(NASA-TM-111110) PROGRESSIVE VECTOR QUANTIZATION ON A MASSIVELY PARALLEL SIMD MACHINE WITH APPLICATION TO MULTISPECTRAL IMAGE DATA (Hughes STX) 20 p

N96-11514

Unclas

G3/62 0068498

Accepted for publication in the January 1996 issue of the IEEE Transactions on Image Processing

PROGRESSIVE VECTOR QUANTIZATION ON A MASSIVELY PARALLEL SIMD MACHINE WITH APPLICATION TO MULTISPECTRAL IMAGE DATA

Mareboyana Manohar
Hugh STX Corporation
NASA Goddard Space Flight Center

James C. Tilton
Information Sciences and Technology Branch
NASA Goddard Space Flight Center

Abstract - A progressive Vector Quantization (VQ) compression approach is discussed which decomposes image data into a number of levels using full search VQ. The final level is losslessly compressed, enabling lossless reconstruction. The computational difficulties are addressed by implementation on a massively parallel SIMD machine. We demonstrate progressive VQ on multispectral imagery obtained from the Advanced Very High Resolution Radiometer instrument and other Earth observation image data, and investigate the trade-offs in selecting the number of decomposition levels and codebook training method.

EDICS Category IP 1.1 (Coding).

Dr. M. Manohar, (current address) Computer Science Department, Bowie State University, Bowie, MD 20715. (301) 464-7852 (Voice), (301) 286-1776 (FAX), manohar@chrpalg.gsfc.nasa.gov (Internet). Dr. James C. Tilton, Mail Code 935, NASA GSFC, Greenbelt, MD 20771. (301) 286-9510 (Voice), (301) 286-1776 (FAX), tilton@chrpisis.gsfc.nasa.gov (Internet). Address correspondence to Dr. Tilton.

I. Introduction

The large data volumes from space-borne instruments, such as those being developed for the Earth Observation System (EOS), pose significant data handling problems [1]. We describe here a progressive vector quantization data compression approach that can serve as a basis for an effective data archive and distribution system.

A. Vector Quantization

The basic premise of Vector Quantization (VQ) is to represent vectors by scalar indices. Compression occurs because the scalar indices are smaller than the vectors they represent. We refer the reader to Gray [2] for a detailed description of and rationale for VQ. In the case of two-dimensional image data, VQ vectors are obtained by systematically extracting non-overlapping blocks from the image. Such vectors allow VQ to exploit the two-dimensional spatial correlations in the image data. If the image is multispectral, non-overlapping cubes may be used,

allowing VQ to exploit the 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. A discussion of VQ compression techniques for multispectral images can be found in [3].

VQ requires that a dictionary of representative vectors, called codevectors, be created before an image can be VQ encoded. An image is VQ encoded by replacing each image vector with the address of the closest codevector from this dictionary, commonly called the VQ codebook. The VQ codebook is created through a training process in which an optimal codebook is learned from the input samples using an error criterion such as mean squared error. The most widely used training algorithm is the generalized Lloyd algorithm (GLA) as described in [4]. An alternative approach is based on the self organizing feature map (SOFM) developed by Kohonen [5]. Our specific implementation of SOFM training is described in [6].

Both the training and coding phase of VQ require finding the codevector which is the closest match to a given vector. Computing this closest match requires computations proportional to the size of the codebook, making this step computationally expensive. Computational cost can be reduced by employing suboptimal approaches such as tree-searched VQ (TSVQ) [2]. We prefer to overcome the computational burden by employing a massively parallel implementation of full search VQ on a SIMD (Single Instruction, Multiple Data stream) massively parallel computer, as described in a following section.

While VQ codebook training and data encoding are computationally intensive, data decoding is a table lookup process that can be performed quickly on conventional sequential computers. This makes VQ attractive for the one central encoder and many decoders environment of a data archive and distribution center. In contrast, most transform based compression techniques have nearly the same computational requirements for encoding and decoding.

B. Progressive Compression

Progressive compression is a form of image data compression in which the first part of the compressed data can be used to reconstruct an approximate representation of the image. The

remaining portions of the compressed data contribute increasingly detailed information to the reconstruction of the image until the image is losslