, the third factor in the new formula weights the randomness of a probability distribution.

4.2 Model Order

In this paper, variable order Markov models for lossless image compression have been investigated for various orders. A zero order Markov model does not use a context. A first order Markov model uses one context for each symbol to be encoded, a second order Markov model uses two contexts, etc. For each context, a separate symbol probability distribution is estimated. Initially, all probability distribution are empty except for the escape symbol. If the next symbol to be encoded has not yet been seen in the current context, an escape symbol is encoded and the model order is decreased. This process is repeated until the symbol is found in a lower order, or, if is not contained in any order, a default context will be used.

Higher order Markov models generally achieve improved compression for text if the model order is increased up to about 3 or 4 [Williams 91]. Higher model orders use larger contexts to predict the next symbol. The predictions should therefore be more specific. Larger model orders stand a greater chance of not giving rise to any prediction at all. More escape symbols are then encoded to reduce the model order. Each escape symbol reduces the coding efficiency slightly. To accumulate accurate higher order statistics, enough data has to be processed. That is why larger model orders are less suited for very small data. Because of the high redundancy in text data, there are only few contexts used. Hence, medium order Markov models contain accurate enough statistics to compress well for texts of some ten or hundred thousand characters. Obviously, there is a trade-off between data size and model order as well as redundancy contained in the data. The larger the data is, the higher the order of the Markov model can be, supposing there is enough redundancy in the data.

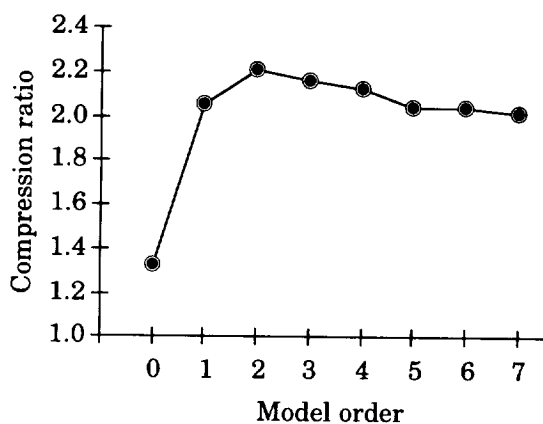


Figure 1: Compression ratios for test image *bodensee* for different model orders.

Unfortunately, higher order Markov models do not provide the same compression gain for images as for text if the model order is increased. This is due to the higher number of contexts yielding a less reliable probability distribution. Further, the number of distinct symbols is larger in a typical probability distribution for images than for texts. Hence, higher order Markov models are less suited to image compression. As a consequence, zero order and first order Markov models have been widely applied to image compression, or symbol alphabets with highly skewed frequency of occurrence of the individual symbols have been used. For example, PPPM (Prediction by Partial Precision Matching) uses a fourth order Markov model with error contexts [Howard 93].

If the probability distributions get sparser, more escape symbols have to be encoded therefore decreasing the achievable compression ratio. Hence, the model order should be selected carefully. Our experiments with a large test set of images indicate that second order Markov models always yield higher compression ratios than first order models. Third order Markov models occasionally achieve a slightly better compression than second order models. Figure 1 shows as an example of how the compression ratio for the test image *bodensee* varies when different model orders are used. The graph shows that the best compression is achieved when a model order of two is chosen and that it deteriorates slightly when the model order is increased beyond this.

4.3 Templates

Higher order Markov models use a number of contexts during the encoding of the pixels. A separate probability distribution is associated with each context. Generally, the nearest neighbouring pixels already encoded are used as context pixels.

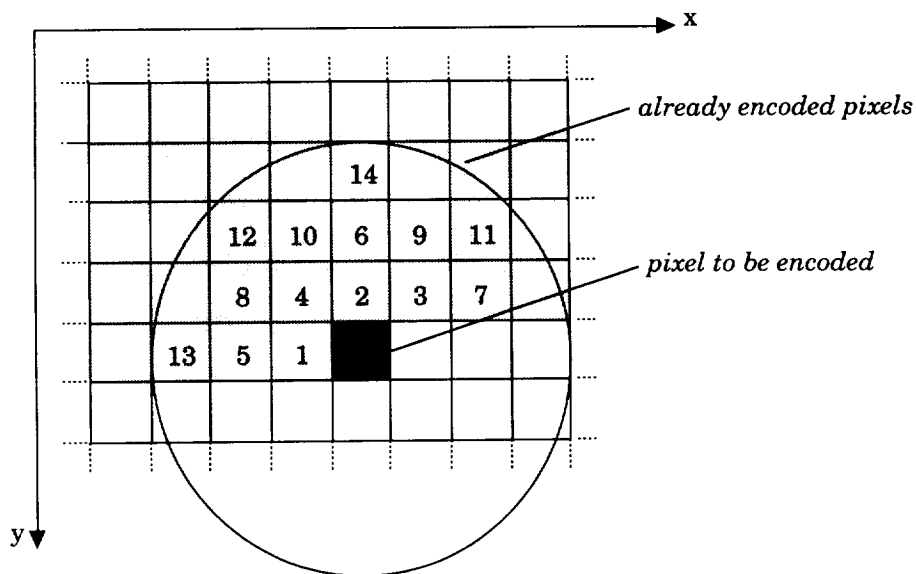


Figure 2: Context template T . Pixels to be used in the computation of the context values should be selected in the order given by the context template.

The higher the model order that is used, the more context pixels are used. Context pixels should on the one hand be highly correlated to the pixels to be encoded and on the other hand loosely correlated among themselves. There are two pixels of equal distance in a first order model, namely the pixel to the west and the pixel to the north of the pixel to be encoded. Of course, these two context pixels are symmetrical and can be chosen arbitrarily. In a second order model, there are four pixels within a small neighbourhood of the current pixel if symmetrical situations are excluded. The best compression is achieved if the two nearest neighbouring pixels already encoded are selected as context pixels, namely the pixel to the west and the pixel to the north [Ohnesorge et al 93].

Instead of using pixel combinations directly as context, they can be preprocessed. For example, in a first order Markov model the compression is improved if the average of the two nearest neighbouring pixels is used as context value [Ohnesorge 95]. Hence, higher order models may use precomputed *context values* from a larger neighbourhood of the current pixel. The selection of the pixels to be used in the computation of the context values should follow the principle sketched above. On the one hand only pixels should be selected that are highly correlated to the pixel to be encoded, and on the other hand the context pixels should be decorrelated. The context template T orders the neighbouring pixels and has proven to be successful in selecting the image pixels to be used in the computation of the context values (Figure 2). The context template indicates the order in which the pixels should be selected for the computation of context values.

4.4 Precision

Noise in digitized images distorts the probability distributions used to encode pixels. The achievable compression is therefore lower, the noisier the image is. Noise in images can be reduced by appropriate image filtering procedures. Lossless compression can then be achieved by separately encoding the preprocessed (filtered) image and the difference to the original image. Whereas the filtered image yields high compression ratios, the difference image is almost incompressible because it contains nearly random data.

In this paper, context precision and symbol precision are introduced as two novel techniques that achieve substantially improved compression ratios for noisy images. Noise is introduced during the digitization of an image and means that spurious effects modify the grayvalues of pixels. In a compression system based on Markov models, the contexts as well as the probability distributions are affected by noise resulting in reduced compression ratios. Because of small deviations of the true grayvalues of pixels, noisy images contain more different contexts. This has two consequences. First, the symbols to be encoded are scattered among more contexts, and second, the probability distributions of contexts are less relevant for the image data. Since noise modifies the true grayvalues of pixels, the probability of a symbol is not accurately captured in a probability distribution. An improved compression can be obtained if 1) noise is removed or at least reduced from the context pixels and 2) the probability distributions are smoothed.

Reduction of noise in context pixels can be attained by using reduced pixel precision or considering fuzzy pixel contexts where grayvalues are considered to be smeared. Pixels are encoded using the probability distributions of different contexts. If the content of an image changes considerably, very different contexts are used. In a small area of an image, the image content generally changes only a little. Therefore, a small subset of all contexts will usually be used for the encoding of the pixels within a small image area. Noise introduced during the image acquisition process leads to small, arbitrary modifications of the context pixels. The least significant bits of the context pixels have therefore arbitrary values and can be masked from the context pixels. Context pixels with the same most significant bits get together and form just one context value. Masking of the least significant bits is the simplest and most efficient technique to combine noisy context pixels. This context precision technique will be called strategy *P1* and is computed as

$$K_{P1}(\text{Contextpixel}, \text{Prec}) = \text{Contextpixel} \wedge 2^8 \cdot (1 - 2^{-\text{Prec}}), \quad (7)$$

where *Prec* indicates the precision of the context pixel, i.e. the number of bits that are not considered to be noisy. Masking of the least significant bits means that the grayvalues of the context pixels are always mapped to equal or smaller values.

Since noise effects can increase the grayvalue of a pixel, a more effective context precision strategy *P2* uses not only smaller values but also larger values. This strategy combines the context pixels whose grayvalues lie within a small range around the grayvalue of the pixel selected as context pixel from the image. Formula (8) determines the grayvalues of the context pixels to be combined.

$$K_{P2}(\text{Contextpixel}, \text{Prec}) = \{k | k \in (\text{Contextpixel} - (9 - \text{Prec}), \text{Contextpixel} + (9 - \text{Prec}))\} \quad (8)$$

Both strategies use an argument *Prec* that specifies the number of context pixels to be combined, where $\text{Prec} \in \{0, \dots, 8\}$. For the strategy *P1*, *Prec* denotes the number of most significant bits of the context pixels that are not modified by the strategy. By masking the most significant bits, the least significant bits are set to zero. If strategy *P1* and 7 bit precision is used in a first order Markov model, two contexts are combined, and in a second order Markov model, four contexts are combined. For the strategy *P2*, *Prec* specifies the range of grayvalues around the grayvalue of the context pixel that should be combined. For a first order Markov model and 7 Bit precision, strategy *P2* combines three contexts, and for a second order Markov Model, nine contexts are combined. The smaller the argument *Prec* is chosen, the more context pixels are combined. For example, strategy *P1* does not combine any contexts with 8 Bit precision, and all contexts are combined to a single context of the next lower order if *Prec* is set to 0. These two context precision techniques lead to drastically improved compression for noisy images. Whereas reduced pixel precision (strategy *P1*) is faster, fuzzy contexts (strategy *P2*) achieve better compression.

Smoothing of the probability distributions can be achieved by entering into a probability distribution the grayvalue of the encoded pixel and some additional nearby grayvalues. In a first symbol precision strategy *Q1*, additionally to the grayvalue of the encoded pixel, all grayvalues with the same most significant bits and all permutations of the least significant bits are entered into the current probability distribution (Figure 3a). A second symbol precision strategy *Q2* enters all grayvalues within a small range around the grayvalue of the pixel to be encoded into the probability distribution (Figure 3b). Therefore, this strategy takes into account that noise increases or decreases the true grayvalue of a pixel. For both strategies, the number of additional symbols entered into a probability distribution is given by an argument *Prec*. For the strategy *Q1*, *Prec* denotes the number of most significant bits that are kept constant, and for the strategy *Q2*, *Prec* specifies the range of grayvalues around the grayvalue of the encoded pixel that should be entered additionally. Both of these symbol precision techniques improve the attainable compression.

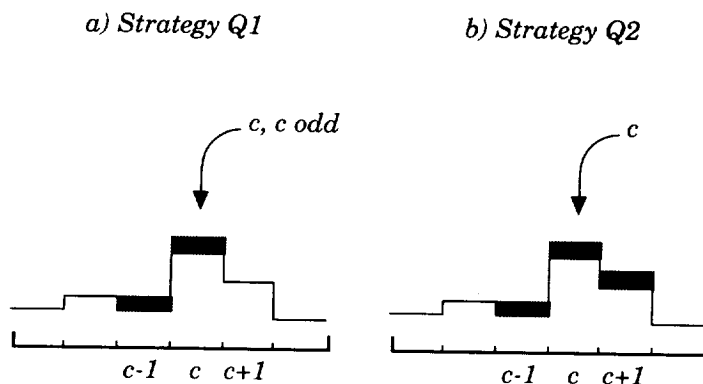


Figure 3: Entering symbols into probability distributions for 7 bit precision. a) Strategy Q1 enters two symbols, the grayvalue c having the least significant bit set to 0 and to 1, i.e. c and $c-1$ if c is odd. b) Strategy Q2 enters three symbols, i.e. $c-1$, c , and $c+1$.

5 EXPERIMENTS

Three compression experiments were carried out using the Kreska compression system. Twelve test images of different sizes were compressed with a second order Markov model since this model order generally achieves the best compression results. The test images are illustrated in the Appendix.

In a first experiment, the four escape techniques PPMA, PPMC, NELSON3, and KRESC were experimentally investigated. Table 1 shows that KRESC always achieves slightly better compression than the other three techniques. PPMC has been reported to be one of the better methods to estimate the escape probability. For the twelve test images, PPMC achieves better compression results than PPMA except for the images *bodensee* and *saudiarabia*. NELSON3 as well as KRESC estimate the escape probability more accurately and therefore obtain better compression than PPMA or PPMC. For all images, KRESC is even a more precise estimation than NELSON3 and therefore achieves the best compression results.

In a second experiment, the two context and the two symbol precision strategies were used with 7 bit and 6 bit precision for each image. Table 2 shows that context precision improves the compression results of all images except for the two images *jellybeans* and *montblanc*. Context precision strategy P2 always achieves better compression results than strategy P1. For most images, 6 bit precision yields a better result than 7 bit. Symbol precision improves the compression results of all images except for the three images *jellybeans*, *montblanc*, and *beauty*. Symbol precision strategy Q2 always achieves better compression results than strategy Q1. 6 bit symbol precision yields for 6 images the best compression results, and 7 bit precision yields the best compression results for 3 images. For most images, context and symbol precision improve the compression results.

Table 1: Compressed sizes in bytes for the 12 test images when they are compressed by a second order Markov model and different escape techniques. The best compression result for each image is shaded.

<i>Image</i>	<i>Size</i>	<i>PPMA</i>	<i>PPMC</i>	<i>NELSON3</i>	<i>KRESC</i>
jellybeans	256 ²	22675	21554	21419	21360
house	256 ²	38652	37046	36506	36446
tree	256 ²	49475	45422	44771	44605
D23	512 ²	200398	186305	182645	181435
D88	512 ²	159528	150637	148329	147836
D97	512 ²	251785	226544	219426	216184
bodensee	1024 ²	477131	480471	473976	473807
montblanc	1024 ²	500851	491367	485196	484819
saudiarabia	1024 ²	563589	565219	555316	554768
beauty	1752×2412	1100989	1097596	1088662	1088576
bottles	1812×2400	2455286	2401265	2386500	2381648
granny	1896×2304	2811870	2759957	2727910	2724362
Total	17071776	8632229	8463383	8370656	8355846

Table 2: Compressed sizes for the 12 test images when they are compressed using different context precision and symbol precision strategies. The best compression result is shaded separately for context and symbol precision. Context or symbol precision techniques do not improve the compression result for images with no shaded result.

<i>Strategy</i>	—	<i>P1</i>		<i>P2</i>		<i>Q1</i>		<i>Q2</i>	
		7	6	7	6	7	6	7	6
<i>Precision</i>	8								
jellybeans	21360	21839	24664	21415	24256	22934	26352	22503	25868
house	36446	35198	34480	34361	33869	35802	36157	34971	34986
tree	44605	43279	42808	42446	42237	43776	43747	42977	42904
D23	181435	175901	171247	172507	169012	178250	176180	175962	173716
D88	147836	143606	142181	141092	139645	145808	146603	143533	142859
D97	216184	213371	210149	211808	208373	213392	211389	212060	210979
bodensee	473807	468226	473421	463537	465135	473855	486877	467990	472296
montblanc	484819	486836	488144	488850	487766	569450	638536	603644	660643
saudiarabia	554768	539898	535939	533169	529892	547458	548735	541060	538888
beauty	1088576	1093474	1142767	1077802	1121871	1282688	1530305	1358787	1487227
bottles	2381648	2346149	2344059	2325106	2322496	2369491	2393850	2345362	2350578
granny	2724362	2674858	2651719	2652082	2636161	2697050	2689027	2676657	2669296
Total	8355846	8242635	8261578	8164175	8180713	8579954	8927758	8625506	8803588

Table 3: Image-by-image comparison of *Kreska standard* and *Kreska best precision* with other loss-less image compression systems. The best compression result for each image is shaded.

<i>System Parameter</i>	<i>uncom- pressed</i>	<i>JBIG optimal</i>	<i>JPEG optimal</i>	<i>FELICS optimal</i>	<i>Kreska</i>	
					<i>standard</i>	<i>best precision</i>
jellybeans	256 ²	20827	24187	22071	21360	21360
house	256 ²	36441	35794	34943	36446	32777
tree	256 ²	44662	45554	42416	44605	40984
D23	512 ²	177864	165669	168682	181435	163646
D88	512 ²	148157	149558	142453	147836	137341
D97	512 ²	217021	202350	207677	216184	200042
bodensee	1024 ²	521701	505476	483467	473807	462922
montblanc	1024 ²	738347	822693	809476	484819	529487
saudiarabia	1024 ²	616039	568692	567907	554768	484819
beauty	1752×2412	1177750	1453395	1314289	1088576	1088576
bottles	1812×2400	2442277	2547759	2432178	2381648	2313365
granny	1896×2304	2857214	2783696	2757259	2724362	2623662
Total	17071776	8998300	9304823	8982818	8355846	8098981

In a third experiment, context and symbol precision strategies P2 and Q2 were used together. In Table 3, *Kreska standard* indicates the compression result when neither context precision nor symbol precision are employed. *Kreska best precision* gives the compression result when optimal precision arguments are used for the context precision strategy P2 and the symbol precision strategy Q2.

For all images in the test set, *Kreska best precision* achieves better compression results than JPEG and FELICS, and the compression results are better than those of JBIG with the exception of the image *jellybeans*. Although JBIG has been designed for compression of bilevel and grayscale images with a small number of bits per pixel, its compression ratio supersedes the compression ratio achieved with loss-less JPEG for six of the 12 test images. FELICS is a very fast image compression system which achieves good compression ratios for most images. There are two exceptions namely images *D23* and *D97* where lossless JPEG obtained better compression than FELICS. Both images contain fine details which FELICS' error modelling technique does not seem to handle very well. With the exception of the image *jellybeans* which was compressed best by JBIG, *Kreska best precision* achieves the best compression results among the systems compared. For the large and very large images, *Kreska standard* already achieves better compression results than JPEG, JBIG, or FELICS. Therefore, image compression based on finite context variable order Markov models achieves good compression results for large image data. For small and medium images, additional techniques such as context and symbol precision are indispensable to a state-of-the-art compression result.

6 CONCLUSION

Variable order Markov models achieve high compression ratios for natural images provided care is taken to some subtle points during development of the compression system. An accurate escape symbol estimation technique together with an appropriate model order and context and symbol precision techniques achieve better compression results than the best known widely used image compression systems such as lossless JPEG, JBIG, or FELICS.

REFERENCES

- [Arps et Truong 94] R. B. Arps, T. K. Truong, Comparison of International Standards for Lossless Still Image Compression, *Proceedings of the IEEE*, Vol. 82, No. 6, 1994, pp. 889–899.
- [Bell et al 90] T. C. Bell, J. G. Cleary, I. H. Witten, *Text Compression*, Prentice Hall, 1990.
- [CCITT 93] –, Draft Recommendation T.82 & ISO DIS 11544: Coded Representation of Picture and Audio Information – Progressive Bi-Level Image Compression, 1993.
- [Howard 93] P. G. Howard, The Design and Analysis of Efficient Lossless Data Compression Systems, Ph.D. Thesis, Brown University, Providence, Rhode Island, 1993.
- [Howard et Vitter 93] P. G. Howard, J. S. Vitter, Fast and Efficient Lossless Image Compression, *Proceedings Data Compression Conference (DCC'93)*, Snowbird, Utah, March 1993, pp. 351–360.
- [Moffat 90] A. Moffat, Implementing the PPM Data Compression Scheme, *IEEE Transactions on Communications*, Vol. 38, No. 4, 1990, pp. 1917–1921.
- [Nelson 91] M. Nelson, *The Data Compression Book*, M&T Publishing, 1991.
- [Ohnesorge et al 93] K. W. Ohnesorge, P. Stucki, M. Bichsel, Two-Dimensional Modelling for Lossless Compression of Natural Images, *Proceedings of IEEE Workshop on Visual Signal Processing and Communications*, Melbourne, Australia, September 1993, pp. 53–56.
- [Ohnesorge 95] K. W. Ohnesorge, Modelle für die verlustfreie Bilddatenkomprimierung, Ph.D. Thesis, University of Zürich, 1995 (in progress).
- [Pennebaker et Mitchell 93] W. B. Pennebaker, J. L. Mitchell, *JPEG Still Image Data Compression Standard*, Van Nostrand Reinhold, 1993.
- [Williams 91] R. N. Williams, *Adaptive Data Compression*, Kluwer Academic Publishers, 1991.
- [Witten et al 87] I. H. Witten, R. Neal, J. G. Cleary, Arithmetic Coding for Data Compression, *Communications of the ACM*, Vol. 30, No. 6, 1987, pp. 520–541.

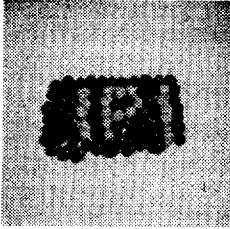
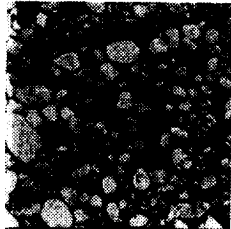


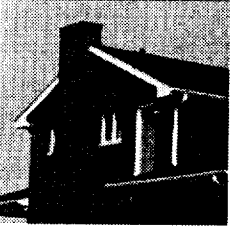
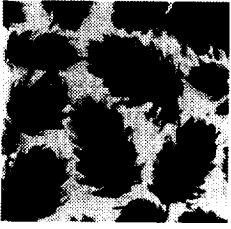
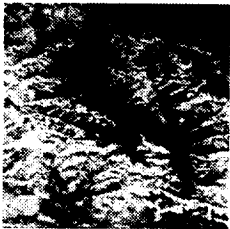


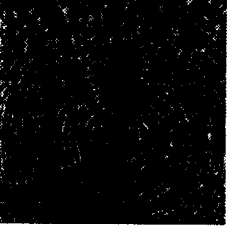


[Witten et Bell 91] I. H. Witten, T. C. Bell, The Zero Frequency Problem: Estimating the Probabilities of Novel Events in Adaptive Text Compression, *IEEE Transactions on Information Theory*, Vol. 37, No. 4, 1991, pp. 1085–1094.

[Witten et al 94] I. H. Witten, A. Moffat, T. C. Bell, *Managing Gigabytes: Compressing and Indexing Documents and Images*, Van Nostrand Reinhold, 1994.

[Ziv et Lempel 78] J. Ziv, A. Lempel, Compression of Individual Sequences via Variable-Rate Coding, *IEEE Transactions on Information Theory*, Vol. 24, No. 5, 1978, pp. 530–536.

APPENDIX

Table 4: The twelve test images. In this table, all images have been scaled to 3 cm×3 cm.

<i>small</i> 256×256	<i>medium</i> 512×512	<i>large</i> 1024×1024	<i>very large</i> ~2048×2048
			
1) jellybeans	4) D23	7) bodensee	10) beauty
			
2) house	5) D88	8) montblanc	11) bottles
			
3) tree	6) D97	9) saudiarabia	12) granny

Alternate Physical Formats for Storing Data in HDF¹

Mike Folk and Quincey Koziol

National Center for Supercomputing Applications

University of Illinois

59-82
5309
N96-15455

Abstract. Since its inception HDF has evolved to meet new demands by the scientific community to support new kinds of data and data structures, larger data sets, and larger numbers of data sets. The first generation of HDF supported simple objects and simple storage schemes. These objects were used to build more complex objects such as raster images and scientific data sets. The second generation of HDF provided alternate methods of storing data elements, making it possible to do such things as store extendible objects within HDF, to store data externally from HDF files, and support data compression effectively. As we look to the next generation of HDF, we are considering fundamental changes to HDF, including a redefinition of the basic HDF object from a simple object to a more general, higher-level scientific data object that has certain characteristics, such as dimensionality, a more general atomic number type, and attributes. These changes suggest corresponding changes to the HDF file format itself.

1. Introduction

The next several decades will bring vast increases in the amount and complexity of data generated by the U.S. Global Change Research Program, and by the EOSDIS program in particular. Some data collections call for files with thousands of small data structures, and others need to store small numbers of very large image, arrays or tables. Some will have complex, collections of interrelated data of many different types, and some will have large amounts of metadata. In addition, the patterns of access for any given set of data will vary, with some users wanting to access entire datasets, others wanting subsets of the data, and others wanting subsamples or browse images describing the data. There is no single best way to physically organize such complex mixtures of different data structures and metadata and still satisfy a broad range of access needs efficiently.

HDF has been selected as a potential standard format for storing EOSDIS data, and as such must support many of these different kinds of data collections. In working with data producers and users to store their data in HDF, it has become clear that HDF needs to provide better ways to think about and describe its data (better data models), and at the same time provide more than one way to physically organize data within a file. In this paper, we describe some of the alternative physical storage schemes that have been incorporated into HDF to support the variety of different types of data that are likely to be stored in HDF, and we look at future possible data models and physical storage schemes for HDF.

¹ The work reported on here is supported in part by a grant from NASA NRA-94-MTPE-02 "Information System Technology Applicable to EOSDIS," and Cooperative Agreement #NCC4-106 "Test Applications and Digital Library Technologies in Support of Public Access to Earth and Space Science Data."

2. Data object storage in HDF -- first generation

The basic building block of an HDF file is the primitive *data object*, which contains both data and information about the data. A data object has two parts: a 12-byte *data descriptor (dd)*, which provides basic information about the type, location, and size of the data, and a *data element*, which is the data itself (Figure 1).

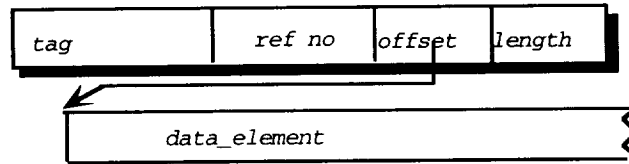


Figure 1. Data descriptor (dd) and data element.

The dd has four fields: a 16-bit *tag* describing the type of data, a 16-bit *reference number*, a 32-bit *offset* indicating the byte position in the file where the data element is stored, and a 32-bit *length* specifying the number of bytes occupied by the data element. The dds are stored together in a collection of *dd blocks*, which in turn point to the data elements stored separately in the file, each occupying a stream of contiguous bytes.

3. The need for alternate structures and methods for storing objects

This structure is simple and works well in most cases, but there are many times when different, more elaborate structures can be very beneficial in terms of efficiency and functionality. As HDF has evolved, we have attempted to extend the HDF format to support alternate methods for storing HDF objects.

Special requirements that have motivated additions to the basic HDF structure include:

Size reduction. By decreasing the size of a data file using *data compression* we can increase I/O and transmission speeds, and decrease storage costs. We can also decrease the size of a data file by removing the constraint that the number of bits in elementary data types must be a multiple of 8, that is, by supporting the storage of *n-bit data*, where n is not a multiple of 8.

Extendibility of data elements. In many applications, it is not convenient at the time of creation to specify the ultimate size of an object, or to allocate the exact amount of space in a file that will be needed to store the object. For instance, when building an array, it is desirable to be able to incrementally add "slabs" to the array. Or when building a table, it is sometime desirable to incrementally add "records" to the table. One way to support extendibility without disturbing the other structures within an HDF file is to make it possible to store a data element as a collection of *linked blocks* that can be extended whenever and object needs to grow.

Decoupling data objects from their host file. Sometimes the nature of a data object is such that it logically needs to be included with other information in an HDF file, but physically should be stored somewhere else. For instance, it might be preferable to store raw data on one device and to store everything else, including the dd, on a different device. Or it might be that a single data object, such as a table with metadata, has information to be shared among many logical

files. Support for data element storage as *external elements* makes it possible to physically separate one part of an HDF "file" from another.

Proliferation of APIs. HDF originally supported only images and multidimensional arrays. This was adequate to satisfy the needs of most early users. But over time new types of data structures, and new, more flexible APIs were introduced to satisfy HDF's expanding user community. The result was an assortment of structures and APIs that have some things in common but are just different enough that they all must be maintained. Recent investigations suggest that we can do much to unify the structures and APIs in ways that will make HDF simpler and still support the variety of applications that it now supports.

Increasing variety of types of data. EOS data sets span a large variety of types of data. Although much of it can be represented easily in HDF, there are some types of data and data structures that do not map in a natural way to the structures that HDF supports. A fresh look at the basic units and structures that make up HDF is in order.

Support for very large objects. A "very large" data object in HDF is any structure that is greater than two gigabytes. This is the case because the length and offset fields of a *dd* are represented by a 32-bit signed integer, whose largest value is approximately two gigabytes. Some HDF users need to store arrays that are much larger than this. External elements can sometimes be used to store very large data objects, but a simpler, more general solution suggests a change in the HDF file structure at its most basic level.

Support for large numbers of objects. The original design of HDF assumed that most scientific data files would contain a few large structures. With EOS data this is often not the case. It is not unusual to need to store thousands of small datasets in one HDF file, which can take inordinate amounts of time both for storage and retrieval. We are investigating changes the internal structure of an HDF file to support efficient access to large numbers of images or other types of data from HDF files.

Improving partial dataset access. Sometimes the way we access data in an array is very different from the way the data is stored in the array. For example, in the common case where the data is a very large array, often the usual row-major or column-major storage order will not perform as well as a *chunked arrangement*, where the array is divided into subarrays and stored as a sequence of subarrays. Chunked arrangements also can be of benefit in MPP environments, in which different processors independently deal with different subarrays.

The HDF file structure and supporting library has been extended to address the first three of these requirements, and the NCSA HDF group is investigating ways to address the others. In the section that follows we describe how the current version of HDF (the "second generation") addresses the first three requirements. In the next section we discuss future changes that could be made to HDF to address the other requirements.

4. Data object storage and access in HDF--second generation

In this section we look more closely at each of the alternate methods and structures that are now available for storing data in HDF. Table 1 lists the special requirements that these address, together with HDF features that have been added or will be added.

Special requirements	HDF features
Size reduction	Data compression for raster images and SDS (multidimensional arrays) Support for n-bit numbers, where $n < 32$.
Extendibility of objects	Linked block storage for special objects, such as multidimensional arrays and tables
Decoupling data objects from host file	External element storage for special objects, such as scientific data sets, images, and tables.

Table 1. Special requirements, and corresponding features in HDF.

4.1. The Extended Tag Structure

Using the original HDF data object storage scheme described above, any data element has to be stored contiguously, and all of the objects in an HDF file have to be in the same physical file. The contiguous requirement caused many problems for HDF users, especially with regard to appending to existing objects. For instance, to append data to an object that was not stored at the end of an HDF file, the entire data element had to be deleted and rewritten to the end of the file.

Beginning with HDF Version 3.2, a mechanism was added to support alternative methods of physical storage. The new mechanism is called the *extended tag*. The structure for storing extended tags involves an extra level of information, as illustrated in Figure 2. The *dd* points to an *extended tag description* record, which in turn points to the data.

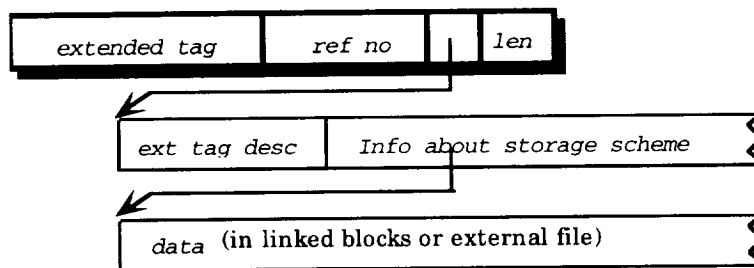


Figure 2. Extended tag structure.

Extended tags currently support three styles of alternate physical storage: linked block elements, compressed blocks and external elements.

4.2. APIs

Data producers and consumers deal with alternate physical formats through the HDF APIs. An assumption in the design of the HDF APIs is that HDF users should not have to deal with, or even be aware of, different storage formats unless they wish to influence their use in some way. For instance, a data producer might know that a certain form of data compression works best with its data, and hence might want to cause the HDF library to compress the data accordingly, while a consumer of that same data would likely not care how the data was stored, as long as it met the consumer's needs for accuracy.

On the other hand, we do not assume that data consumers would *never* need information about the physical storage of data. For example, if an image was stored using an irreversible (lossy) compression method, this information might be important in determining whether the user could carry out certain computations on the image data.

4.3. Data compression

The HDF 4.0 library and file format support optional data compression for 8-bit and 24-bit raster images and multidimensional arrays. This means that a program can, before writing data to an HDF file, specify that it is to be compressed using one of a number of compression schemes. The compressed data is appropriately identified within the HDF file, so that it can be automatically uncompressed when read back from the file.

It is possible to compress any HDF object, but the APIs that HDF users normally use support data compression only for 8-bit and 16-bit raster images, and SDS arrays.

The compression schemes currently supported in HDF were chosen because they address a range of requirements that have been identified by HDF users. Some were chosen for their speed, others for their applicability to certain classes of data, and some because their average performance is good.

Compression schemes supported in HDF 4.0 are

- run length encoding (RLE)
- JPEG (image compression)
- Adaptive Huffman
- LZRW3a

Access to compressed data elements in HDF is handled entirely by internal library routines and does not require intervention by a user after the initial compression parameters have been chosen.

For example, to compress an SDS array composed of 32-bit integers using the adaptive Huffman scheme, only the following function call is needed within an application:

```
int32 comp_type=COMP_CODE_SKPHUFF;
comp_info cinfo;

cinfo.skphuff.skp_size=DFKNTsize(DFNT_INT32);
SDsetcompress(sds_id,comp_type,&cinfo);
```

in which the compression-related parameters are:

int32 comp_type	The type of compression to use
comp_info cinfo	Additional information needed for certain compression types

Subsequent writes to the dataset would transparently compress the 32-bit integer data using an adaptive Huffman algorithm as it was being transferred to the disk file.

4.4. N-bit data

HDF 4.0 includes several new routines to let users define, write and read n-bit SDSs. N-bit data support is currently incorporated in the routine `SDsetnbitdataset`.

For example, to store an unsigned 12-bit integer that is represented unpacked in memory as an unsigned 16-bit integer, with no sign extension or bit filling and which starts at bit 14 (counting from the right with bit zero being the lowest) the following setup and call would be appropriate:

```
intn bit_len= 12;      /* n = 12 */
intn start_bit= 14;   /* Highest end of n-bit data is bit 14 */
intn sign_ext= FALSE; /* Don't use top bit to sign-extend */
intn fill_one= FALSE; /* Don't fill "background" bits with 1's */
SDsetnbitdataset(sds_id, start_bit, bit_len, sign_ext, fill_one);
```

Further writes to this dataset would transparently convert the 16-bit unsigned integers from memory into 12-bit unsigned integers stored on disk. The corresponding FORTRAN function name is `sfsnbit`.

4.5. Linked-block storage of arrays and tables

The HDF 4.0 library and file format use extended tags to support storage of certain objects in the form of a linked list of blocks, rather than the default contiguous block within the HDF file. By storing the data element of a data object as a series of linked blocks, we can append data to the object without moving it, even if it is followed within the file by other dd blocks or data elements. This mechanism can in theory be applied to any HDF data element, but the HDF 4.0 APIs only support it for SDS arrays and Vdatas (tables).

In the case of an SDS, an SDS array is appendable if one dimension is specified as "unlimited" when the array is created. Only one dimension can be specified as unlimited, and it must be either the first (C) or the last (FORTRAN). When this is done, the HDF library causes the corresponding file structure for storing the data element to be a linked-block structure rather than a contiguous block.

The Vdata case is simpler because the structure of a Vdata is more proscribed than that of an SDS. Vdatas can be appended to only by adding records, not by adding fields. Furthermore, it is assumed that the number of records is always extendible, so there is no need for a user to specify this. When a program seeks beyond the end of a Vdata, the HDF library automatically converts the file structure of the Vdata data element to a linked-block structure, unless such a structure already exists. Users are not made aware of this conversion.

4.6. External element storage

The HDF 4.0 library and file format support storage of certain objects in separate files, or "external elements," rather than the default contiguous blocks within the HDF file. Only the data element is stored externally; the dd remains in the original HDF file. External storage elements can be very useful when it is not convenient to keep

data elements in the same file as the dds.

We have found some useful applications of this feature of HDF. In one application involving the CM-5, external elements are used to write the data portions of large arrays to a scalable disk array, resulting in a 10-fold increase in I/O throughput. In this case, the external elements are move off of the disk array and inserted into the main file once computations are completed. Another application involves the external storage of arrays that might be too large to fit in one file, or on may even be too large for a file system. By using external element storage, the arrays can remain conceptually a part of one file while being stored separately.

One problem created by the use of external elements is involves describing where an external element actually resides. Users often encounter situations (e.g., disk space shortage, different filesystem names) in which the external file containing the data of the external element has to reside in a directory different from the one it was created. The user may set up symbolic pointers to forward the file locations but this does not work if the external filename is an absolute path type containing directory components that do not exist in the local system.

To solve this problem, a feature was added to HDF 4.0 that enables an application to provide a list of directories for the HDF library to search for the external file. This is set by a function call (HXsetdir) or via the environment variable \$HDFEXTDIR.

A similar feature was added to direct the HDF library to *create* the external file of a new external element in a given directory. For example, an application may want to create multiple external element files with certain naming conventions (e.g., Data950101, Data950102) while all these files share a common parent directory (project123/DataVault). This can be set by the function call HXsetcreatedir or the environment variable \$HDFEXTCREATEDIR.

5. The next generation

The changes described so far address some important requirements for HDF. Table 2 lists other requirements from our list that these changes do not address, together with some possible changes to HDF that would address these requirements.

Special requirements	HDF changes
Proliferation of APIs	New data model.
Increasing variety of types of data	New data model and new internal structure to describe objects.
Support for very large objects	New internal structure to describe objects.
Support for many datasets	New internal structure to describe object directories.
Improving partial dataset access	Alternate physical storage schemes

Table 2. Future requirements, and possible changes to HDF.

Because requirements described in Table 2 are not as easy to satisfy with the current data models and file structures of HDF, the corresponding changes call for a dramatic redefinition of HDF, both in terms of data models and of the basic structure of HDF. They are the focus of a research project called the BigHDF project, supported by the NASA AISR program, which involves a complete redesign of HDF, one that will

support backward compatibility with existing HDF structures and data models, but will address the next generation of requirements for HDF.

5.1. A new data model

HDF supports three primary data structure types (raster images, SDSs, and Vdatas), and three ancillary structure types (palettes, user-defined attributes, and annotations). A seventh structure is the Vgroup structure, which is not really a data structure but a way to group other structures explicitly. This hodgepodge of different structure types came into being over many years as HDF expanded to meet users' needs. One way to address some of the requirements listed in Table 2 would be to expand or change these structures still further, but this would inevitably increase the complexity of HDF, both for users and for developers.

Perhaps a better approach would be to recognize that in fact, many of these types share fundamental characteristics and do not really need to be separated as they currently are. In particular, the primary data structure types (raster images, SDSs, and Vdatas) can all be thought of as having the same basic characteristics, including rank and dimensionality, an atomic number type, attributes, and different physical storage options. They differ only in how these characteristics are manifested in the data models we use to describe them and in the ways that we have implemented them. For instance, raster images must always be of rank 2, SDSs can have any rank, and Vdatas must always be of rank 1. Table 3 describes how these three structures differ in their implementation of these characteristics.

structure type	rank	dimension restrictions	atomic number type	attribute support	physical storage options
raster	two	fixed size	scalar; multiple scalar components	predefined (RIS-specific); user-defined	contiguous; compression
SDS	any	fixed size, plus one unlimited dimension	scalar	predefined (SDS-specific); user-defined	contiguous; linked-block; external; compression
Vdata	one	unlimited	any scalar; multiple scalar components composite	predefined (Vdata-specific)	contiguous; linked-block

Table 3. Characteristics of HDF raster images, SDSs, and Vdatas.

In the BigHDF project, we are looking at ways to simplify and unify the data model or models that HDF supports. We are, for instance considering whether the three primary structure types (raster, SDS, Vdata) could be replaced by a single unified data model whose combined characteristics would be sufficient to describe the current structures, as well as some new ones. That is, raster images, SDSs, and Vdatas (tables) would be instances of the same model. New structures, such as a multidimensional array of records, or a set of points, might be special cases of the

same basic model. In addition, it might be useful to expand somewhat the set of allowable primitive number types that could be supported, including some sorts of pointer types for instance.

A first approximation of such a model would be to define an object type from which all, or nearly all, of the current structures would derive. Such an object would have certain essential components, such as dimensionality, and atomic-element description (like a number-type, but allowing more complex atomic elements than simple scalars), and attributes. The ancillary structure types (palettes, user-defined attributes, and annotations) would be instances of attributes.

Such a change to the data model could conceivably be supported by extending the current HDF file structure, but not without further increasing the complexity of an already overly complex structure that is hard to understand, to maintain, and to optimize for performance. Because of this we are investigating the possibility of changing the internal file structure of HDF.

5.2. A new internal structure to describe file characteristics and objects

HDF currently consists of primitive data objects comprised of dds and data elements (Figure 1). All higher level HDF structures, such as raster images and SDSs are built from combinations of these. When the higher level structures have many characteristics (rank, dimensionality, attributes, special physical storage schemes, etc.), then the combination structures that describe them can be extremely complicated, difficult to manage, and difficult to optimize for performance. The new data model described above, together with performance demands, suggests that we reexamine our assumptions about what the most primitive data object should be and how data objects should be organized. We describe here some of our ideas for a new internal structure for HDF.

Every HDF file now contains a directory of dds stored as a linked list of dd blocks, where each dd points directly to a data element, as described above. Each dd stores the size and offset of an object in 32-bit signed integers, which restricts the size and offset that an object can be to 2 gigabytes. This is one of several requirements of HDF that make it difficult to change HDF to meet new requirements. Another is the linked-list structure of dd lists, which guarantees poor performance when searching for one object among thousands of objects.

We propose that this structure might be replaced by one in which dds would be replaced by object IDs, which would point to object headers, rather than directly to data. The "object" that would be associated with an object ID would not be the very simple primitive object that HDF now supports, but one that is more like the new object type described in the preceding section. Each object header would then contain essential information that every object would share, such as dimensionality, an atomic-element description, information about the physical storage of the object, attributes of the object, and possibly a set of valid methods for the object..

To address the need to support varying sizes of files and objects, from very small to some very large, a boot block would be added that would describe such things as the sizes of offset and size values, so that large integers could be used when necessary to describe very large objects. The boot block might also describe the structure used to store the object IDs, leaving open the possibility that a different structure, such as a B-tree, could be used when large numbers of objects were involved. Figure 3 illustrates the components of this new structure.

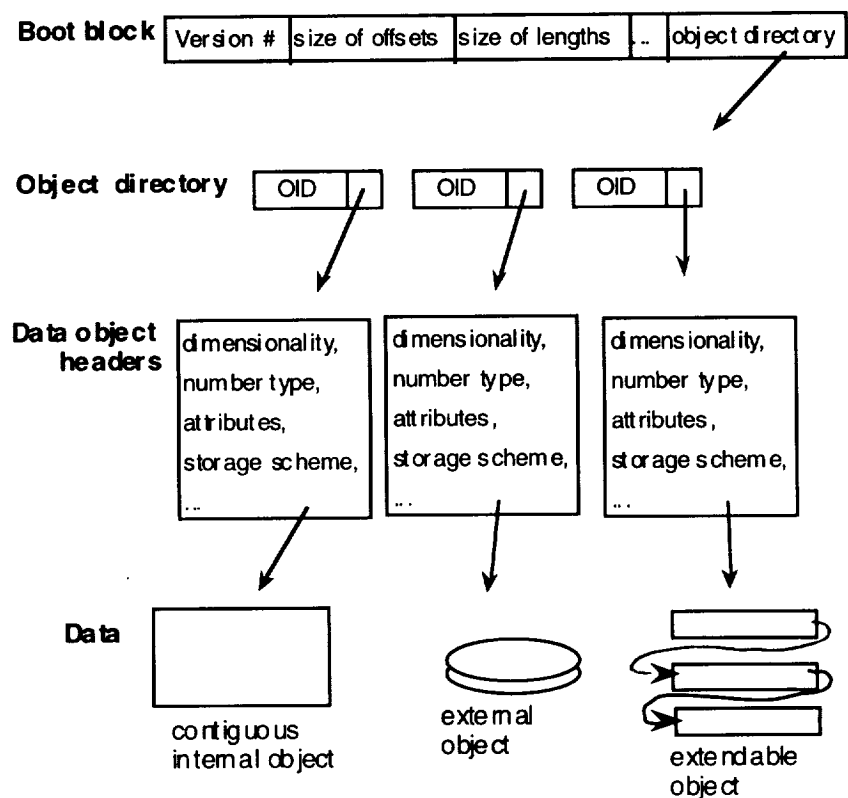


Figure 3. Possible new HDF file structure.

5.3. Alternate storage schemes

The ways that HDF now stores data elements do not address the performance problems that occur when large datasets are accessed differently by different users. HDF also does not store large arrays in ways that facilitate I/O in MPP environments. There are a number of well-known data structures, such as tiles and quad-trees, that support different modes of access better than the traditional row-major or column-major order for storing arrays. Some of these structures can be combined with data compression to provide both efficient I/O and efficient storage.

We are investigating alternative physical layouts for data elements that will improve performance when different modes of access occur. Our co-investigators in the University of Illinois Computer Science department are investigating the effects chunking on I/O performance. Chunking allows efficient assembly of subarrays in multiple dimensions from disk to main memory and resembles a common approach to distributing arrays across multiple processors in a distributed memory computer. Array chunking in memory improves performance by increasing the locality of data across multiple dimensions. This typically results in reducing the number of disk accesses that an application must perform in order to obtain the working set of the data in memory.

Over the next year we plan to implement some form of chunking as an additional alternative storage scheme for HDF.

6. Summary

HDF is evolving to meet new requirements to support EOS data. HDF has already added features aimed at reducing the size of dataset storage, being able to extend or add to data structures, and decoupling data from a host file. We need to address a number of other requirements in the future, including dealing with a proliferation of APIs, a need to support more data types, a need to support efficient storage of large objects and large numbers of objects, and a need to support partial dataset access efficiently. These future needs suggest dramatic changes in HDF, including a redefinition of the basic HDF object from a simple object to a more general, higher-level scientific data structure, and corresponding changes to the HDF file format itself.

Content-Based Retrieval of Remote Sensed Images Using a Feature-Based Approach

Asha Vellaikal^{†,‡}, Son Dao[†] and C.-C. Jay Kuo[‡]

[†]Information Sciences Lab
Hughes Research Laboratories
3011 Malibu Canyon Road
Malibu, CA 90265
{asha,son}@isl.hrl.hac.com

[‡]Signal and Image Processing Institute
Department of Electrical Engineering-Systems
University of Southern California
Los Angeles, CA 90089 - 2564
cckuo@sipi.usc.edu

510-82

5310

p-12

ABSTRACT

A feature-based representation model for content-based retrieval from a remote sensed image database is described in this work. The representation is formed by clustering spatially local pixels, and the cluster features are used to process several types of queries which are expected to occur frequently in the context of remote sensed image retrieval. Preliminary experimental results show that the feature-based representation provides a very promising tool for content-based access.

1 INTRODUCTION

Efficiency and flexibility of remote sensed image storage and management is a very important issue, since there will be a tremendous increase in the size of image data in the coming few years. To give an example, it is expected that image data with sizes exceeding terabytes everyday will be downloaded after EOS becomes operational. It is the intent of NASA to allow access to the image database to the general public [1], and it is important to allow very flexible means of retrieving the stored information to be able to make good use of available data.

Current database systems index remote sensed images based on metadata such as the time and geographical location over which the image was taken, the satellite information and so on. This does not give the user the flexibility to retrieve images based on its content, as given in a sample query by Dozier [2]:

- Find the Landsat Thematic Mapper image closest to April 1, 1992, for the Bubbs Creek area that is cloud-free and has snow. Map the snow-covered area and snow grain-size index.

The ability to access data based on its content will greatly enhance the utility of the remote sensed data sets. Apart from remote sensing, content-based retrieval of images is currently a research issue in other fields such as multimedia and telemedicine [3,4]. Most of the work has been in the area of feature-based indexing where features which can aid content-based retrieval is extracted from the images at the time of insertion. Indexing is done on these

features and queries are processed by converting them to a condition involving the features. In this paper, we extend these ideas to the remote sensing domain. In our approach, feature primitives are first extracted from the images and then the entire image is encoded in terms of these feature primitives to form a feature-based representation of the image. The feature primitives which we extract from the images are based on clustering in the multispectral space. Related work in this area has been reported by Barros et al [5].

This paper is organized as follows. Section 2 gives the background material about feature-based representation models and the concept of content in remote sensed images. Details of the feature-based representation of remote sensed images are given in Section 3 and query processing using the feature primitives is explained in Section 4. Experimental results are given in Section 5.

2 BACKGROUNDS

2.1 Feature-Based Representation Models

Traditional databases have typically dealt with data that have clear semantic meaning which makes it easier to manipulate them using query languages where conditions on each attribute can be specified exactly. Compared to such data, images are unstructured and the basic unit of information, which is the pixel, is just a measurement of intensity as recorded by a sensor. The term “content” will depend on the manner in which the image is interpreted and will differ widely depending on the application domain. For example, content-based retrieval in standard multimedia images rely on content-descriptors such as color and texture which have direct correlation with human perception [3]. Since it is practically impossible to analyze each image in response to a query, it is important to develop a representation of the image using which content-based queries can be answered. There are several characteristics which these representations should ideally possess. The representation should be able to support a wide variety of queries, should be compact so as to minimize storage space and be computationally efficient and should consist of data types for which good indexing methods exist. By using some appropriate models for an image class, the content of the images can be determined using the derived representation. This approach can be termed as feature-based representation.

Figure 1 shows the schematic of a feature-based representation data model. Here the image data are distinguished into physical images, which are the actual images and logical images which are derived from the physical images. The logical images, which are a feature-based representation of the physical images, possess all the characteristics described earlier. While the physical images can be stored in secondary or tertiary memory devices, the logical images can be stored in primary memory and all access methods can be applied to the logical image which is much more compact. Information regarding different content attributes can be derived from the logical image using knowledge bases or class models. These content attribute data can be dynamic in the sense that they can always be calculated from the logical image data and need not always be stored in the system. As opposed to content attribute data, the logical image data are static as they are a permanent representation of

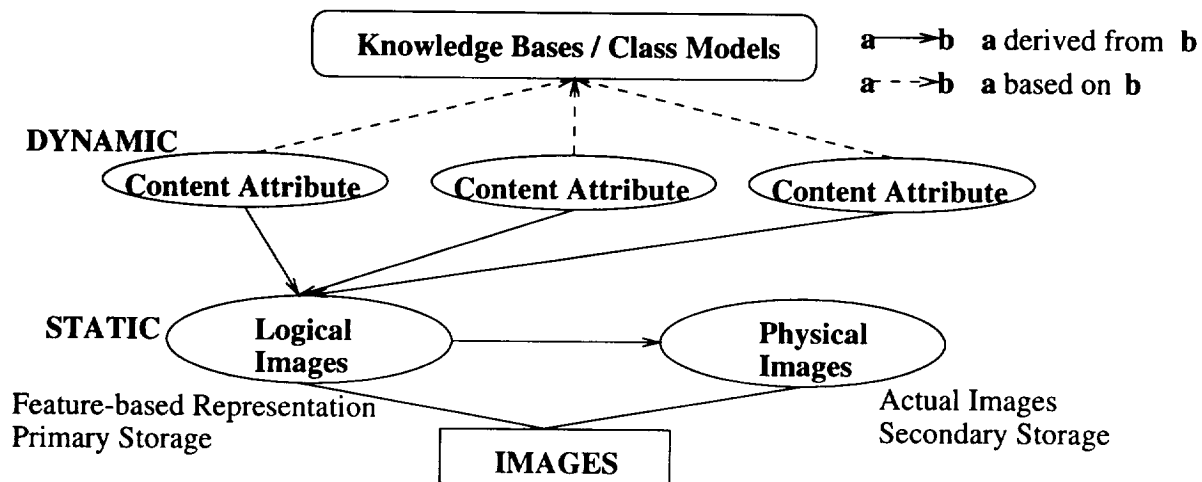


Figure 1: Schematic of a feature-based representation data model. the physical image data.

2.2 “Content” in Multispectral Images

Remote sensed images are mainly multispectral in nature with the data for the same region being gathered through different wavelength bands of interest. The measured intensity values are quantized into discrete brightness levels which determine the radiometric resolution of the image. The majority of the wavelengths utilized for earth resource sensing lie in the visible/infrared and the microwave range. These varied ranges are useful as the materials being interrogated interact differently depending on the wavelength of the electromagnetic radiation. This property is useful in distinguishing between different classes present in an image as would be explained below. In addition, the number of pixels in an image along with the spatial resolution determine the geographical area covered by an image which after georeferencing will span a definite location on earth as given by the latitude and longitude co-ordinates for each pixel. Figure 2(a) shows the four bands, wavelengths and the corresponding classified image for an NOAA Advanced Very High Resolution Radiometer(AVHRR) image which was used in our experiments. The original images had a radiometric resolution of 12 bits which were calibrated to 8 bits. The corner latitudes and longitudes (not shown) along with projection information can be used to calculate the co-ordinates of each individual pixel.

Two of the main uses of remote sensed images are as follows:

1. **Classification into ground classes:** Here the remote sensed images are analyzed so as to categorize each pixel of the image into one of the ground classes. Even though features such as texture, shape and size are used to aid classification, the dominant method is one which makes use of the fact that pixels belonging to the same class tend to cluster together in multidimensional space. For example, Figure 2(b) shows a subspace of the multispectral case using which the AVHRR image can be classified to be snow, land or cloud [6]. Quantitatively, methods such as maximum likelihood

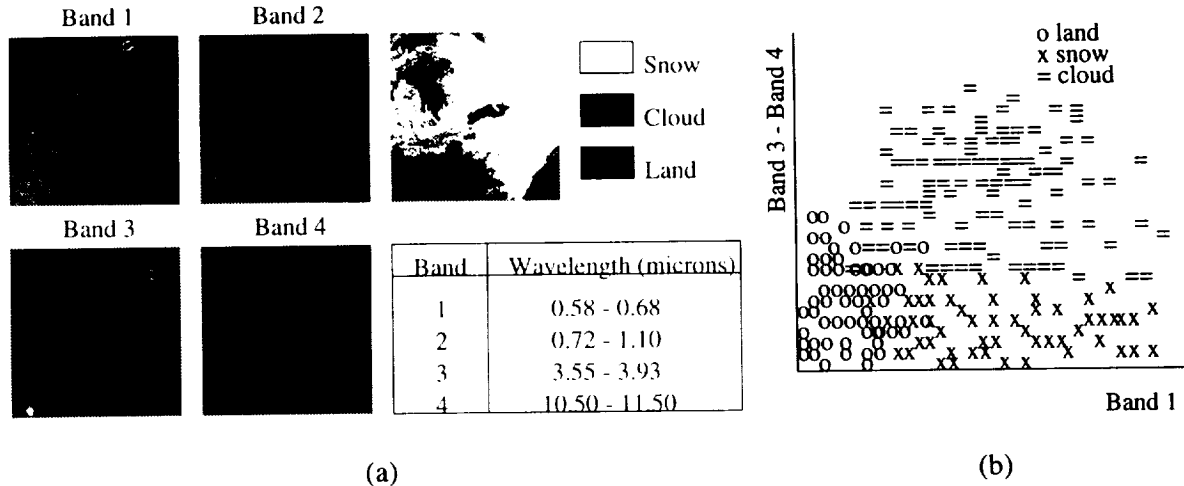


Figure 2: (a) Details of the different bands for NOAA AVHRR image, (b) classification space for land, cloud and snow.

classification can be employed if the characteristic values for the different classes are known [7]. To explain briefly, let \mathbf{x} denote a pixel vector where each component of the vector is the intensity value in a particular band. Given K classes, the maximum likelihood decision rule can be used to determine the class which \mathbf{x} belongs to as follows:

$$\mathbf{x} \in \omega_i, \text{ if } p(\omega_i|\mathbf{x}) > p(\omega_j|\mathbf{x}) \text{ for all } j \neq i,$$

where $p(\omega_i|\mathbf{x})$ is the conditional probability of class ω_i with the given vector \mathbf{x} . It is common to represent the probability distribution for each class as a multivariate normal distribution

$$p(\mathbf{x}|\omega_i) = (2\pi)^{-N/2} |\Sigma_i|^{-1/2} \exp \{-1/2 (\mathbf{x} - \mathbf{m}_i)^t \Sigma_i^{-1} (\mathbf{x} - \mathbf{m}_i)\}$$

where \mathbf{m}_i , and Σ_i are the mean vector and covariance matrix for the data in class ω_i .

2. **Calculation of Derived Products:** Another important use of remote sensed images is to calculate secondary products, also referred to as level 3 data. These products are typically calculated using band math on the different bands of an image. Some examples are products such as Sea Surface Temperature(SST) and the Vegetation Index(NDVI). The numerical values of these products are directly dependent on the intensity values of the pixels.

It is important to note that one characteristic which distinguishes “content” in remote sensed images is that the pixels are geographically registered to a physical location on the earth’s surface. Thus queries will often refer to a geographic location and typically this would only involve a small subsection of a particular image. Thus queries regarding subsections of an image would be predominant in content-based retrieval of remote sensed images as opposed to images of other kinds.

3 FEATURE-BASED REPRESENTATION OF MULTISPECTRAL DATA

There is intrinsically a lot of redundancy present in multispectral images. This is due to the fact that nearby pixels have a high probability of belonging to the same class which results in a lot of adjacent pixels having similar intensity values in all the bands. Compression methods exploit this redundancy to achieve a reduction in the image file size so as to ease storage and transmission problems. However schemes based on transform coding (e.g. JPEG, wavelets) are not useful in the compressed domain as they require to be decompressed before selected access or interpretation is possible. A technique which is naturally suited for joint feature extraction and encoding is that based on clustering, such as vector quantization (VQ) [8]. Tilton et al. [9,10] have studied the use of VQ for multispectral image compression. However the strength of VQ techniques with respect to content was not considered as the emphasis was more on compression. In this section, we will develop a feature-based representation of the multispectral image which is based on clustering of pixels in the multidimensional space.

Let $\mathbf{X} = \{\mathbf{x}_i : i = 1, 2, \dots, n\}$ be a set of n p -dimensional pixel vectors from a spatially local region of a remote sensed multispectral image. These pixels will typically tend to group together in the multispectral irradiance space as discussed above. The measures that describe the clusters which are formed are natural features to describe the intensity distributions for the multispectral image. Typical measures to represent the clusters are the cluster means, cluster variances per band, cluster covariances, number of vectors in each cluster, intercluster distances, etc. The number of clusters is dependent on the accuracy with which \mathbf{X} must be represented. It is desirable to choose the number of clusters to be larger than the number of classes which are expected to be present in that part of the image. Once the number of clusters N ($n \gg N$) is determined, \mathbf{X} is mapped onto a set of clusters $\mathbf{C} = \{C_i : i = 1, 2, \dots, N\}$ by trying to minimize some distortion measure $J(\mathbf{X}; \mathbf{C})$. Our representation of each cluster consisted of the mean, the covariance matrix and the number of pixels per cluster given as

$$C_i = C_i(\mathbf{m}, \Sigma, \phi)$$

where \mathbf{m} is the cluster mean, Σ is the cluster covariance matrix and ϕ , the number of vectors which were mapped to that particular cluster. These clusters were formed while trying to minimize a distortion measure given by

$$J(\mathbf{X}; \mathbf{C}) = \frac{1}{n} \sum_{i=1}^n \| (x_i - C_j(\mathbf{m})) \|^2$$

where x_i is mapped to cluster C_j using the *nearest mean reclassification rule* [11]. Once the clustering is complete, the index map I for that particular region can be obtained by using nearest neighbor classification in a manner similar to VQ-based compression. Note that I can be used along with \mathbf{C} to answer spatial queries.

There are two methods to encode spatial and spectral information for a multispectral image. In one case, the image is segmented to form homogeneous areas having spectral

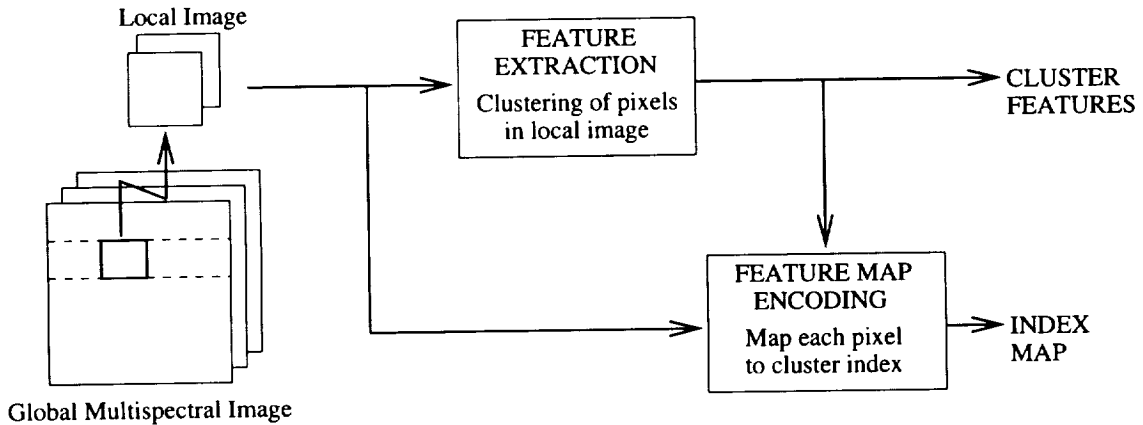


Figure 3: A feature-based representation scheme for multispectral images.

values close to each other within a given tolerance. The boundaries of these regions can be encoded along with the region features to form an approximate representation. In another case, the image can be subdivided into spatially contiguous but not necessarily homogeneous regions with feature parameters and the associated index map forming the image description. We followed the latter approach. There were several reasons in subdividing an image. First of all, clustering is computationally expensive and to cluster the entire image into thousands of clusters is impractical. Clustering efficiency can be increased by clustering each local subimage into a lower number of clusters. Such clustering is also more robust as local clusters are tighter as global clusters tend to spread out due to averaging of local distributions and perturbations due to class variance, moisture, sun angle, etc. Also, as mentioned in the previous section, queries in a remote sensing image database system will often only involve subportions of an image. Thus it is better to have developed features separately for local regions in an image. Also clustering itself does not retain any spatial information and it is more practical to have index maps I which cover only a subportion of the entire image.

Figure 3 shows our feature-based representation scheme for multispectral images. Each global image source is first divided into local sources, which are then clustered to get the cluster feature primitives. The source is then compared with the cluster feature set to form the feature index map as described above. Our representation can be summarized as follows:

$$IMAGE = \{S = (s_k : k = 1, 2, \dots, p), metadata\}$$

$$s = \{C, I, metadata\}$$

Here IMAGE is the full image which consists of various sectors S and other metadata such as sensor name, geographical location etc. Each sector s consists of the cluster set C along with the feature index map I and other related metadata.

4 QUERY PROCESSING USING MULTISPECTRAL FEATURE PRIMITIVES

The feature-based representation for multispectral images developed in Section 3 can serve as the logical image in the data model discussed in Section 2.1. Queries regarding

content can be processed by using this logical representation. There are two ways of processing queries at this point. In the first case, queries regarding certain content attributes are anticipated in advance and the logical image is analyzed apriori and information regarding that particular content attribute is stored separately. The second case consists of ad hoc queries which are different from standard queries and for which the processing need to be done directly on the logical image features at run time. The techniques for query processing in the two cases will differ as the former case can allow processing operations on the logical image which are much more computationally expensive. This paper will not consider this aspect of query processing but will limit itself to the methods by which the logical image can help query processing without too much regard to computational complexity.

Due to the lack of literature with scenarios containing well defined content-based queries, we consider the Dozier query to be a benchmark. This query references a particular satellite sensor, time, location, classes(snow,cloud) and derived data(snow grain index). From our perspective, we consider only that part of the query which refers to class and derived data as being based on “content”. In this paper, we restrict ourselves to showing how the feature representation developed in Section 3 can help answer queries regarding classes and derived data.

- **Query by Class** : Such queries refer to actual ground classes and can be regarding its presence, extent of area and so on. We shall assume that the characteristic spectral values for a particular class for a given sensor is available in the form of a mean \mathbf{m}_c and a covariance matrix Σ_c for a Gaussian distribution. Our basic premise is that the clusters with the means lying closer to a class mean will contain pixels which are very likely to belong to that class. As the clusters move farther away from the class mean, the possibility of the cluster containing pixels belonging to that class diminishes. This distance between a cluster and a class mean should be measured in terms of the variances of the class distribution. Thus we formulate a decision rule to determine whether a cluster contains pixels belonging to a certain class as follows: Cluster C contains pixels belong to a certain class if

$$(C(\mathbf{m}) - \mathbf{m}_c)^t \Sigma_c^{-1} (C(\mathbf{m}) - \mathbf{m}_c) < (m\sigma_c)^t \Sigma_c^{-1} (m\sigma_c)$$

where m is a parameter and σ_c is the standard deviation vector for that particular class. Obviously as m increases, more clusters will qualify as that particular class. Note that this decision rule only uses the mean value of the clusters and not the covariance matrix information. From a database access standpoint, the above query will be computationally complex to process unless the value for the above function is calculated and indexed earlier than the query. The above decision rule can form the basis for all types of queries regarding classes. In case of a query concerning extent of area of a class in an image, all C for that image is checked to find the ones which qualify and then the ϕ corresponding to the selected C are summed to find the area percentage. Experimental results with respect to the above decision rule are presented in Section 5. Note that the above decision rule is very simplistic and in practice a more sophisticated rule might be needed as distribution functions of several classes can overlap.

- **Query by Value:** As mentioned in Section 2.2, the numerical values of the derived products are typically dependent directly on the intensity values of the pixels. Thus queries about these products can be considered to be a query regarding the intensity values in the image. These queries can be directly processed using the feature based representation as the cluster features are directly representative of the intensity distributions in the image.
- **Data Mining:** Data mining refers to algorithms which tries to discover interesting concepts from data. Crompton and Campbell [12] discussed some important applications of data mining to remote sensing. We believe that our representation of remote sensed images can be applied to some aspects of data mining. For instance, anomaly detection is a method by which artifacts, such as outlier values from the standard statistical range, are discovered to be a manifestation of a previously unknown phenomena. The cluster representation of the multispectral image is very concise and lends itself to statistical processing. We present some preliminary results towards this end.

5 EXPERIMENTAL RESULTS

The experiments were conducted on NOAA AVHRR imagery which were available from the National Operational Hydrologic Remote Sensing Center(NOHRSC). Some details of these images were already presented in Section 2.2 and Figure 2. The images had already been georegistered and radiometrically calibrated at this stage. The difference between band 3 and band 4 was treated as a separate band and called band 5. We had 199 sets of images in our database. These images were taken over different parts of Northern America. The sizes of these images averaged around 1000×1000 pixels. The primary use of these images was for snow monitoring. The classification scheme was that suggested by Baglio and Holroyd [6] which used band 1 and band 5 as the primary bands for distinguishing the pixels into three classes, viz. snow, land and cloud. For our experiments, we developed a maximum likelihood classifier for which we estimated the characteristic values for the three classes from the classified images which were available to us. One set of characteristic values were not sufficient to provide good classification for all the images, primarily because of the differences in the mean intensities for these images. Thus we had five sets of characteristic values for each class depending on the mean intensity value of band 1 for a particular image, which gave very good classification performance compared to the original classification. These characteristic values were also used for answering queries regarding classes. Table 1 gives the mean and the variances for the different bands and classes.

For the clustering stage, each image was divided into 16 equal sectors. Clustering was then carried out on each sector separately to get 64 clusters per sector. To get a measure of the clustering performance, the mean squared error(MSE) was calculated between the original image and the final image where the final image is the result of mapping each pixel to the mean of the cluster to which it belongs to. The MSE performance for bands 1,4 and 5 are given in Figure 4(a)-(c) respectively. The average MSE was measured to be 2.82,2.08,2.22 for bands 1,4 and 5 respectively. Since MSE does not take into account the variance of each band, a better performance indicator to compare the performance of each band would be

Table 1: Mean and Variances for Different Classes.

Band	Total Mean	Total Variance	Cloud Mean	Cloud Variance	Snow Mean	Snow Variance	Land Mean	Land Variance
1	36.9	700.6	56.0	612.9	43.2	481.4	16.8	119.5
2	34.7	468.1	50.0	436.1	38.4	267.6	18.3	103.7
3	86.7	155.8	91.3	111.8	81.5	29.1	90.1	31.0
4	79.1	157.4	71.9	110.6	78.3	27.5	85.6	34.7
5	7.01	73.9	19.4	78.4	3.18	14.2	4.60	16.1

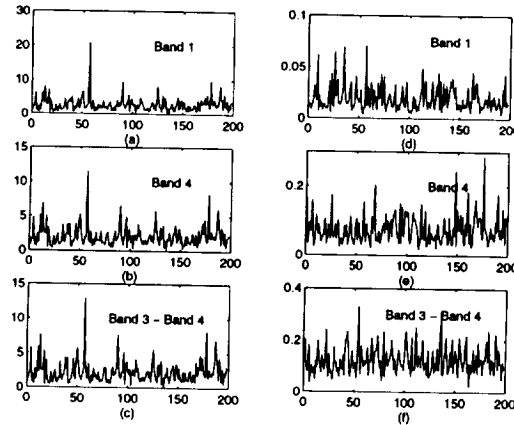


Figure 4: (a)-(c) MSE for Band 1, Band 4 and Band 5 (d)-(e) NMSE for Band 1, Band 4 and Band 5.

the normalized MSE (NMSE), which is defined for the i band as

$$NMSE_i = \frac{MSE_i}{\sigma_{ii}^2}$$

The NMSE performance for the bands 1,4 and 5 are given in Figure 4(d)-(e). The average NMSE was measured to be 0.019,0.075 and 0.122 for bands 1,4 and 5 respectively. As can be seen, the NMSE performance for the band with higher variance(band 1) is considerably better than that for the bands with lower variances(bands 4 and 5). This implies that the clustering process is dominated by bands with larger variances. We had earlier reported an approach based on clustering with different distortion measures which resulted in comparable NMSE performance for the different bands, but at the expense of higher computational complexity [13]. Improved clustering methods are certainly needed to give comparable performances between bands.

Query by Class : We experimented to see the performance of the decision rule given in Section 4. Different values of m were tried to find how the performance varies as more clusters which lie farther away from the class mean are accepted as belonging to that class. This experiment was carried out for the three different classes. The correct classification for each cluster is taken to be the class from the maximum likelihood decision rule. The performance of the classes are expected to be different as the covariances for the classes

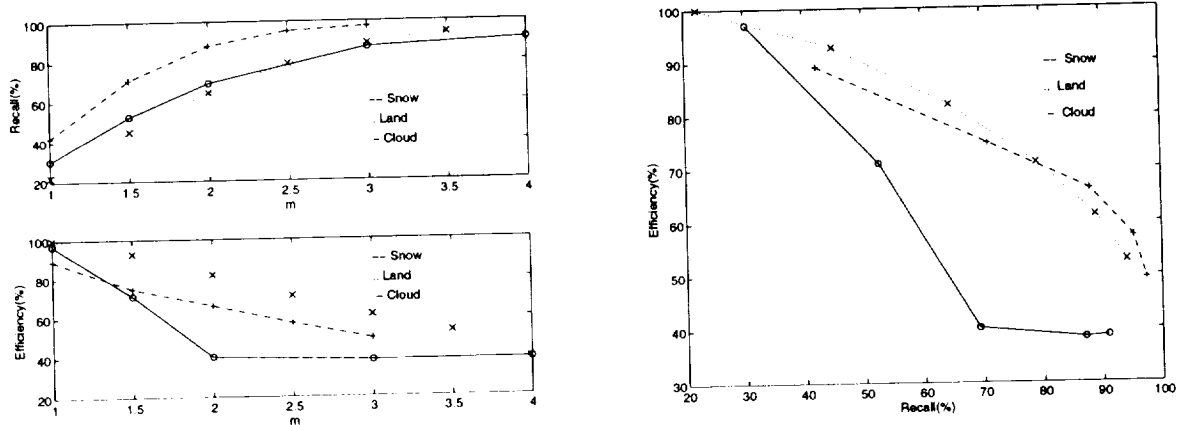


Figure 5: (a) *Recall* versus m (b) *Efficiency* versus m (c) *Recall* versus *Efficiency*.

vary considerably. The retrieval performance was judged using two measures - *recall* and *efficiency*. if A represents the number of images which were correctly retrieved, B represents the number of images which were falsely retrieved and C represents the number of relevant images which were missed, *recall* and *efficiency* are defined as

$$recall = \frac{A}{A + C}, \quad efficiency = \frac{A}{A + B}.$$

Figure 5(a)-(b) shows the performance of *recall* and *efficiency* as a function of m for the different classes. As expected for all classes, *recall* increases as m increases. This is as expected, since more clusters qualify as belonging to a certain class for larger values of m , resulting in more relevant clusters being retrieved. However this is at the cost of decreasing *efficiency* as more unwanted clusters are also retrieved. The plot of *recall* versus *efficiency* is shown in Figure 5(c). Ideally we would like *recall* to increase without much decrease in *efficiency*. Thus land gives the best results in this case. The reason for this could be the low variance values for land. In comparison, cloud performs the worst and again could be attributed to its large spread. The performance of snow was mixed. It is suggested that for improving the retrieval performance, a decision rule which also incorporates information regarding classes which lie nearby in the multispectral space be studied.

Query by Value : Here, we experimented with very simple queries which were primarily to check how well the feature-based representation could retain the original values in the image so as to support queries which will have direct bearing on the spectral values. Unlike the class queries which checked the performance of each cluster, here the queries would be regarding intensity distributions in each sector. Since we had divided each image into 16 sectors, we had 3184 sectors to experiment with. The queries were of the type : Find sectors containing pixels with *condition on band values & condition on percentages*. Table 2 gives the queries and the corresponding retrieval performance values. As can be seen, the performance of the queries is dependent on the stringency of the conditions on the bands and the percentages. As the ranges for the bands and the percentages are made narrower, *recall* performances decreases. However in general it is seen that *efficiency* performance is more stable with respect to different query conditions. It should be possible to increase the

Table 2: Retrieval Performance for Different Queries by Value.

Condition on Bands	Condition on Percentages	Recall(%)	Efficiency(%)
$Band5 \geq 11$	≥ 20	95.83	99.58
$Band5 \geq 11$	$\geq 20 \ \& \ \leq 50$	20.06	100.0
$3 \leq Band5 \leq 5$	≥ 50	10.40	98.66
$Band1 \leq 20 \ \& \ Band5 \leq 3$	≥ 50	85.76	100.0
$Band1 \leq 20 \ \& \ Band5 \leq 3$	$\geq 50 \ \& \ \leq 70$	18.32	45.21

recall performance by relaxing the conditions at the expense of *efficiency*. Of course, the performance of the queries can in general be improved by increasing the number of clusters used for representation.

Data Mining (Anomaly Detection): Since the cluster features were organized in a database management system, it was possible to statistically analyze them using standard functions such as *avg, variance* etc. A simple query was to find sectors containing pixels which differed considerably (four times) from the average values for bands 1 and 5. By crosschecking the answers from the cluster information and the actual images, we calculated a *recall* of 23% and *efficiency* of 100%. These were anomaly detections in terms of intensity values and their potential for interesting phenomena cannot be determined without help from an application domain expert.

6 CONCLUSIONS AND FUTURE WORK

We presented a feature-based representation scheme to help content-based retrievals from a remote sensed image database. The scheme which depends on clustering of spatially local pixels in multispectral space offered amongst other advantages, compactness and the ability to answer several kinds of queries which could be posed regarding content in a remote sensed image. We experimented with queries regarding classes and spectral values while also attempting some preliminary ideas in data mining. It was observed that the simple decision rule which uses only the means and covariances of a class present in a query might not have enough distinguishing capability. Also the performances of the three classes differed considerably leading us to believe that it is dependent on the variances and the amount of overlap between different distribution functions. The performance of the queries regarding spectral values were satisfactory, diminishing though with tighter ranges in the query condition. Future work will involve the processing of more sophisticated queries, using other cluster features such as covariance information along with the index map and the use of multiresolution techniques along with clustering to have a hierarchy of representations.

7 ACKNOWLEDGMENTS

The work of the first author was supported by the Howard Hughes Doctoral Fellowship. The work of the third author was supported by the National Science Foundation Presidential faculty Fellow (PFF) Award ASC-9350309. The authors would like to thank Tim Szeliga

of NOHRSC for some help with the classification algorithms and for providing part of the AVHRR images used in the presented research.

8 REFERENCES

- [1] "The Future of Remote Sensing from Space: Civilian Satellite Systems and Applications," OTA-ISC-558, U.S. Government Printing Office, Wash., D.C., July 1993.
- [2] J. Dozier, "How Sequoia 2000 Addresses Issues in Data and Information Systems for Global Change," *Sequoia Technical Report 92/14*, 1992.
- [3] W. Niblack, R. Barber, W. Equitz, M. Flickner, E. Glasman, D. Petkovic, P. Yanker, C. Faloustos, and G. Taubin "The QBIC project: Querying images by content using color, texture and shape," *Proceedings of SPIE Storage and Retrieval for Image and Video Databases*, February 1993.
- [4] M. Arya, W. Cody, C. Faloustos, J. Richardson, and A. Toga, "Qbism: Extending a dbms to support 3d medical images," *Tenth Int. Conf. on Data Engineering (ICDE)*, pp. 314-325, 1994.
- [5] J. Barros, J. French, W. Marin, P. Kelly and J. White, "Indexing Multispectral Images for Content-Based Retrieval," *Proceedings of 23rd AIPR Workshop in Image and Information Systems*, Wash., D.C., Oct. 1994.
- [6] J.V. Baglio and E.W. Holroyd, "Methods for Operational Snow Cover Area Mapping Using the Advanced Very High Resolution Radiometer," *Technical Report, National Operational Hydrologic Remote Sensing Center*, March 1989.
- [7] J.A. Richards, *Remote Sensing Digital Image Analysis, An Introduction*, Springer-Verlag, New York, 1993.
- [8] A. Gersho and R.M. Gray, *Vector Quantization and Signal Compression*, Kluwer Academic Publishers, Boston, 1992.
- [9] J.C. Tilton, D. Han and M. Manohar, "Compression Experiments with AVHRR Data," *Proceedings of the IEEE Data Compression Conference*, pp. 411-420, 1991.
- [10] M. Manohar and J.C. Tilton, "Progressive Vector Quantization of Multispectral Image Data Using a Massively Parallel SIMD Machine," *Proceedings of the IEEE Data Compression Conference*, pp. 181-190, 1992.
- [11] B.S. Everitt, *Cluster Analysis*, Edward Arnold, London, 1993.
- [12] R.F. Crompton and W.J. Campbell, "Data Mining of Multidimensional Remotely Sensed Images," *Proceedings of the Conference on Information and Knowledge Management*, Wash., D.C., pp. 471-480, 1993.
- [13] A. Vellaikal, C.-C. Kuo and Son Dao, "Content-Based Retrieval of Remote Sensed Images Using Vector Quantization," *Proceedings of SPIE Visual Information Processing IV*, Orlando, 1995.

Machine Learning for a Toolkit for Image Mining*

Richard L. Delanoy

Massachusetts Institute of Technology / Lincoln Laboratory

244 Wood Street, Lexington, MA 02173-9108

(617) 981-2545 rld@ll.mit.edu

5311
P- 8**ABSTRACT**

A prototype user environment is described that enables a user with very limited computer skills to collaborate with a computer algorithm to develop search tools (agents) that can be used for image analysis, creating metadata for tagging images, searching for images in an image database on the basis of image content, or as a component of computer vision algorithms. Agents are learned in an ongoing, two-way dialogue between the user and the algorithm. The user points to mistakes made in classification. The algorithm, in response, attempts to discover which image attributes are discriminating between objects of interest and clutter. It then builds a candidate agent and applies it to an input image, producing an "interest" image highlighting features that are consistent with the set of objects and clutter indicated by the user. The dialogue repeats until the user is satisfied. The prototype environment, called the Toolkit for Image Mining (TIM) is currently capable of learning spectral and textural patterns. Learning exhibits rapid convergence to reasonable levels of performance and, when thoroughly trained appears to be competitive in discrimination accuracy with other classification techniques.

1. INTRODUCTION

Vast image databases being accumulated today are overwhelming the capabilities of users to find particular images. For example, much of the data currently obtained from military and civilian sensors are recorded and never accessed, in part because of the difficulty of image search. And the problem is getting worse, given the fleet of imaging satellites being planned and deployed. How these data can be made available to researchers with varying levels of computer expertise and hardware resources is the subject of a continuing national debate [1,2].

Computer assistance will become increasingly necessary to facilitate such searches. Metadata (key words and computed indices used to label each image as a whole), can be used to locate images for some applications. For example, one might ask for Landsat images collected over Florida during June 1993 in which a computed cloud index was greater than some threshold. While very useful, metadata is insufficient and impractical for many other search problems, such as finding images of moveable targets in military applications, finding tropical storms in weather satellite images, prospecting for mineral deposits, locating potential archeological sites, or finding a face in images generated by an airport surveillance camera.

* The work described here has been supported by the United States Air Force. Opinions, interpretations, conclusions, and recommendations are those of the author and are not necessarily endorsed by the United States Air Force.

For these and many other applications, the only solution is to search the image contents directly.

Until recently, the only means for searching through image data on the basis of content has been customized algorithms designed to search for a particular signature. While these are often effective, such algorithms are expensive to create and maintain, suitable only for limited applications, and require a computer vision expert to build and maintain. An active area of current research and product development has been the construction of generalized database retrieval and classification tools with user-friendly environments for users with no experience in computer vision. These tools generally allow the user to provide a single example, be it a drawn shape or a box enclosing a subimage of pixels. The algorithm then attempts to characterize the example as a set of attributes (e.g., edges, spectral characteristics, eigenvalues) and compares these attributes with those computed for images in the database. A "distance" calculation comparing the attributes of the example with those of images in the database is computed and used as the criterion for selection. While capable of attention-getting demonstrations, these tools have exhibited only fair performance and have been applied only to small databases in limited domains. Examples of this class of tools are QBIC, developed by Niblack, et al. at IBM [3] and Multispec, developed by Landgrebe and Biehl at Purdue University [4]. The main shortcoming with this approach is that of using only a single example. The algorithm cannot decide which attribute values associated with that example are useful for discrimination and therefore uses them all. There is some chance that a user might get good discrimination from a single example. But, when the user is not getting the desired results, there is no way to make the agent any better other than to try a different example. A more ideal algorithm would be one capable of learning from mistakes, incorporating not only examples of the object but also counter-examples of false alarms. From these, an algorithm can learn which attributes are discriminating.

Using tools of knowledge-based signal processing developed at Lincoln Laboratory [5,6,7], a prototype Toolkit for Image Mining (TIM) has been implemented that learns which attributes or combinations of attributes are discriminating from a user simply pointing at mistakes of classification. This paper describes the intended look and feel of a proposed TIM environment, along with details of the underlying techniques of machine learning used to implement the TIM prototype. An evaluation of the technique and a discussion of possible applications is also presented.

2. THE TOOLKIT ENVIRONMENT

The premise of TIM is that a user, having found one image containing some object or region, would like to find other such images in a large database. An agent is generated in a collaborative process in which the user indicates mistakes of omission and commission to TIM and TIM responds with a visual representation of its current understanding of the users intentions. When finished, the agent can be used in a variety of ways, including autonomously searches through large databases.

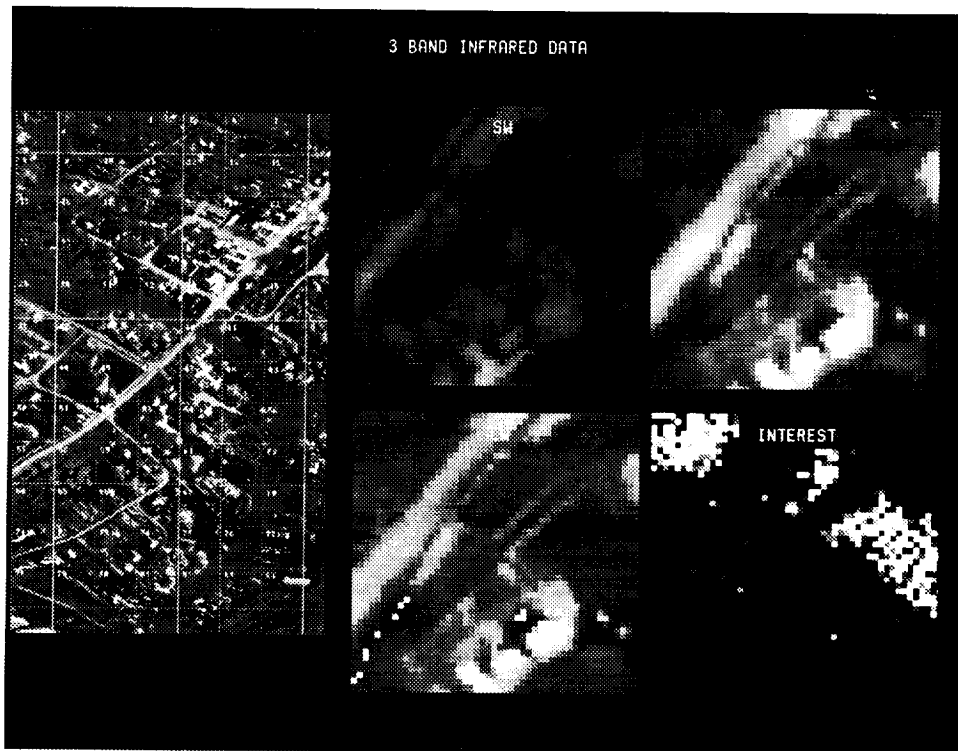


Figure 1. Example session of the TIM prototype. On the left is one of 3 infrared bands of an image of a town. Frames labelled SW, MW and LW are the short wave, medium wave and long wave bands of the subimage at tile 40 (near the center of the image on the left), which contains two bridges crossing over a river). The frame marked INTEREST is the result of applying an agent constructed for the detection of water from the set of example and counter-example pixels supplied by the user. These pixels are marked on frame LW in blue and yellow, respectively (appearing as black pixels in the river and white pixels along the highway at lower left in the gray-scale rendering shown here).

The first step of an interactive session with TIM would be to fill in information about the sensor and the data available. This information can include filenames of images to be used for training. Menus would provide options for those attributes that the algorithm is to consider as candidates for learning. For example, if the data source is multispectral, then the value of each spectral band and the value of each ratio of bands might be considered as the set of candidate attributes.

Following initialization, the main user/machine interface is a screen that would look similar to Figure 1. The screen presents an image consisting of several spectral bands (in this case long, medium and short wave infrared bands) and a feedback image called an interest image (right). With the left mouse button, the user clicks on pixels that are examples of the objects or regions to be sought. With the right mouse button, the user indicates counter-examples of objects or regions that should be ignored. With each input by the user, the algorithm builds a new template encoding the current understanding of the users intentions, applies the template to the various bands of the input image, and displays the resulting interest

image. The process repeats, using the same or other images, until the user is satisfied with the classification performance.

Once satisfied, the user can apply the agent to database searches. The user might specify existing metadata (time, place, or sensor) that might help constrain the search. In addition, the user would select from a menu some criterion for image retrieval. For example, the user might request that the agent retrieve any image for which the average interest value is greater than some threshold. The user explores the images retrieved by the agent to assess performance and to look for interesting object or region variants found by the agent. At this point, the user has the option of reentering TIM to further refine the agent.

3. MACHINE LEARNING

The core technology of TIM is a new machine learning algorithm, based on techniques of knowledge-based image processing, in particular *functional template correlation* (FTC), developed at Lincoln Laboratory. Functional templates are generalized matched filters, based on fuzzy set theory, that can be used to create customized, knowledge-based image processing operations [5,6]. FTC can most quickly be understood as a simple extension of *autocorrelation*. Where the kernel of autocorrelation consists of literal image values (essentially a subimage of the image to be probed), the kernel used in FTC is a set of indices that reference scoring functions. These scoring functions encode a mapping between image pixel values and scores to be returned for each element of the kernel. High scores are returned whenever the input image value falls within the fuzzy limits of expected values. Low scores are returned whenever the input value falls outside these limits. The set of scores returned from the scoring functions are averaged and clipped to the continuous range (0,1). As in autocorrelation, if the feature being sought can vary in orientation, then the match score is the maximum average score computed across multiple orientations of the kernel.

Scoring functions allow the encoding of ranges of acceptable values, thereby encoding uncertainty. In addition, scoring functions can be used to encode physical properties (knowledge) of the object being sought. As a result, image processing operations constructed as functional templates generally tend to be more powerful than their standard, generic analogs. Functional templates have been used in a variety of ways, including 2-D shape matching, spectral pattern matching, edge detection, thin line detection, fuzzy morphology, convolution, homotopic thinning, and data fusion. FTC has been successfully applied to several computer vision applications, including automatic target recognition [8,9] and the automatic detection of hazardous wind shears in Doppler weather radar images for the Federal Aviation Administration (FAA) [7,10,11].

In order to probe multispectral images, scoring functions are constructed for each spectral band. In this case, each functional template consists of a series of kernels, each having a single element with an associated, unique scoring function. Transformations of multiple bands into a single value, such as the ratio of two bands, might also be considered as inputs and likewise have a scoring function. In addition, texture can be decomposed into primitives (e.g., local variance, fractal dimension, contrast, periodicity, orientation) that can be encoded as pixel values representing a local neighborhood. The scoring functions for each spectral band, spectral transformations, and textural transformations would all be applied in tandem, producing a single average (i.e., interest value) for each pixel of the output interest image.

Given this structure for spectral and textural functional template correlation, functional template learning is implemented by constructing scoring functions for each band value or computed attribute. Briefly, the approach we have taken is based on the construction of distribution histograms, one for example and another for counter-example values of each attribute. The scoring function is a mapping of the degree of overlap between examples and counter-examples at each bin of the two histograms. For example, high scores are returned for input values that have many examples and few or no counter-examples. Low scores are returned for values that have many counter-examples and few or no examples.

The N most discriminating scoring functions are collected by iteratively building test functional templates that are applied to the set of examples and counter-examples. First, the single scoring function is found that does the best job in discriminating between examples and counter-examples. Then, the second scoring function is found that, when combined with the first scoring function in a single functional template, achieves the best discrimination. Additional scoring functions are added until either N scoring functions have been collected or until the addition of any other scoring function fails to achieve an incremental gain in performance.

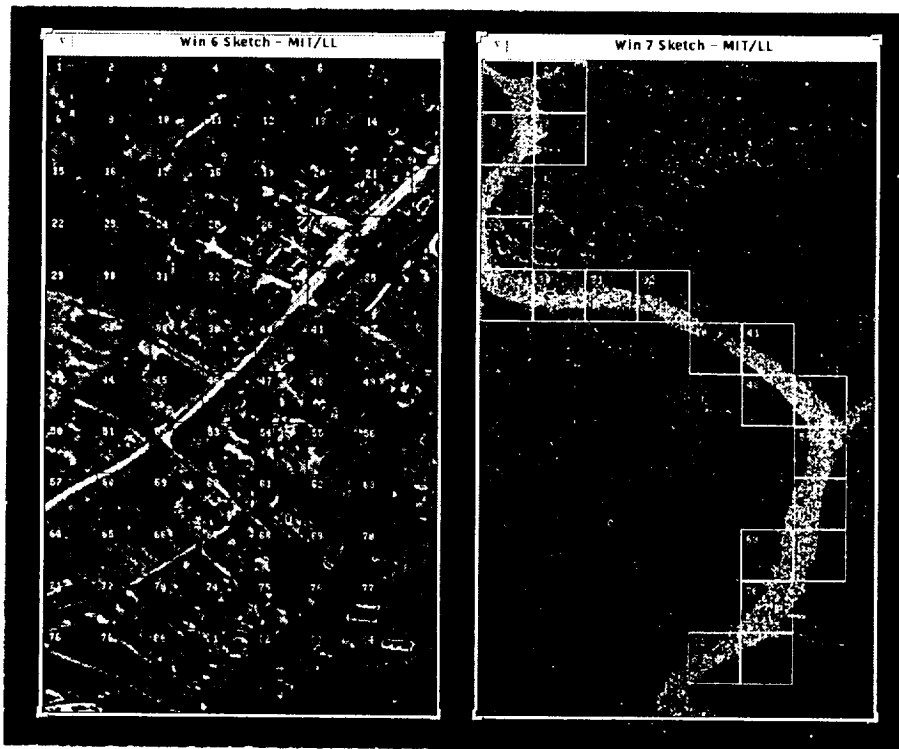


Figure 2. Input image and interest image with selected subimages resulting from the application of the functional template taught to recognize water.

4. EXAMPLES AND EVALUATION

Agent training and use are illustrated with two different examples. In the first, 3-band (long wave, medium wave, and short wave) infrared images of urban and suburban scenes

were used. Images were divided into 50 by 50 pixel tiles and trained on one tile at a time. Figure 1 shows the end of training an agent (functional template) to detect water. Figure 2 shows the application of this agent to the overall image, resulting in the interest image on the right. The white boxes enclose tiles that have average interest values exceeding a selected criterion threshold. In the same way that an agent might select tiles from a large image, the agent might use the agent with a selection criterion to select images from an image database.

A more difficult task is the classification of vegetation types from Landsat Thematic Mapper (TM) images. Huber and Casler [13] have described work in which they evaluated a variety of simple classification metrics and formulas developed by wildlife biologists. Their area of study, a region in central Colorado, was classified pixel-by-pixel into 14 vegetation types. Actual ground data were collected and used for scoring. They concluded that "using remote sensing information in complex terrains can be a difficult and relatively inaccurate process". With no technique did the percentage of correct classifications exceed 50%. When they added digital elevation data to determine the direction in which terrain was facing, they reached nearly 70% detection, although with substantial numbers of false alarms.

Using the same TM data, Augusteijn et al. [14] used a Cascade-Correlation neural net (an enhanced version of back-propagation) [15] to classify homogenous regions on the basis of spectrum and texture. The data chosen by them for training and testing consisted of 8X8 pixel boxes containing only one vegetation type. Of the 14 vegetation types, only 9 were available in boxes of this size. The task was to train the neural net on half of these boxes and then attempt to identify vegetation types in the remaining boxes. Using this approach, they were able to achieve 99% correct classifications using only spectral data.

The same TM image was used to perform a preliminary evaluation of functional template learning. Using the TIM environment, agents were trained to identify pixel-by-pixel each of the vegetation types in 100 X 100 pixel subimages. Convergence to a solution was typically rapid, with reasonable detection performance (e.g., 80% detection with 5% false alarms) in as little as 50 pixel inputs. The average classification accuracy for 13 thoroughly trained agents (1 vegetation type was not present in the training image) was 94% correct detections with 6% false alarms on the training image. The templates were then applied to adjacent images, the results were scored, and the process of indicating classification errors continued. After several images were used in this fashion, the agent reached an asymptote average performance of 90% correct detections and 6% false alarms on subimages not every used for training.

These results are clearly better than those reported by Huber and Casler, even when they incorporated topographic data in addition to the 6 TM bands. It is more difficult to compare our results with those presented by Augusteijn et al. Where the agents generated using TIM were forced to classify every pixel, the previous study attempted only to classify homogenous 8X8 pixel patches. Many, if not most, of the classification mistakes made by TIM were in the boundaries between regions. These boundaries often have transitional, ambiguous spectral characteristics that are likely to be misclassified by the human truthers as well as by TIM.

5. DISCUSSION

We have designed a way for a user and a machine learning algorithm to collaborate in the training of agents to recognize particular spectral or textural patterns. Being provided with a set of examples of the intended pattern and counter-examples of patterns to be ignored, the learning algorithm creates a functional template of the intended pattern. The trained agent would then be sent into a large database to autonomously search for other images containing the same pattern. As mistakes are made, the user can continue to refine the agent by adding more examples and counter-examples. Because functional templates have a simple, readily interpretable representation, it is possible to extract how each agent discriminates and to edit learned agents to achieve predictable changes in performance. It is believed that these techniques can be extended to the learning of shapes.

Customized search algorithms dedicated for a single application may well outperform agents generated using TIM. However, this approach is expensive and therefore impractical for most potential users of image databases. The interactive training of agents using the TIM environment allow anyone with an understanding of a particular image domain to build reasonably good search and classification tools.

Agents generated by TIM might be used for several different applications other than content-based image retrieval, including (1.) discovering equations for indices that can be used as metadata to tag database images, (2.) discovering which spectral bands are necessary for a particular task when bandwidth limitations prevent the transmission of all data collected, (3.) prioritization of data (e.g., should data be stored on primary, secondary, or tertiary storage devices) on the basis of image content, (4.) building components for more complex search algorithms, and (5.) quantitative analysis and trend analysis for scientists and policy planners.

It is likely that a tool such as TIM, by enabling users with limited computer experience to inexpensively build useful computer vision tools, should facilitate the exploration and use of image databases.

REFERENCES

1. NASA's EOSDIS development approach is risky. U. S. General Accounting Office Report GAO/IMTEC-92-24, February 1992.
2. Broader involvement of the EOSDIS community is needed. U. S. General Accounting Office Report GAO/IMTEC-92-40, May 1992.
3. W. Niblack, R. Barber, W. Equitz, M. Flickner, E. Glasman, D. Petkovic, P. Yanker, C. Faloutsos, and G. Taubin. The QBIC project: Querying images by content using color, texture, and shape. *Computer Science*, February 1993.
4. D. Landgrebe and L. Biehl. *An Introduction to MultiSpec*. School of Electrical Engineering, Purdue University, 2.15.94 edition, February 1994.
5. R. L. Delanoy, J. G. Verly, and D. E. Dudgeon. Functional templates and their application to 3-D object recognition. In *Proc. of the Intl. Conf. on Acoustics, Speech, and Signal Processing*, San Francisco, CA, March 1992.

6. R. L. Delanoy and J. G. Verly. Computer apparatus and method for fuzzy template matching using a scoring function. U. S. Patent No. 5,222,155, June 1993.
7. Richard L. Delanoy and Seth W. Troxel. Machine intelligent gust front detection. *Lincoln Laboratory Journal*, 6(1):187–212, Spring 1993.
8. R. L. Delanoy, J. G. Verly, and D. E. Dudgeon. Machine intelligent automatic recognition of critical mobile targets in laser radar imagery. *Lincoln Laboratory Journal*, 6(1):161–186, Spring 1993.
9. J. G. Verly, R. L. Delanoy, and C. H. Lazott. Principles and evaluation of an automatic target recognition system for synthetic aperture radar imagery based on the use of functional templates. *Automatic Target Recognition III, Proceedings of SPIE*, Vol. 1960, Orlando, FL, 14–16 April 1993, pp. 57–71.
10. Richard L. Delanoy and Seth W. Troxel. Automated gust front detection using knowledge-based signal processing. In *IEEE 1993 National Radar Conference*, Boston, MA, April 1993.
11. M. M. Wolfson, R. L. Delanoy, M. C. Liepins, B. E. Forman, and R. G. Hallowell. The ITWS microburst prediction algorithm. *Preprints 5th Conference on Aviation Weather Systems*, Vienna, VA, August, 1993.
12. R. L. Delanoy and R. J. Sasiela. Machine Learning for a Toolkit for Image Mining. Technical Report 1017, MIT Lincoln Laboratory, March 1995.
13. H. P. Huber and K. E. Casler. Initial analysis of Landsat TM data for elk habitat mapping. *Int. J. Remote Sensing*, 11(5):907–912, 1990.
14. M. F. Augusteijn, L. W. Clemens, and K. A. Shaw. Performance evaluation of texture measurements for ground cover identification in satellite images by means of a neural network classifier. Technical Report EAS-CS-93-8, University of Colorado, Colorado Springs, CO, October 1993.
15. S. E. Fahlman and C. Lebiere. The cascade-correlation learning architecture. In D. Touretzky, editor, *Advances in Neural Information Processing Systems 2*. Morgan Kaufmann, San Mateo, 1990.

Dissemination of compressed satellite imagery
within the Navy SPAWAR Central Site Product Display environment*

Oleg Kiselyov and Paul Fisher
Computer and Information Sciences, Inc.
303 N. Carroll, Suite 108
Denton TX 76201

Phone: (817) 484-1165, FAX: (817) 484-0586
Email: oleg@ponder.csci.unt.edu, oleg@unt.edu
http://replicant.csci.unt.edu/~oleg/SIMDC95/Paper_Summary.html

5312
P 8

Abstract

The paper presents a case study of integration of compression techniques within a satellite image communication component of an actual tactical weather information dissemination system. The paper describes history and requirements of the project, and discusses the information flow, request/reply protocols, error handling, and, especially, system integration issues: specification of compression parameters and the place and time for compressor/decompressor plug-ins. A case for a non-uniform compression of satellite imagery is presented, and its implementation in the current system is demonstrated. The paper gives special attention to challenges of moving the system towards the use of standard, non-proprietary protocols (smtp and http) and new technologies (OpenDoc), and reports the ongoing work in this direction.

I. History of the project

Central Site Product Display (CSPD) system has been developed for the Space and Naval Warfare Systems Command (SPAWAR) for disseminating a whole range of weather and oceanographic information to a variety of clients afloat or ashore [1]. The clients (not necessarily related to the Navy) can request/receive data via serial (dial-up), Internet and VSAT connections. The available information includes pressures, temperatures and wind directions on and above land/ocean surfaces, barometric heights, temperatures on the ocean surface and at specific depths, precipitation, primary observational data (weather reports), storm warnings, and a number of satellite images of different kinds (IR, Visual, DMSP). The satellite images are usually 512x512 or 1024x1024 8-bit graylevel pictures.

CSPD is being developed upon the foundation of a very popular software system NODDS built by the Fleet Numerical and Meteorology Center (FNMOC) at Monterey, CA. NODDS has been in operation for more than 4 years. The server part of the system runs on a Cyber mainframe. It was written in FORTRAN with modules dating as far back as 1987. The client portion of NODDS was written in QuickBASIC for PC-DOS. The hardware base of the old system is obviously in need of updating, so is the software. The new CSPD server has to work on modern-day workstations (HP 9000, Sun SPARC, Concurrent's MAXION RISCstation) while the clients are to run on almost any platform: UNIX workstations, Macintosh and PC computers. Furthermore, there is a clear need to expand the system to include more, specialized weather products, to enhance display of received information, and decrease communication time. Improving satellite imagery communication is of special priority, especially as far as flexibility of requesting and compressing images is concerned. The new system should allow the user to browse and request all available satellite imagery (rather than a few pre-selected pictures), to choose a compression algorithm and its parameters, and evaluate the anticipated image degradation when a

* This work supported in part by US Navy SPAWAR Grant N00039-94-C-0013 "Compression of Geophysical Data" and by US Army Research Office TN 93-461 administered by Battelle Research Office.

lossy compression has been selected. The user should also be able to specify a non-uniform compression of certain areas/features that he considers particularly important. Again, *the rationale for using compression is to reduce the communication time.*

Computer and Information Sciences, Inc. (CIS) has been working on this project as one of the contractors. CIS is solely responsible for handling of satellite imagery within CSPD. We have written the software to compress/decompress satellite images using technology developed at CIS, as well as code to convert satellite imagery from DEF to TIFF formats, to view images under CSPD and standalone, and to catalog imagery. We have also completely re-written the server part of CSPD in C++, and perform its maintenance and enhancement.

II. Request-delivery information flow and protocols

The current CSPD system is client-server, though there are capabilities for a broadcast-type dissemination. A NODDS server distributes weather and satellite information in response to queries from CSPD clients. The server uses NODDS/TEDS databases (managed by *empress* database manager) as well as a collection of flat files to look up the requested data. The net functionality of the server is as follows: reading a NODDS input (request) string, retrieving grid data from a TEDS database, encoding them using Navy encoding routines (*rainform-gold* or *OTH-gold*), and transmitting the encoded data to the requester. Text files, synoptic and upper air reports, and satellite image requests are handled as well.

To formulate a request, a user has to select a geographical area he would like to get weather information for, and pick the products he is interested in for the selected area. Satellite imagery is treated as a regular 2D product. Request for a satellite image *always* carries in it specifications for a compression method to use, and its parameters. Originally they are set to some "default" compression specifications, which can be evaluated and changed by the user using the compression manager (see below).

Whenever the server comes across a request for a satellite image, it queries the database to see if an image of the corresponding type (IR, visual, etc.) covering a specified area (and valid for a specific period of time) is present in the database. If the image exists, the server obtains the name of the corresponding TIFF file and launches a compressor plug-in to compress the file according to the parameters specified in the request. If the compression was successful, the resulting binary code is included in the return packet. Otherwise the server sends a message detailing the error (as a matter of fact, the entire contents of `stderr` produced by the compressor).

When the client receives the reply, it unpacks and preprocesses it. At this time, the client runs a decompressor plug-in, which decodes the received binary code into a PGM file. The latter is used by a display module to show the picture, and, possibly, lay temperature, pressure, etc., isoplots and other information over the satellite image.

It should be stressed that the compressor and decompressor are separately compiled and run executables (plug-ins), which are launched (spawned) whenever there is a need to encode/decode image data. The modules are available for a variety of platforms, and rather versatile: they support very simple and unified interface. The fact that request/reply parsing and compression/decompression are performed by separate executables has several advantages. Indeed, one can modify the compression module (say, by adding more sophisticated compression techniques) without any need to recompile/relink client or server executables. Moreover, the compressor can be run by programs other than the CSPD server, as the functionality the plug-in provides - image compression - can be used in some other contexts, for example, within WWW environments.

Another important advantage of having compression/decompression modules as separately run plug-ins is a better error handling and improved robustness to error conditions. The root NODDS server module can only detect very simple errors, mainly related to the syntax of a request

string. Most semantic errors, for example, when a requested image file cannot be found or read, there is no space on device for the encoded file, when the specified compression parameters do not make sense, etc., – are detected when actually performing the compression. When the server root module launches the compressor through the system() function, it redirects the compressor's stdout/stderr into a temporary file. If the compressor succeeded, it writes "OK" into a special file and creates a file with the compressed image data. The latter is included by the NODDS server in its output stream, PACKED.DAT (following appropriate headers). UU-, Base64-, etc., encoding can also be performed if necessary at this stage. If the compressor failed, its standard error stream (which explains in detail compressor's actions and the reasons of failure) is inserted into the PACKED.DAT as a special message to a client. In any event, a problem with locating/compressing of an image would never cause the NODDS server to crash, and the user would always get some answer to his query.

As was mentioned before, we have operational compressor/decompressor plug-ins for a variety of platforms: Sun Sparc, HP 9000/7xx, Macintosh. The modules are rather versatile and independent: they can be run as "filters" that translate their input into output (that is, read a TIFF/PGM/XWD file and make a compressed output, or read the compressed image code and prepare a TIFF/PGM/XWD file). This makes it straightforward to integrate the plug-ins into various environments.

For example, recently we have cast the decompressor modules as Web browser helpers, with the help of a very simple shell script. This made it possible to view the (compressed) satellite images retrieved from the NODDS/TEDS databases through Mosaic/Netscape. That is, clicking/linking on an item of type image/x-sowlp-gray (extension .slpg) causes the compressed image be delivered, unpacked, and displayed. From the user's point of view, it looks no different than retrieving a JPEG picture. It is even possible to play with different compression ratios: one merely needs to type the value of a quality parameter on a specified slot of web page [2] and click on the "Compress Now" button. Having received the request, an HTTP server compresses the image and sends the compression ratio, compression time, and the compressed data back to the client. The user thus can note how long it took to communicate the image, and see how good the decompressed picture turns out to be. This interactive compression demo is available from [2]; please contact the authors for decompression helpers.

Since http and smtp both use the same MIME protocol to deal with non-textual data, the decompressor modules-helpers can be used just as they are to view encoded images sent through e-mail, that is, asynchronously. The same viewer that helps out Mosaic when image/x-sowlp-gray stream comes in can be used to make sense of an e-mail message with a image/x-sowlp-gray "attachment". The mailers take care of ASCII encoding if necessary, and they do it transparently and automatically. Using electronic mail to send out satellite imagery has a tremendous advantage: one is no longer limited to TCP/IP or any other particular communication protocol or media, one can send a mail from a FIDONET site to a mainframe, from a PC on a Novell network to a Mac, etc. One can also take the full advantage of all e-mail frills and embellishments, like mailing lists, automatic broadcasting, mail aliases, mail exchangers, etc.

III. The compression manager

The NODDS server always compresses satellite images for transmission. The compression method and its parameters are taken from a satellite product request string received from a client. When the user first selects a satellite product in formulating a request, the corresponding request string receives some "default" compression specifications. It is the compression manager that lets the user customize the default settings. Thus the place of the compression manager can be illustrated by the following chart on Figure 1.

The compression manager allows interactive selection of a compression method and its parameters. Note, the compression is almost always lossy. To give the user an idea how well the delivered

image would look like and whether important features the user is looking for would be preserved, the compression manager loads up some (relatively small) sample image and displays it along with the compressed and decompressed picture, using the parameters the user has set up. Figure 2 shows a snapshot of the component manager screen. The new version of the compression manager supports non-uniform compression as well, see below.

At present, the compression manager implements a few variants of a modified Laplacian pyramid compression method (multi-resolutional analysis with overcomplete frames) discussed in papers presented at Data Compression Conferences [3], [4]. See also [5] for a complete list of references.

IV. Non-uniform compression

The very nature of environmental images, or any image for that matter, suggests that not every detail of the picture is equally important to the observer. For example, the area of the hurricane eye on a satellite image should be of high resolution, while the tight cloud cover of the hurricane body is less informative and may be rendered with a lower resolution, though it cannot be completely discarded. In disseminating weather information over a fleet, a meteorologist at a particular ship needs very accurate data on the cloud cover, wind direction, temperature, etc., just in the vicinity of his ship. The information about what is going on outside that small area is used for prognosis and does not have to be of very high precision. Accordingly, the amount of loss and inaccuracy that can be tolerated during the communication varies not only from one user to another but also from one region of the image to another. This raises the problem of a non-uniform, lossy compression, i.e., compression where the loss varies with the location/features/frequencies, etc., and tailoring such compression to a particular user and circumstances. Preserving the information during compression to the extent the user needs, but not more, helps to drastically reduce the amount of data that has to be transmitted.

In the present implementation of non-uniform compression [3], the regions of special interest are specified by assigning to them a degree of relative importance, using an “alpha-channel”, or a “mask”. The mask is an 8-bit image of the size of the original one with pixel values telling a relative weight of the corresponding area: the bigger the weight, the higher the importance. Note that the mask is necessary only for compression; a non-uniformly compressed image code has the same format and structure as a uniformly compressed one. That is, non-uniform compression does not require a special decompression; the same decoder restores both kinds of compressed imagery.

To simplify transmission of the mask in a request to the server, we, at present, have limited specification of the importance area to a single rectangular region. Therefore, only a few extra characters in the request string are necessary to solicit the non-uniform compression. For example, a specification

[10,(1,2)-(127,255)]

means that a rectangular area of an image with the upper left corner at (1,2) and the lower right corner at (127,255) is to be assigned a weight of 10 during the compression. The rest of the image (outside this rectangle) is assigned weight 1 by default. Thus, features within the rectangular area would incur 10 times less distortion than features outside the rectangle. Note, x_{min} , x_{max} , y_{min} , and y_{max} are specified in pixels of the target image: therefore, one has to have some idea of the size of the image to compress before specifying the non-uniform compression parameters.

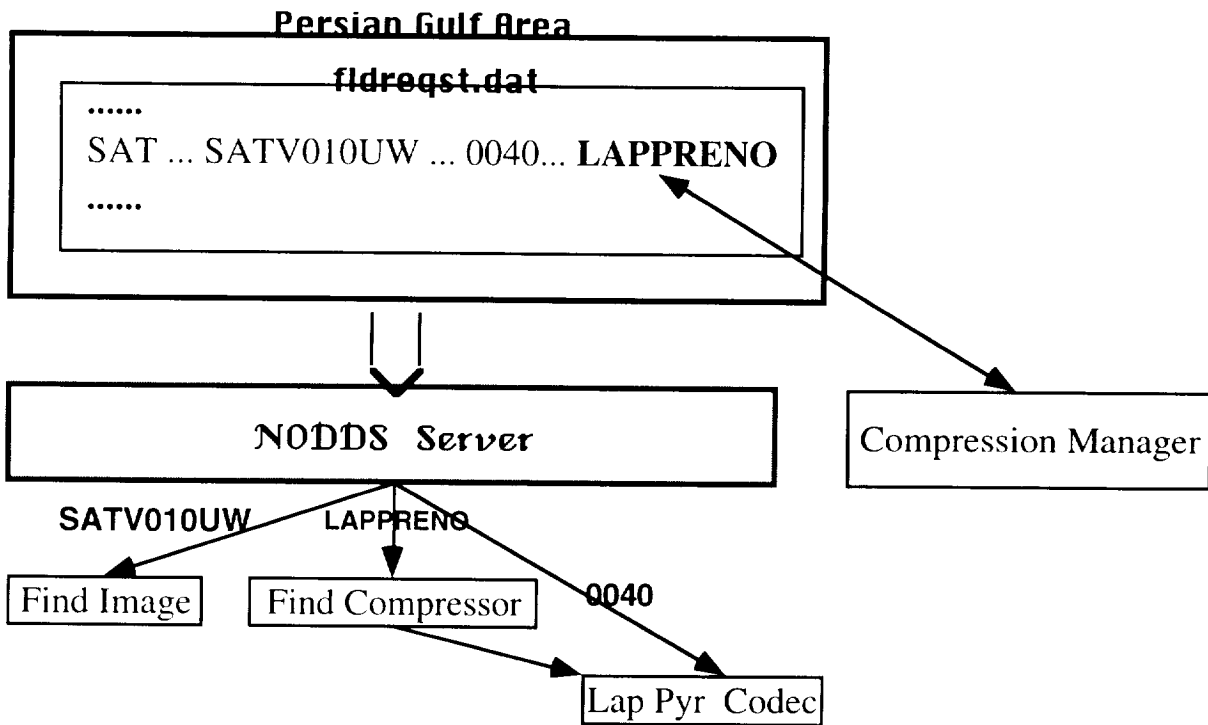


Fig. 1. Compression Manager's place in the overall information flow

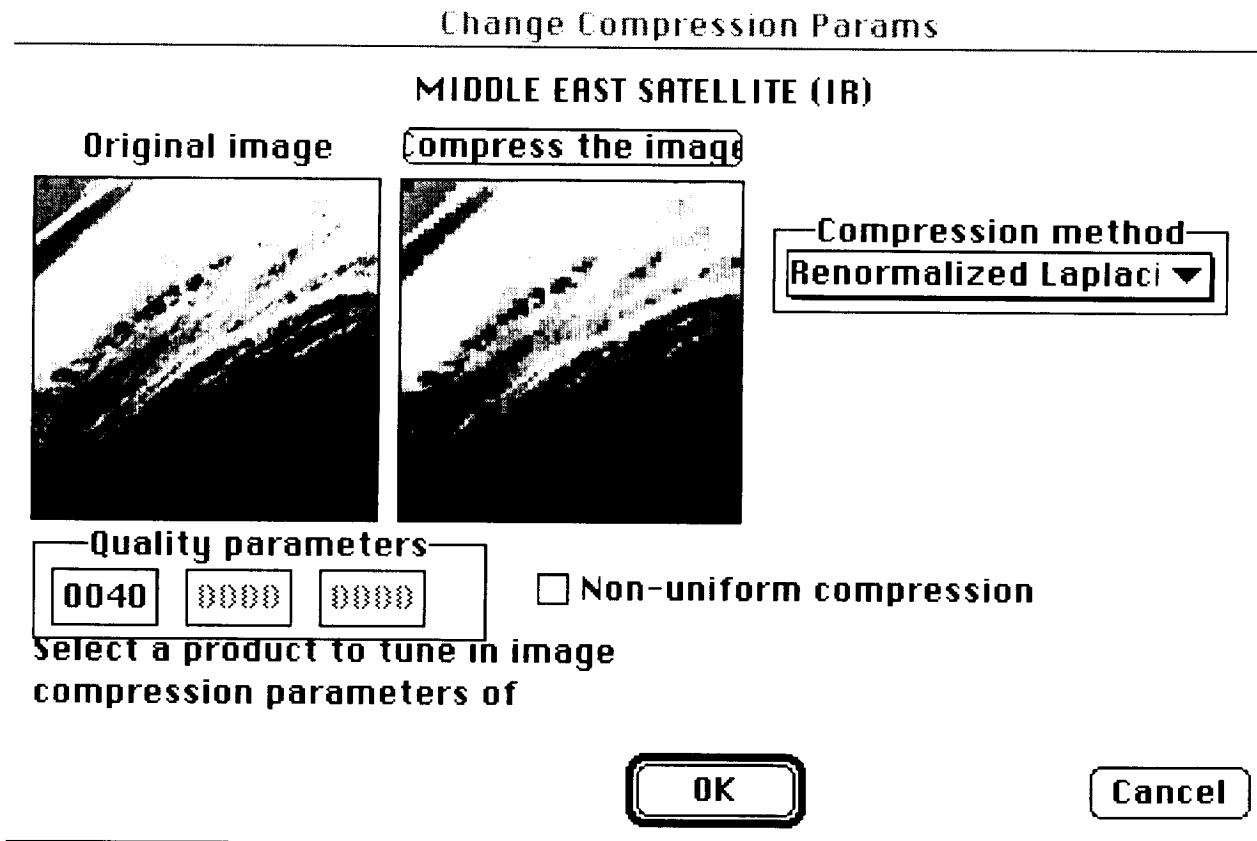


Fig. 2. Snapshot of the Component Manager screen

Note, that the region of low distortion is blended smoothly into the somewhat more distorted “background”, without any sharp and conspicuous boundaries between the areas of different importance. Thus the picture restored after decompression does not have a “cut-and-paste”, or “patched” look at all. These smooth transitions are a notable feature of the present non-uniform compression method.

The compression manager discussed above can also be used to select a region of special interest, assign a degree of importance to it, and preview the effect of the compression on a sample image:

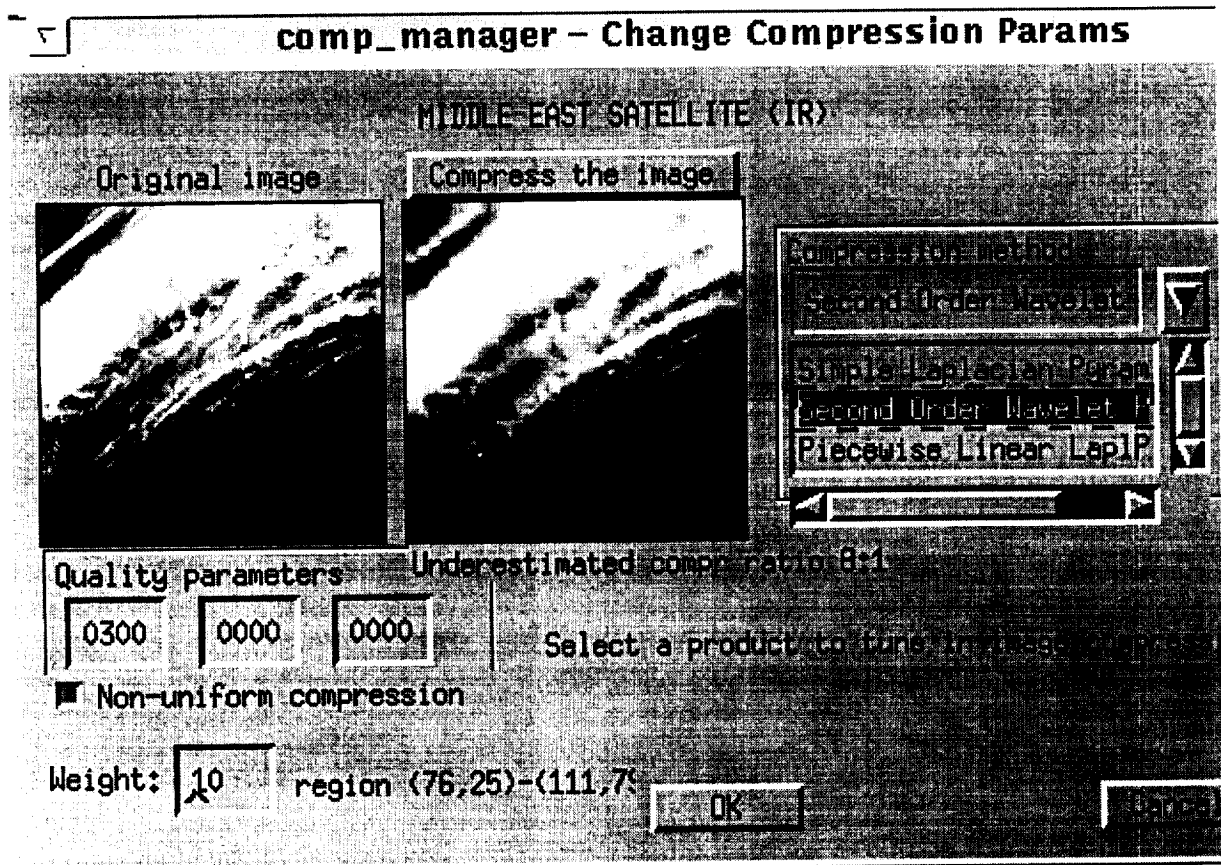


Fig. 3. Snapshot of the Component Manager screen with preview of non-uniform compression

Note that the compression ratio shown on the picture is significantly underestimated. The sample image is subsampled to be as small as 128x128. It is well known that smaller images are tougher to compress. In our experience, when the same method is applied to a full-size 512x512 satellite image, the compression ratio would be *at least* 4 times as much.

V. Challenges and future directions

The new version of the CSPD clients and server is being phased in. However, the system, and especially its satellite image delivery part is under further development. The challenges here are:

- Allowing interactive browsing of new imagery and letting a client know that the server has acquired a new image available for retrieving;

- Keeping local/customary configurations (*profiles*) of clients on the server, which would simplify the request protocol and allow
- Automatic delivery of weather products: each client receives (without an explicit request) periodical updates of information which it "usually" asks for. The information is also tailored to the current location of clients: for example, the region of special importance in non-uniform compression changes as the client moves;
- Binding of an image, that is pinning the picture down to the exact area on the globe it pertains to;
- Overlay of relevant geographical and meteorological information on a full-gray-scale satellite image.

Another challenge is integration of the request/delivery of satellite imagery and other grid products within `http/smtip` protocols: developing *weather-conscious* Web servers/browsers. One possible scenario: the user starts with a map of the entire world or some predefined region, as in the Xerox Map server [6]. The user can zoom in/out, etc., until he nails down precisely the area he is interested in. Then he can select weather products from a list of products and/or a list of product groups. A click on the submit button brings, after some communication delay, a familiar "weather map" view. Note, the server may have already recognized the user (or his site), so it can present him with a hot-list of his personal requests, that is, requests which were previously defined/asked for. Thus the user can skip on the area definition and communication, and simply say, "I want my usual". Also, it is very easy for the server to determine if there are new products added to the database since the last time the user had talked to the server. The server then can make a list of the new products on user's request. Since the server knows exactly whom it is talking to, it may also hide some products which are "for your eyes only".

It should be stressed that the scenario above looks very similar to the existing CSPD/NODDS interface, on the surface of it. The challenge now is the employing of standard tools as Mosaic, Netscape and `httpd`, and standard protocols, instead of proprietary clients and proprietary communication engines. Another difference is that the product lists and other configuration information are now stored on the server, not on a client site. Therefore, the information always stays current, and there is no need to waste bandwidth broadcasting product updates to all clients, even those who have already received updates, or who are not interested. One more difference of the proposed scenario is that the defined areas are also stored on the server (as a site "profile"). This arrangement cuts down on the communication time: the only information sent between the client and the server during the initial dialog is just the names of the cataloged areas/profiles. There is also no need to send the whole list of products being requested to the server, very often all the same list, and all over again: the list of products (with all needed configuration information and preferences) is already on the server. Keeping it there also helps in billing (should such feature be necessary) and accounting. Using standard Web browsers helps in one more respect: caching. Many Web browsers, Netscape for one, do their own caching, and do it quite efficiently and customizably. One can tell Netscape where to keep the cache, how often to purge it and how long to keep data in it (for a single session only, or for certain number of days, etc.).

We are also looking into using a new generation of Web browsers, based on the OpenDoc architecture, for example, CyberDog.

References

1. <http://www.metoc.navy.mil/metoc/>
2. http://mozart.compsci.com/pictures/sowlp_compress.html

3. Kiselyov, O. and P. Fisher, "Wavelet Compression with Feature Preservation and Derivative Definition," in *Proc. DCC'92, 1992 Data Compression Conference*, Snowbird, Utah, p.442, March 24-27, 1992
4. Kiselyov, O. and P. Fisher, "Pyramidal Image Decompositions: A New Look," in *Proc. DCC'93, 1993 Data Compression Conference*, Snowbird, Utah, p.479, March 30-April 2, 1993.
5. <ftp://replicant.csci.unt.edu/pub/oleg/README.html> or
<http://replicant.csci.unt.edu/~oleg/ftp/>
6. <http://pubweb.parc.xerox.com/map/>

**Data Management and Scientific Integration
within the
Atmospheric Radiation Measurement Program**

5313
P- 8

Deborah K. Gracio
Larry D. Hatfield
Kenneth R. Yates
Jimmy W. Voyles
Pacific Northwest Laboratory, Richland, Washington

Joyce L. Tichler
Brookhaven National Laboratory, Upton, New York

Richard T. Cederwall
Mark J. Laufersweiler
Martin J. Leach
Lawrence Livermore National Laboratory, Livermore, California

Paul Singley
Oak Ridge National Laboratory, Oak Ridge, Tennessee

1. Introduction

The Atmospheric Radiation Measurement (ARM) Program has been developed by the U.S. Department of Energy with the goal to improve the predictive capabilities of General Circulation Models (GCMs) in their treatment of clouds and radiative transfer effects. To achieve this goal, three experimental testbeds were designed for the deployment of instruments that will collect atmospheric data used to drive the GCMs. Each site, known as a Cloud and Radiation Testbed (CART), consists of a highly available, redundant data system for the collection of data from a variety of instrumentation. The first CART site was deployed in April 1992 in the Southern Great Plains (SGP), Lamont, Oklahoma, with the other two sites to follow in September 1995 in the Tropical Western Pacific and in 1997 on the North Slope of Alaska.

Approximately 400 MB of data is transferred per day via the Internet from the SGP site to the ARM Experiment Center at Pacific Northwest Laboratory in Richland, Washington. The Experiment Center is central to the ARM data path and provides for the collection, processing, analysis and delivery of ARM data. Data are received from the CART sites from a variety of instrumentation, observational systems and external data sources. The Experiment Center processes these data streams on a continuous basis to provide derived data products to the ARM Science Team in near real-time while providing a three-month running archive of data.

A primary requirement of the ARM Program is to preserve and protect all data produced or acquired. This function is performed at Oak Ridge National Laboratory where leading edge

technology is employed for the long-term storage of ARM data. The ARM Archive provides access to data for participants outside of the ARM Program.

The ARM Program involves a collaborative effort by teams from various DOE National Laboratories, providing multi-disciplinary areas of expertise. This paper will discuss the collaborative methods in which the ARM teams translate the scientific goals of the Program into data products. By combining atmospheric scientists, systems engineers, and software engineers, the ARM Program has successfully designed and developed an environment where advances in understanding the parameterizations of GCMs can be made.

2. History

Planning for the ARM Program began in the fall of 1989; a description of the initial program is available in the ARM Program Plan (U.S. Department of Energy 1990). The technical approach of the ARM Program and the design of the CART sites is discussed in more detail by Stokes and Schwartz (1994).

The design of the CART data systems was part of the initial program plan (Melton et al. 1991). The plan called for a distributed environment, the CART Data Environment (CDE), which included an Archive, an Experiment Center, and a collection of data systems distributed across all of the CART facilities for the collection and processing of data. The CDE was to be implemented “based on use of existing solutions wherever possible, evolutionary implementation of functionality, and parallel implementation of independent subsystems” (Melton et al. 1991). Figure 1 shows the flow of data through the CART Data Environment.

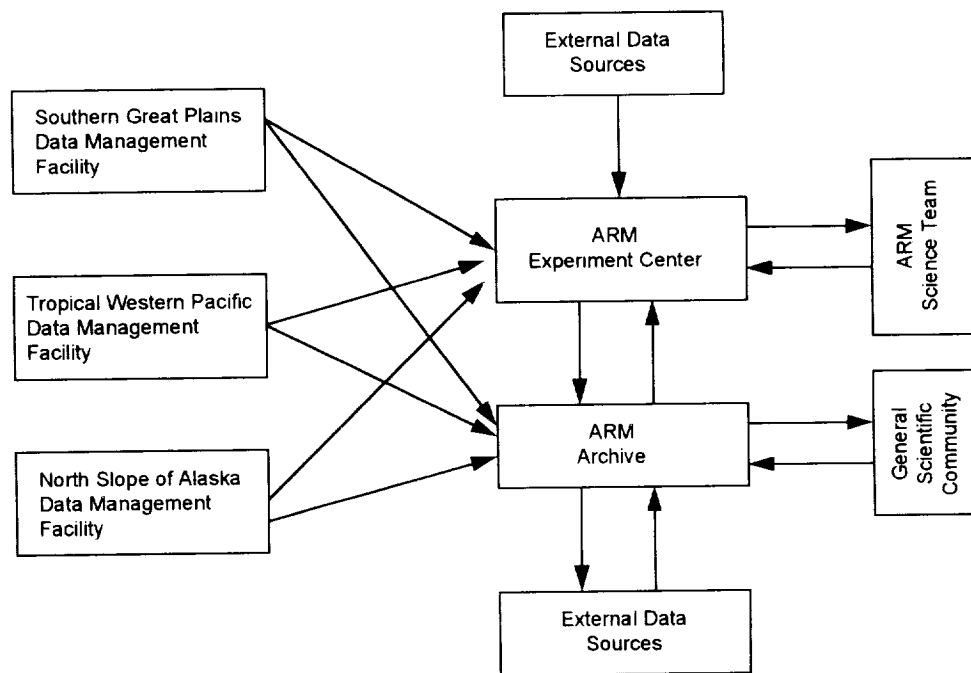


Figure 1: Data flow through the CART Data Environment

The Southern Great Plains (SGP) CART site was the first to become operational, with instrument deployment beginning in April 1992. Figure 2 is a map of the SGP site and the instruments deployed there. The site is slightly larger than a GCM grid cell, and is approximately 350 km (N-S) by 250 km (E-W). The central facility contains the greatest number and variety of instruments. The principle objective of these experimental testbeds is to quantitatively describe the spectral radiative energy balance profile within a wide range of meteorological conditions. Measurements are taken at a finer spatial resolution than GCMs utilize. The purpose for these measurements is to improve the parameterizations of sub-grid scale processes for use in GCMs. The sites will be operated for 7-10 years, continuously collecting measurements of atmospheric radiation and associated atmospheric and surface properties (Stokes and Schwartz 1994).

3. How the Technical Infrastructure Facilitates the Scientific Activities of the ARM Science Team

Interaction between the Data and Science Integration Team (DSIT) and the ARM Science Team (ST) is a primary mechanism that guides data collection and management within the ARM Program. This interaction leads to the translation of scientific goals and objectives into data requirements that define experiments for individual ST members. Figure 3 is a conceptual diagram of the science-related functions of the ARM technical infrastructure. The flow illustrated in the figure begins with a scientific dialogue between the ARM scientists,

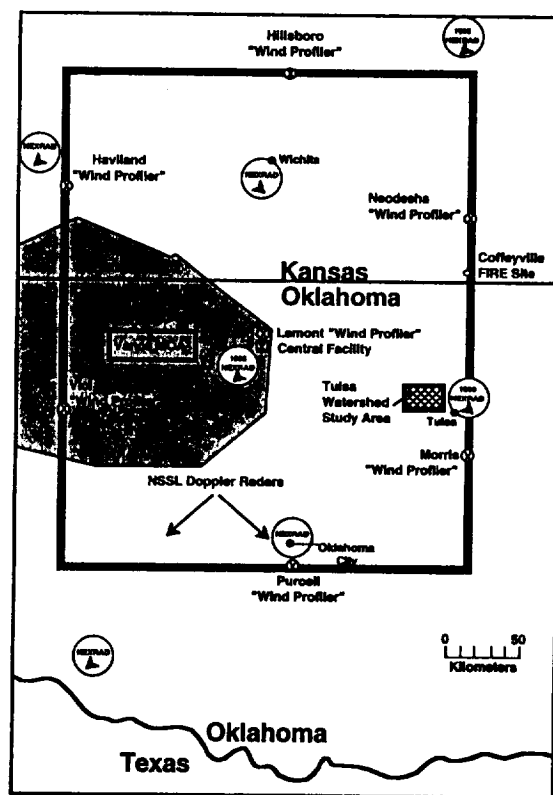


Figure 2: SGP CART site with Instrument Locations

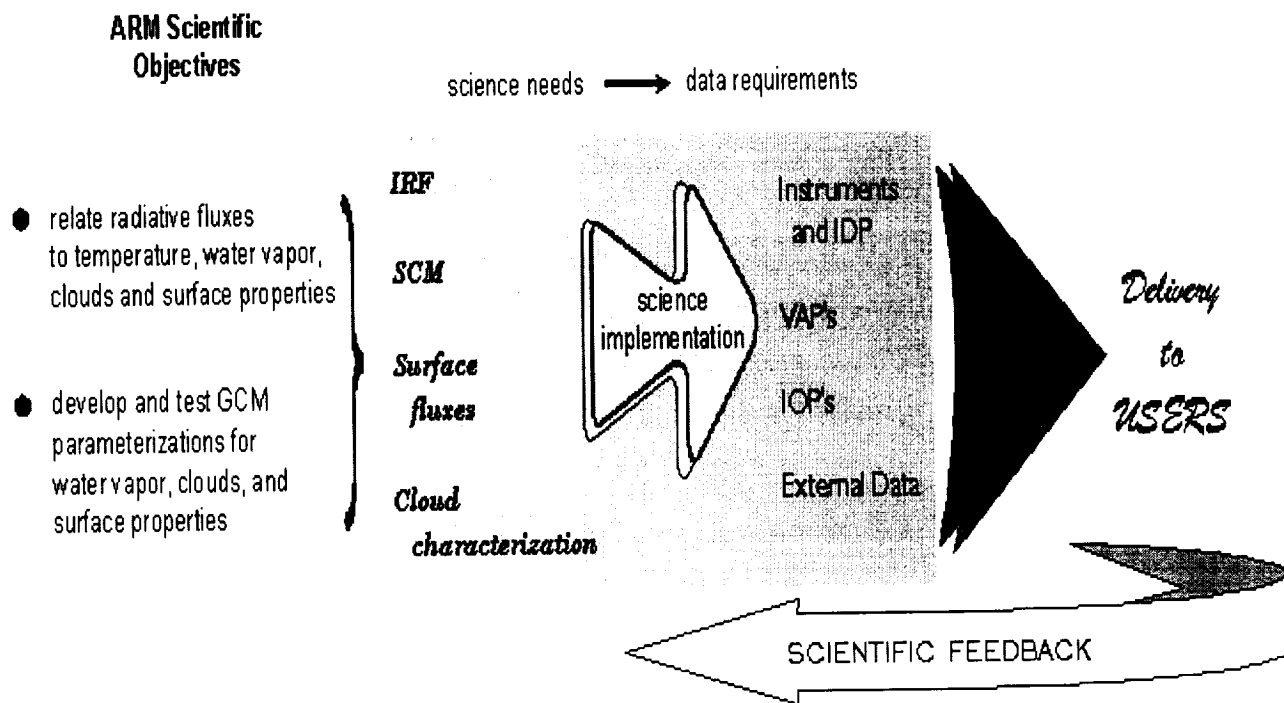


Figure 3 Scientific Functions of the ARM Infrastructure

collectively or individually, and the ARM infrastructure to define specifically what data are needed to accomplish the stated scientific objectives. From this point, the infrastructure implements the actions necessary to provide the required data to the scientists. Scientific feedback from users of the data is a key feature for improving the quality and usefulness of the data.

ARM is unique in its approach of interacting proactively with funded scientists. The interactive process between the DSIT and the ST is facilitated by a liaison from the DSIT, who understands the ST member's planned research and assists in the identification and acquisition of required ARM data. This process identifies data requirements not currently met by data within ARM. There are several actions that the ARM Program can take to obtain the required data. The acquisition of required data can present difficulties that are not encountered in the management of routine data. In some cases, the required data may be available from a source outside ARM.

Several steps are taken to enhance the observational capability of the applicable ARM Cloud and Radiation Testbed sites when data requirements point to the need for more observations. This enhancement can sometimes be as simple as the implementation of an Intensive Observation Period (IOP), where the temporal resolution of the observations is increased (e.g., more frequent radiosonde launches). Enhancement may be gained through the acquisition and

deployment of additional instrumentation at existing facilities. Finally, the development and testing of new instruments may be required, and this is done within the Instrument Development Program, a component of ARM.

In many cases, data requirements can be satisfied by new calculated data streams. A calculated data stream may be the merger of two or more observations to produce an unobserved quantity (e.g., vertical velocity from the horizontal wind divergence). Algorithms used in calculations may be defined by scientists within or outside ARM or developed within the infrastructure.

One of the strengths of the Science Team concept is the potential for synergism through interaction and information exchange among ST members. This interaction happens most naturally around science issues and related data sets of common interest. To date, the two most active interaction areas have been clear-sky instantaneous radiative flux (IRF) and single-column modeling (SCM). The DSIT is a catalyst for developing such interactive working groups and communicating the needs of the working group to the ARM infrastructure. The identification of “showcase” data sets of common interest helps focus ST and infrastructure attention that increases the benefit of the collected data sets and stimulates progress not possible by ST members working in isolation.

4. ARM Data Sources and Products

ARM data are normally collected in an ongoing, continuous manner, punctuated by IOPs. These two methods of collecting data complement each other. The “first-generation” of a data stream is the observations taken directly from the instrumentation in the field. The quality of the data is assessed and documented via simple minimum, maximum and delta threshold checks. The ongoing nature of regular ARM operations requires an automatic approach in analyzing the data. To this end, the concept of Derived Products has been defined.

A Derived Product is the definition of a procedure which creates a “second-generation” data stream by using existing ARM data streams as inputs and applying algorithms or models to them. The procedure is run automatically and continuously as long as there is input data, and the output (called a “derived product”) becomes a new data stream.

Prospective derived products may be identified by any part of the program, from instrument mentors to science team members or any place in between. There are two distinct types of derived products. The first type consists of data processing, including smoothing, interpolation, extrapolation, time synchronization of different data streams, and/or time averaging. Another example of this type of product would be a procedure that applies new calibrations to existing data, thus creating a new data stream. These products are designed to either reformat the data to make it easier for ST members or models to use, or to reprocess the original dataset to improve the quality of the data.

The second type of derived product generates new data streams derived either from physical models driven by inputs from existing ARM data streams, or from data quality comparisons. These algorithms come from ST members who are interested in the derived physical quantities or the models.

Quality Measurement Experiments (QMEs) are a subset of the derived products designed specifically to enhance the ARM data quality by providing ongoing, continuous data streams derived from the intercomparison of different ARM data streams. QMEs are part of a two-pronged effort to ensure that ARM data is of known and reasonable quality. The first approach focuses on self-consistency within a single data stream, using various automated methods at the time of processing. QMEs, in contrast compare multiple data streams that are somehow related against a set of expectations. A QME provides the capability to identify data anomalies, such as inconsistent data across instruments, incorrectly implemented or inconsistent algorithms, and the information needed to identify the root cause of these anomalies.

To meet the scientific objectives of ARM, it is necessary to augment observations from the ARM sites with data from other sources; these are called external data products. External data products which are currently being acquired to augment measurements at the Southern Great Plains site include surface observations from the Oklahoma Mesonet and from a Kansas Mesonet Network operated by the University of Kansas. National Weather Service (NWS) surface and upper air data, the NOAA Wind Profiler Demonstration Network data, including RASS data as they become available, GOES and AVHRR satellite observations, analyses products from the NMC Eta and RUC models and Stage III precipitation products obtained from the Arkansas Basin River Forecast Center. Some data sets are acquired in near real-time, while other acquisitions are delayed until the responsible organization has had the time to perform quality assurance on the data sets.

Products derived from some of these external data sets or from combinations of the external and ARM data are either currently being generated or are planned. Examples include the cloud coverage statistics derived by a group at NASA Langley from the GOES satellite, and integrated surface mesonet data sets which incorporate the various surface meteorological data stations.

It is anticipated that the need for external data will increase with the addition of the CART sites in the Tropical Western Pacific (TWP) and the North Slope of Alaska. Arrangements are already underway to acquire GMS and AVHRR data for the TWP CART site.

5. Challenges

The ARM DSIT team is made up of a diverse set of disciplines including electrical, mechanical, and software engineers, mathematicians, physicists, and atmospheric scientists. Members of the team are located at five different national laboratories, thus, increasing the requirements for enhanced technological communications. Electronic mail, phone and video conferencing, and travel have become significant contributors to communication among team members.

The group responsible for designing and implementing the ARM SGP data system faced a number of challenges. Specifically, the implementation of a complex distributed system in a relatively short amount of time, with team members located in three separate time zones. Furthermore, the SGP site was not close to any of the laboratories at which the team worked and since it was located in the middle of farm land, the site was started with no communication facilities. The team was designing a system which was to run continuously for an estimated 10 years and be capable of near real-time delivery of data to its ultimate users, the ARM Science Team.

The ARM Program Management developed a general method of tracking and resolving problems by means of a subgroup, the Problem Review Board (PRB), which meets via weekly conference calls. A database was developed to assist the PRB by recording all reported problems via Problem Identification Forms (PIFs) and the resolution of problems via Corrective Action Reports (CARs). In addition, the quality of data streams are documented and stored in the database in the form of Data Quality Reports (DQRs).

6. Collaboration with Researchers and Research Programs outside ARM

Collaboration occurs on both the individual and program level. In many instances, scientists outside ARM conduct research related to ARM scientific objectives. The exchange of data and scientific ideas is mutually beneficial to both parties. In some cases, ARM has strengthened these collaborations by identifying the scientists as adjunct ST members. In other cases, ARM has established funding arrangements for providing desired data to ARM (e.g., NASA providing satellite-derived cloud products).

Collaboration with ongoing research programs is an important part of the ARM approach for meeting its scientific objectives. Since ARM addresses a specific part of the overall global climate problem, other programs can provide scientific understanding, observational approaches, algorithms, and data that enhance the results of ARM-funded research. Also, by combining ARM resources with those of other programs, certain scientific goals can be achieved that individual programs could not achieve on their own (e.g., ARM and SHEBA interactions in the Arctic). The density of instrumentation and the long-term period of data collection at ARM CART sites have attracted several programs that wish to take advantage of the ARM sites as benchmarks. Examples of collaborations include the Global Energy and Water Experiment (GEWEX) through its GCIP subprogram in the Southern Great Plains, and the TOGA-COARE program in the Tropical Western Pacific.

7. Future Direction of the DSIT

Translating science needs into data requirements and delivering data streams of known and reasonable quality are fundamental principles of the ARM Program. Maturity of our capability to realize these principles will enhance the scientific productivity of the ARM Principal Investigators (PI).

To keep pace with the increasing capacity of ARM PIs to make progress toward the ARM programmatic objectives to improve GCMs and understand the atmospheric radiation-cloud

radiative feedback, the DSIT will maintain a vigorous policy of upgrading the software and hardware of our data system and optimizing the critical loop between our applied science and modeling efforts with those of the ARM ST. In essence, to make more high quality data, information, and derived products available to our customers.

Focus groups within the DSIT are working to develop effective methods for designing derived products, our system development activities, and the operation and maintenance of our data environment. These focus groups, along with the infusion of new technologies, and the utilization and re-use of previously developed tools, are moving us toward an open-architecture approach to product delivery, processing, tracking, and characterization of data streams. In particular, the use of the World Wide Web for cross-platform access to data and information, object-oriented database techniques to manage meta-data relations at our archive, an integrated development, operations, and maintenance approach, and standard data analysis display tools will continue to have a positive impact.

In the future, as we develop new system requirements and plan the integration of new technologies and algorithms, we will keep our principles and the scientific objectives of the ARM Program in view. Understanding the needs of our customers and how well we meet those needs form the basis of the operational measures of the effectiveness of the DSIT.

References

Melton, R. B., Campbell, A. P., Edwards, D. M., Kanciruk, P., and Tichler, J. L. 1991. *Design of the CART Data System for the U.S. Department of Energy's ARM Program*. American Meteorological Society Proceedings.

Stokes, G. M., and Schwartz, S. E. 1994. *The Atmospheric Radiation Measurement (ARM) Program: Programmatic Background and Design of the Cloud and Radiation Testbed*. Bulletin of the American Meteorological Society. Vol 75, No. 7, 1201-1221.

U.S. Department of Energy. 1990. *Atmospheric Radiation Measurement Program Plan*. DOE/ER-0442.

Object-Oriented Structures Supporting Remote Sensing Databases

Keith Wichmann
Global Science & Technology, Inc.
Code 935
NASA/Goddard Space Flight Center
Greenbelt, MD 20771
kwichma@balmure.gsfc.nasa.gov

Robert F. Crompt
Code 935
NASA/Goddard Space Flight Center
Greenbelt, MD 20771
crompt@sauquoit.gsfc.nasa.gov

Abstract

Object-oriented databases show promise for modeling the complex interrelationships pervasive in scientific domains. To examine the utility of this approach, we have developed an Intelligent Information Fusion System based on this technology, and applied it to the problem of managing an active repository of remotely-sensed satellite scenes. The design and implementation of the system is compared and contrasted with conventional relational database techniques, followed by a presentation of the underlying object-oriented data structures used to enable fast indexing into the data holdings.

1 Introduction

In the late 1990s, NASA will launch a series of satellites to study the Earth as a dynamic system. The information system required to manage data from these satellites will be one of the world's largest, requiring a capacity about 1,000 times the size of the Library of Congress. The enormous size of the data holdings and the complexity of the information system pose several challenges to computer scientists, who must apply advanced techniques from software engineering, artificial intelligence, computer graphics, and signal processing if there is any practical chance of success.

A data management system should provide users the ability to locate data pertinent to their needs. Limited computational resources, however, directly impact the level of detail users can employ in formulating their queries. In an extensive remote sensing repository, it is both impractical and unnecessary to manage every individual pixel of each satellite scene. Such an extreme level of granularity is best left to individual users who can apply a geographic information system to a restricted number of images. Rather, it is better to record limited information about each scene, and use this metadata to satisfy users' queries.

For a number of years, researchers in the Information Sciences and Technology Branch, Code 935, at NASA/Goddard Space Flight Center, have strived to build a fully automated Intelligent Information Fusion System (IIFS) that enables unsophisticated users to access up-to-date satellite imagery [2, 4, 9]. To enable efficient search of large data holdings, the IIFS uses novel methods for extracting image content from the data, specialized data structures for storing data acquired from all over the world, and object-oriented representations to express the complex interrelationships pervasive throughout scientific domains. These advanced techniques are transparent to the users, yet allow for fast and easy access to data which meet the users' needs.

2 Rationale Behind an Object-Oriented Approach

Until recently, designers had no real choice but to apply a relational database when confronted with managing any sizable amount of data. In the late 1980s, however, commercial vendors made it viable for system builders to consider using object-oriented databases for limited purposes, most generally in an evaluative mode. At the same time, educational institutions introduced an object-oriented philosophy in their computer science classes, resulting in a new generation of computer scientists who have been weaned on this approach.

Seemingly, object-oriented databases are now an alternative to relational databases. However, as with any new technology, it is essential that a comparison be drawn with the older, established technology.

A relational database management system (RDBMS) uses the relational algebra as described by Codd [3]. This was a calculated tradeoff from the flexibility that the CODASYL model provided. In return for a limitation on flexibility, RDBMSs provide an increase in performance because of the simplicity of the relational algebra in query processing. In order to apply the relational algebra, the data must be normalized to a well-defined form. The manual process of applying normalization can be time consuming and error prone. Furthermore, normalization can also lead to performance problems during query processing when dealing with complex relationships [10]. In applications that have simple relationships, the relational database is a good choice.

However, in the remote sensing domain, the model can be quite complex to fully express the functionality of the system. In cases like these, the object-oriented database is more likely to perform better. Baroody and DeWitt [1] have previously shown that the object-oriented approach can perform all functions of a relational database. In addition, by allowing the equivalent of pointers from a language such as C to be stored in the database, OODBMSs can store more complex structures and relationships than RDBMSs. This reduces the need for mapping data from their application storage to their database storage (sometimes referred to as the "semantic gap" or an "impedance mismatch"). In many cases, the OODBMS ties in to a host language (e.g., C++) such that there is a direct one-to-one mapping because the structures are identical. Naturally, there are tradeoffs here as well. The mapping causes an extra step whenever an object migrates into memory to set its pointers to something valid. Once an object is in memory, no additional database overhead is necessary until the transaction commits.

The database can represent any in-memory data structure because of this representation strategy. While the relational database relies on associative lookups, joins and indices on key attributes, the object-oriented database can employ domain-specific structures for providing fast access directly to the object or objects of interest. In essence, the database uses structures which allow a very quick, navigational search that only involves referencing a small number of objects. RDBMSs must adhere to the table structure and as such have difficulty representing spatial data and other complex structures [5]. It is necessary to write code to convert the in-memory structure to the relational structure and vice versa [8]. OODBMSs can represent the domain-specific search structures without changes to the database server itself.

The OODBMS differs in the balance of responsibility afforded the client and the server. An RDBMS, in general, performs all processing on the server side and only communicates the results to the client. An OODBMS tends to perform all of its processing on the client side. This leaves the server to handle accessing the disk and transmitting the raw data to multiple clients, allowing the workload to be better distributed.

As a compromise to the two models, hybrid databases improve over RDBMSs by storing complex objects in the nodes of a table, but do nothing to improve on the expressiveness of the relationships. Examples of hybrid databases include Illustra and Postgres. Hybrid models support SQL queries, a standard from the RDBMS community, and researchers/vendors in OODBMSs have begun to embrace this notion. While the hybrid database allows more complex objects, it does so with a cost in performance. It does provide an environment more familiar to RDBMS users, but it does not have the full capabilities of an OODBMS.

The IIFS utilizes an OODBMS for several reasons. The remote sensing domain can be very complex. The need for the system to be scalable in the number of items of data that it can handle causes several problems that an OODBMS can solve. First, the table structure of relational data requires that more data be accessed than is absolutely necessary to carry out a search since all data in the table are grouped together on disk. As the number of objects in the database grows, then the amount of data touched also increases. The OODBMS allows a search structure that facilitates only accessing data that match the search attribute for that structure. Next, the OODBMS requires less coding to convert data into the proper form for storage leading to fewer coding errors and less development time. It is necessary to introduce transactions, but pointers map directly into the database. Finally, the distributed nature of most OODBMSs allows the exploitation of CPU cycles spread across the network.

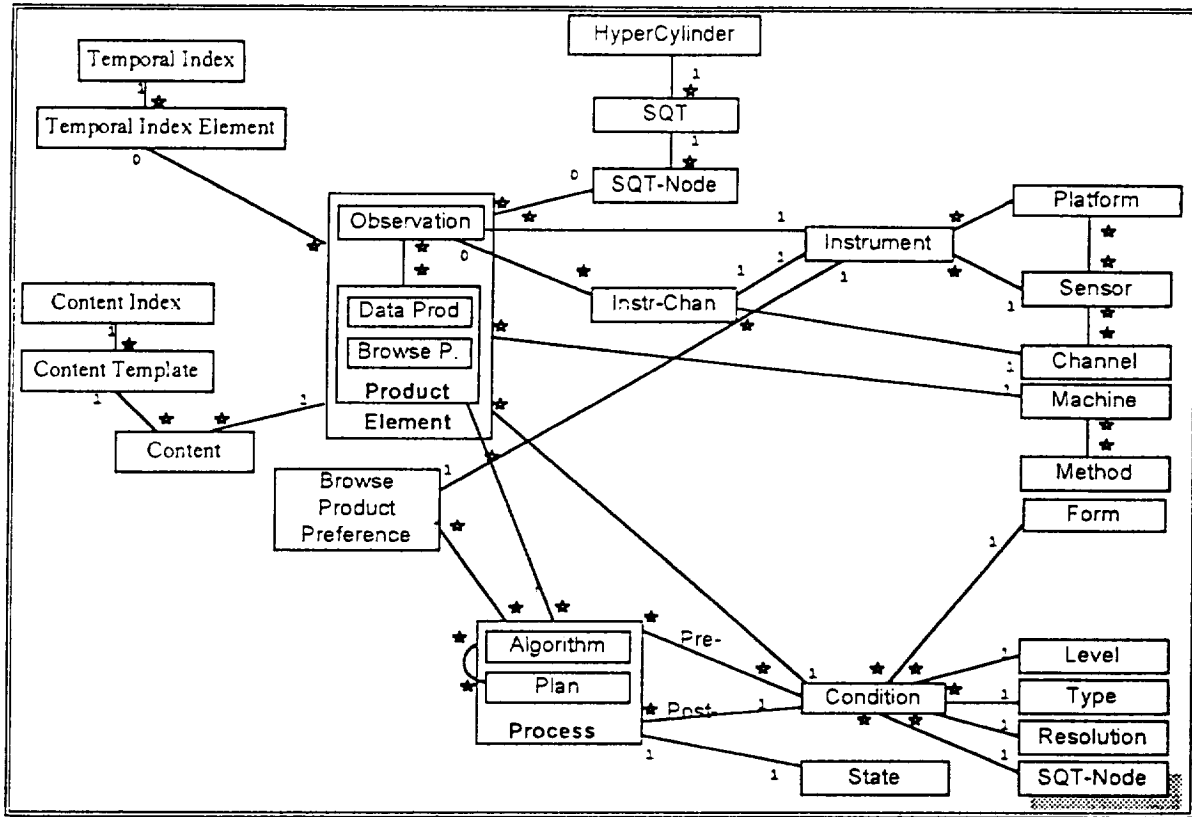


Figure 1: The IIFS Object Model

3 The Object Model

The IIFS employs the object model that is partially illustrated in Figure 1. The diagram is a minor diversion from standard entity relationship diagrams that incorporates the notion of inheritance. Each rectangle represents a class. A superclass is indicated by a containing rectangle. The lines indicate the relationships between the classes and the numbers indicate the cardinality of the relationship with a star representing "many."

The platform, sensor, channel, instrument and instrument channel classes represent the source of the data. The platform is the physical, atomic unit on which the instrument is mounted.

- In space, a satellite such as LANDSAT-5 or the space shuttle is a platform.
- In the atmosphere, a weather balloon or an airplane can be a platform.
- On the ground, a platform gathers in-situ data.

The sensor is the device that gathers the desired data. It has some number of channels, each of which gathers its data at the same time over the same area (e.g. Thematic Mapper is a sensor aboard LANDSAT 5). Channels represent the frequency response of the sensor. Each channel looks at a different yet possibly overlapping frequency range. An instrument is the combination of a specific sensor on a specific platform, while an instrument channel is the combination of a specific channel on a specific instrument. Of particular note, a sensor can be on many different platforms, and a platform can carry many different sensors. All of these classes when instantiated have the ability to report back to the calling program a set of elements that matches their particular instance.

There are three other major attributes used to search for sets of data: temporal, spatial and content. The temporal search employs the temporal index and temporal index element objects. The spatial search uses hypercylinders, sphere quadtrees (SQTs) and sphere quadtree nodes [6]. The content search applies the

```

(query (instrument LANDSAT-5 TM TM-4)
      (complexdate 1 1 4 85 2 4 85 4095 15 1 31))
(query (containment (latlong 33 21 35 23))
      (content USGS (forest 20 40)))
(dump Content)
(ingest iifs@balmure:/mass/data LANDSAT-5 TM 5/4/90 33.2 -43.4)
(add-template LandUse (water urban agriculture rangeland forest))

```

Figure 2: Sample s-expressions for communicating between modules of the IIFS

content index, content template and content object classes. Each of these has the capability of reporting back the set of data that corresponds with the attribute value.

These search structures extract elements from the database that match the given attributes. The primary data structure in the IIFS is an element. Elements in the database are either observations directly from an instrument, or products derived from the original raw data (observation). Since algorithms produce the derived data, information about the routines stored includes the machines it can run on, the accounts needed to access the data and executable program, any necessary parameters, and a record of applied algorithms and their order or application. In this way, if a better algorithm develops, it is possible to regenerate data products that used the old algorithm.

The storage of algorithms allows the implementation of semantic caching. Since storage space will surely never be large enough to store everything, the database will evaluate the data to see what can be deleted safely. The IIFS can recalculate products when needed if the database stores all of the original data as well as all algorithms. Therefore, a good choice would be to remove the least recently used data product from the archive. The system then marks the element as not available. The history facilitates the regeneration of the data by reapplying the algorithms in order.

The IIFS also relies on a planner, scheduler and dispatcher to interact with the data stored in the database. The planner generates the necessary steps to produce data products that do not currently exist or to process newly ingested data sets to get the associated metadata, and in this vein, it acts as a transaction manager. The browse product preferences in the object model lists the algorithms that should run on all new data to generate the metadata. The planner incorporates policies that the user provides such as “accuracy is preferable to shorter compute time” in order to generate the desired product. The scheduler accounts for the resources available in the system and arranges for simultaneous requests to be processed producing an estimate of time required. The dispatcher implements that schedule and monitors the progress of each process.

The entire implementation is in C++. An interface to the outside world is necessary so that all programs that utilize the database do not have to be clients of the database itself. A server exists that accepts a specialized language of s-expressions that describes the query and searches the database on the behalf of the program. Remote procedure calls (RPCs) carry the s-expression language to the server and the results back to the client. The language allows querying of the elements in the database, ingesting of new data, ingesting of new content templates and the extraction of extent information. Multiple interfaces can share this common interface into the database. Figure 2 shows some examples of the s-expression language.

4 Query Objects

The object-oriented domain also allows a useful encapsulation of the query as illustrated in Figure 3. In the IIFS, a query is simply a list of query elements. A query component is an abstract class used to encapsulate the processing of a query and the retrieval of the representation of the query for passing to a query optimizer. It has subclasses that represent each attribute type used for a search such as temporal, spatial, source and content searches. In addition, there is an abstract subclass called query connectors used to provide logical connectives such as AND and OR. Each connector has a list of the query components that it is connecting and any element of that list can be another query connector. A connector object can also be one of the

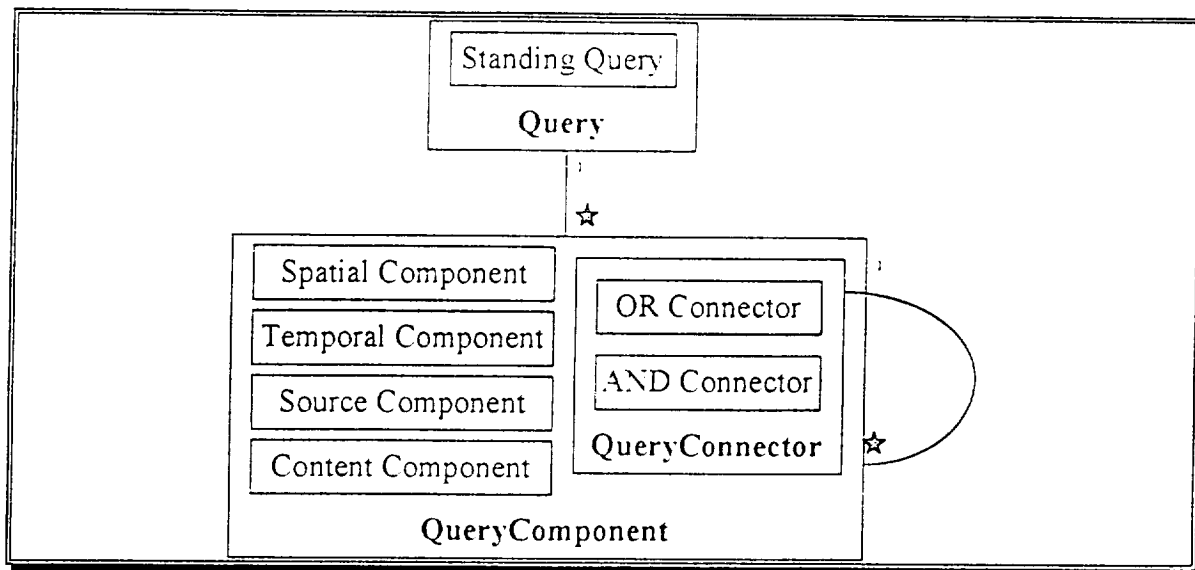


Figure 3: Query Objects

items in the list of query components maintained by a different query connector, thus allowing arbitrary combinations of query components. A query assumes a logical AND connector for each of its elements.

The IIFS also allows specification of a query as a standing query. This is a query that executes at some specified interval automatically. The results are emailed to a list of users until the expiration date supplied with the original query. Each mail message also includes the information about the query and a unique identifier for specifying a query to remove from the system. The system requires this extra information at the creation of the query. The natural mapping of in memory structures to database structures means that there is no additional work to store this in the database other than maintaining the list that has all the standing queries in it.

Each query component can return a string that the OODBMS query optimizer can use as input for a search. The OODBMS generic query processor does not know how to use the navigational search structures built into the database, so it must rely on B-trees, hash tables and simple iteration over the existing objects to determine the query result. Each component can also return the set of data that matches the attributes that it describes by utilizing the navigational search structures described in detail below. This flexibility allows quick implementation of a variety of search techniques and becomes important in the section on multiple attribute searches.

4.1 Spatial Searches

The IIFS employs sphere quadtrees (SQTs) as the spatial search structure [7]. The SQT, in addition to providing a logarithmic search, better represents the globe by not introducing artifacts at the poles as is the case with latitude/longitude-based structures. The SQT uses an icosahedron (a 20-sided polygon) with triangles for each face as its basis. Each face subdivides into four sub-triangles (trixels) recursively. To achieve a resolution of one kilometer over the face of the Earth requires building a SQT to 14 levels, implying that any cell of this tiling can be indexed by answering 14 4-way questions, or 28 yes-no questions. Furthermore, the resulting geometry uses all integer math that produces much faster results than floating point calculations as might be encountered using latitudes and longitudes. Each trixel that has no children (leaf trixel) maintains a collection of the elements that fall in its region. The trixel splits into four new subtrixels when there are too many items in the bin. The system then inserts the original trixel's elements into the appropriate sub-trixel's bin, where the bin size limit can be a fixed number or can vary depending on the depth of the tree.

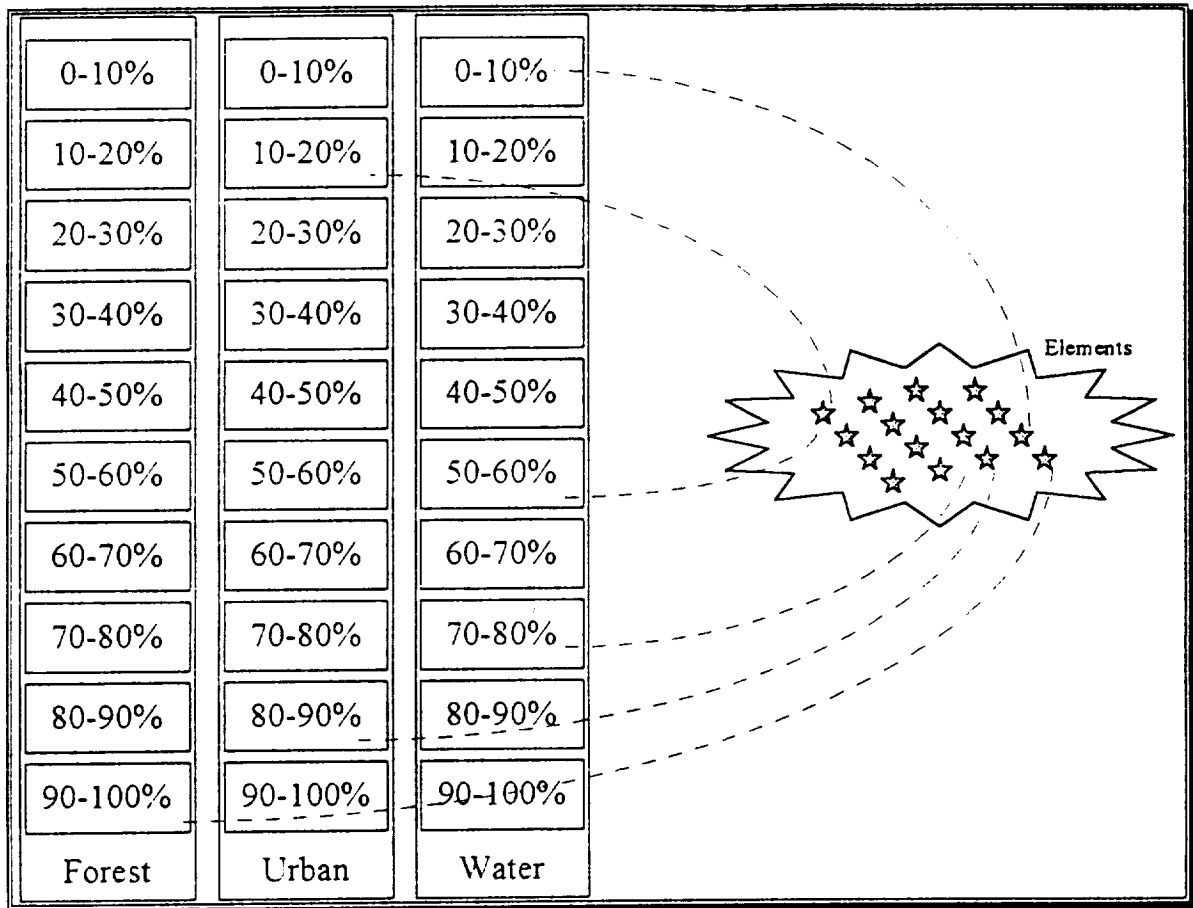


Figure 4: IIFS Content Index Section

4.2 Temporal Searches

Each piece of data described in the IIFS has a reference in the temporal index as well. The structure consists of an indexed list of Julian days, each with a list of the elements that fall on that day. The IIFS creates the day object only at the insertion of the first element with that date. A B-tree indexes the days so that range searches perform very quickly. The system performs a variety of temporal searches down to the granularity of a day. Granularities less than a day require accessing the actual data.

4.3 Content Searches

Searches based on content help fulfill two major requirements of the IIFS. First, the user should be able to search for data in terms that are familiar to their own domain. The system accomplishes the first by requiring algorithms to interpret data into that domain. The second major requirement is that the IIFS must provide the capability to return the smallest possible group. As the number of objects in the system grows, the number of objects returned by a given query will also increase. If no additional attributes are available to search on, this requirement will fail.

The IIFS system allows the introduction of various content templates. Each template consists of a name for the template and a name for each attribute of the template. The content object stores the data as a vector of numbers that correspond to the order of the attributes in the template representing the percentage of that attribute in the data. For instance, a template might be used for land cover. Its attributes might be agriculture, urban, water, forest, rangeland and barren with the vector representing what percentage of the data represents each of those classes. The database maintains a list of all templates. Each template

lists all elements that have had its classification algorithm run. Another step in precomputing all possible queries would be to break up the range of values (typically 0-100 percent) into bins. The bin points to the data that fall in that range for each attribute in the template. This would require more storage, but would produce quicker results during a query. Figure 4 shows a partial example. It shows three of the attributes in a land use/land cover template divided into ten bins each representing ten percent ranges. There are entries pointing to some of the elements in the system that have the given properties. By dynamically adjusting the bin sizes and adding new bins as they reach a given limit, the search can always be logarithmic.

This allows the addition of new templates at any time, but the system will reexamine old data sets to classify them using the new template. This could be a very costly process, but flexibility can be added to the system to dictate when that processing should occur.

4.4 Source Searches

The source in the IIFS models the platform and sensor that it came from. This is an excellent example of where the complexity of the domain is apparent. There is a many to many relationship between platform and sensor that expresses very easily in OODBMSs but requires a mapping to an extra table in RDBMSs. The platform describes the physical location of the instrument used to get the data. The sensor describes the physical properties of the device, including the wavelengths to which it responds. The same sensor can be on many different platforms, and one platform can have many sensors on it. The search structure is identical to this description. Since there are not many platforms, that part of the search executes very quickly, but a hash function minimizes the search time. Each sensor maintains a list of data that it produced. Each piece of data maintains a list of channels that are missing. This list is normally empty and searches of the list are not necessary. This implies that the sensor gathers all channels simultaneously.

4.5 Multiple Attribute Searches

A single attribute search is not expressive enough for most searches. The IIFS allows multiple attribute searches using any combination of attributes. Figure 5 shows the four major search structures with selected connections to the elements. Each attribute as represented by a query element does its own search, each of which is logarithmic. The system then combines the results conjunctively or disjunctively as declared. More elegant strategies are being implemented.

The first strategy is to evaluate which attribute will produce the smallest return set. This implies that there exists a quick way to determine the approximate number of items returned, which is not an easy problem. Obviously, the search stops if any attribute returns zero elements. Furthermore, by performing the intersection on the two smallest sets, the system performs the least amount of work. This also reduces the number of comparisons in the following stages.

The second strategy involves employing the query mechanism of the database. The database can also apply this in addition to the first strategy, but does not require it. Once the search produces a set that is less than some threshold value, the database's query mechanism iterates over each remaining element testing each member for the remaining attributes simultaneously. Indices will not exist previously because the system created the set dynamically. However, the database could create an index on the set if it is beneficial.

4.6 Correlated Queries

A correlated query is a query in which the qualification for membership in the result set is a relationship (correlation) to another element in the result set. This kind of query can never return a single hit because a relationship requires two elements. The return elements will have a relationship with another element in the set. For instance, the query might be for data covering locations where two different sensors passed over in the same hour. This capability allows searches for relationships in the data, and is a very powerful addition to simple searches for attributes. The above search breaks down into three stages. The first stage is a query for the first set of eligible elements from the first platform. The second stage is for the eligible elements from the other platform. The third and final stage tests for the existence of the relationship between the two sets. The IIFS creates and fills in a special bin structure with two bins for each time slot. The reason for two bins

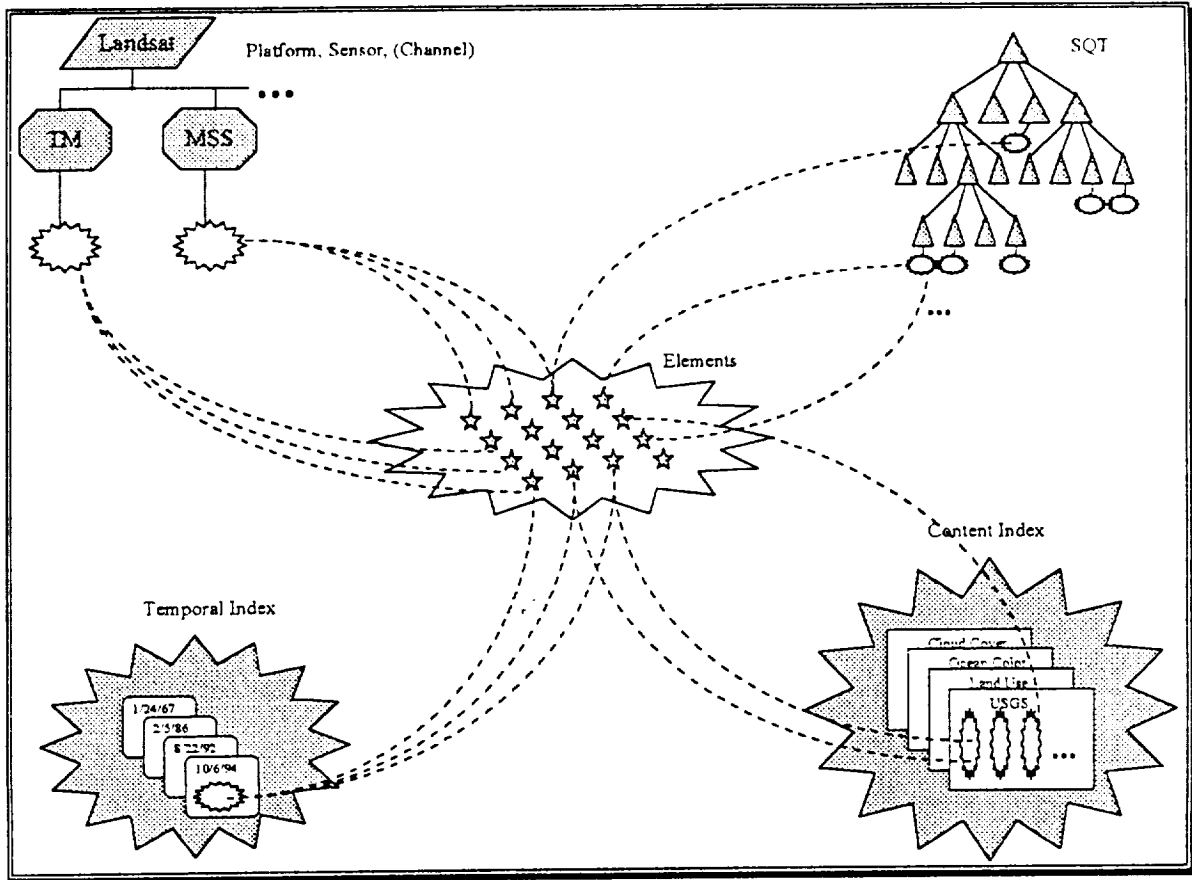


Figure 5: IIFS Search Indices

is to keep hits from the different platforms separate. It could use a single bin, but then it would have to search to see if there are entries from both to determine if the contents of that bin are a hit. By separating them, two non-empty bins from the same time period constitutes a hit and provides an ordering for the hits. Figure 6 illustrates the structure used.

In the case of spatially-correlated queries, the sphere quadtree acts as the bin structure. The IIFS creates two sphere quadtrees and populates them with data from each query. Similarly, the database can correlate queries by source or content. Finally, the system might implement multiply correlated queries. One example would be a search for data from two different platforms that cover the same area and occur in the same period. Also, more complex relationships than equality or proximity would be useful. This would enable a query like two images taken three months apart over the same area.

4.7 Distributed Queries

Distribution of the data can increase the throughput of the system and decrease response time for a given query. In the IIFS, the spatial search structure has some properties that lend themselves to distribution. Each of the twenty faces of the icosahedron can be distributed to separate CPU nodes. If any machine node approached a predefined saturation level, its data could be redistributed to other nodes. The IIFS maintains a master index so that the distribution system could pass queries to the appropriate machine for processing. Throughput increases because independent queries of physically separated areas execute simultaneously. Response time increases for queries that cover more than one face of the icosahedron because the search for the remaining attributes can occur in parallel.

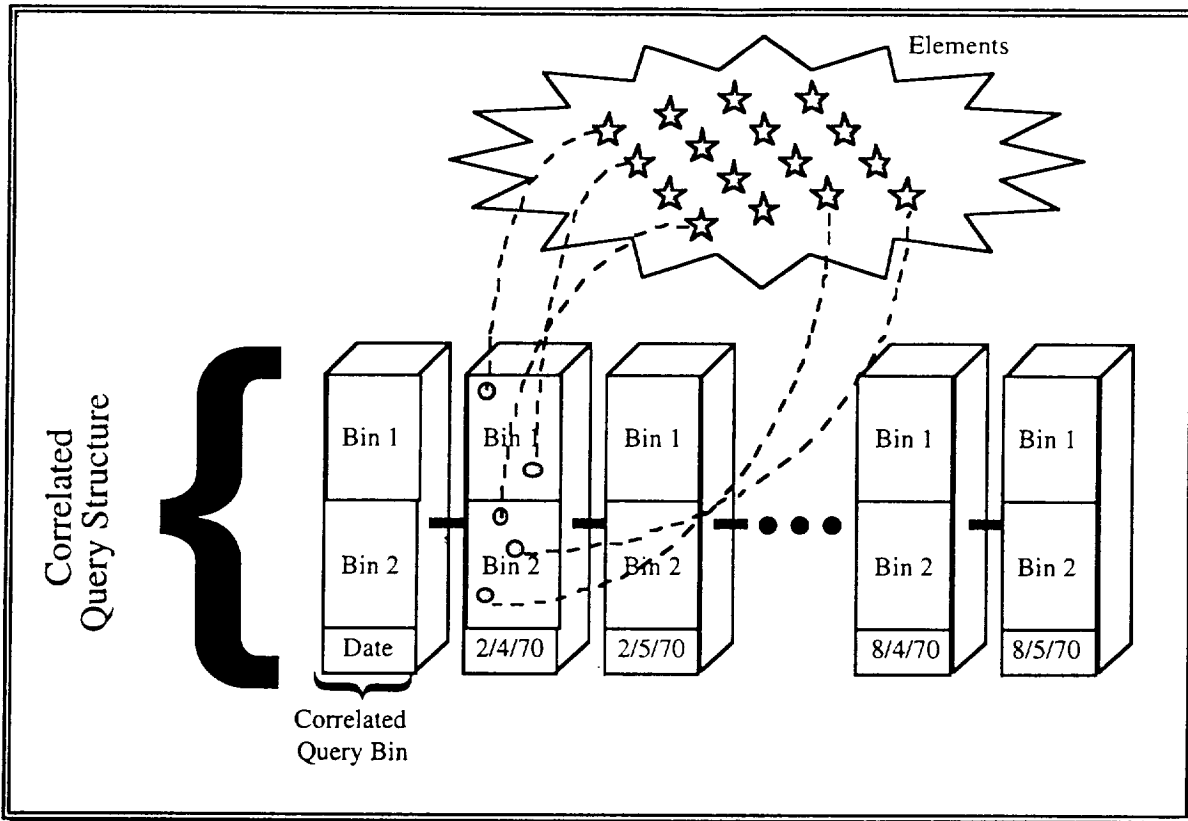


Figure 6: Bin Structure for Correlated Query

5 Current Capabilities

The IIFS in its current form supports queries on the four basic attributes described above: temporal, spatial, source and content. The sphere quadtree stores the center point of each image and allows logarithmic access to those points. The hypercylinder data structure is currently being implemented and will enable the IIFS to return data sets that more accurately match the user's area of interest.

The IIFS implements most of the structures discussed. Notable exceptions are all types of correlated queries except temporal, and the algorithm-related classes. The system uses the techniques described for distribution to segment the database internally but no work has been performed on distributing the SQT over multiple CPU nodes. Instrumentation of the database is currently in progress to produce base figures for evaluating changes to algorithms and storage structures. Ingesting large amounts of random data into the database tests the scalability of the system. The system is implemented on a Hewlett Packard 735 workstation using Object Store from Object Design Incorporated.

A graphic user interface developed in Tcl/Tk implements a front-end to the system for providing queries and displaying results. It communicates with the database using the s-expression language over an RPC connection. The IIFS also includes an interface using the world wide web as the front end, but it is not quite as flexible. Active research is progressing on techniques for extracting useful content information from remote sensing imagery to capitalize on the content searching capability of the system.

6 Conclusions and Future Work

The IIFS system is serving as an active development and research environment for testing scalability and evolvability goals. The base system is nearing completion, allowing for future studies of new data structures and algorithms as well as the effect of minor changes in the current implementation. The current prototype

shows that the techniques implemented are sound and can provide the basis for a more complete system. Future work includes the extension of the spatial search structure to handle the differences in the granularity of the data stored in the database, additions to the content based search structure to increase its performance, and the examination of a number of techniques for handling multiple attribute queries. Furthermore, the system will examine and implement additional types of correlated queries.

Acknowledgments

We would like to thank Bill Campbell (NASA/GSFC), Nick Short (NASA/GSFC), Gyuri Fekete (Hughes), Larry Roelofs (GST), and Walt Ligon (Clemson U.) for their contributions to this work.

References

- [1] A. J. Baroody and D. J. DeWitt. An object-oriented approach to database system implementation. *ACM Transactions on Database Systems*, 6(4):576–601, 1981.
- [2] W. J. Campbell, S. E. Hill, and R. F. Crompt. Automatic labeling and characterization of objects using artificial neural networks. *Telematics and Informatics*, 6(3–4):259–271, 1989.
- [3] E. Codd. A relational model for large shared data banks. *Communications of the ACM*, 13(6):377–387, 1970.
- [4] R. F. Crompt, W. J. Campbell, and N. M. Short, Jr. An intelligent information fusion system for handling the archiving and querying of terabyte-sized spatial databases. In *International Space Year Conference on Earth and Space Science Information Systems*, pages 586–597. American Institute of Physics, 1993.
- [5] R. F. Crompt and S. Crook. An intelligent user interface for browsing satellite data catalogs. *Telematics and Informatics*, 6(3/4):299–312, 1989.
- [6] R. F. Crompt and E. Dorfman. A spatial data handling system for retrieval of images by unrestricted regions of user interest. *Telematics and Informatics*, 9(3/4):221–241, 1992.
- [7] G. Fekete. Rendering and managing spherical data with sphere quadtrees. In *Proceedings of the First IEEE Conference on Visualization*, San Francisco, California, October 1990.
- [8] M. Loomis. *Object Databases: The Essentials*. Addison–Wesley Publishing Co., 1995.
- [9] N. M. Short, Jr., R. F. Crompt, W. J. Campbell, J. C. Tilton, J. L. LeMoigne, G. Fekete, N. S. Netanyahu, W. Ligon, and K. Wichmann. Mission to planet Earth: AI views the world. *IEEE Expert*, to appear.
- [10] N. M. Short, Jr. and S. L. Wattawa. The second generation intelligent user interface for the Crustal Dynamics data information system. *Telematics and Informatics*, 5(3):253–268, 1988.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (<i>Leave blank</i>)		2. REPORT DATE October 1995	3. REPORT TYPE AND DATES COVERED Conference Publication - October 26-27, 1995	
4. TITLE AND SUBTITLE 1995 Science Information Management and Data Compression Workshop			5. FUNDING NUMBERS Code 930	
6. AUTHOR(S) James C. Tilton, Editor				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS (ES) Goddard Space Flight Center Greenbelt, Maryland 20771			8. PERFORMING ORGANIZATION REPORT NUMBER 95B00134	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS (ES) National Aeronautics and Space Administration Washington, DC 20546-0001			10. SPONSORING / MONITORING AGENCY REPORT NUMBER NASA CP-3315	
11. SUPPLEMENTARY NOTES Tilton: Goddard Space Flight Center, Greenbelt, Maryland. This workshop was organized and sponsored by the National Aeronautics and Space Administration (NASA) in cooperation with the Washington/Northern Virginia Chapter of the Institute of Electronics Engineers (IEEE) Geoscience and Remote Sensing Society. Support was received from the Office of Mission to Planet Earth, NASA Headquarters.				
12a. DISTRIBUTION / AVAILABILITY STATEMENT Unclassified - Unlimited Subject Category 59 Availability: NASA CASI (301) 621-0390.			12b. DISTRIBUTION CODE	
13. ABSTRACT (<i>Maximum 200 words</i>) This document is the proceedings from the Science Information Management and Data Compression Workshop," which was held on October 26-27, 1995, at the NASA Goddard Space Flight Center, Greenbelt, Maryland. The Workshop explored promising computational approaches for handling the collection, ingestion, archival and retrieval of large quantities of data in future Earth and space science missions. It consisted of fourteen presentations covering a range of information management and data compression approaches that are being or have been integrated into actual or prototypical Earth or space science data information systems, or that hold promise for such an application. The Workshop was organized by James C. Tilton and Robert F. Crompton of the NASA Goddard Space Flight Center.				
14. SUBJECT TERMS Data Compression, Image Compression, Information Management, Space Science, Earth Science			15. NUMBER OF PAGES 162	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL	

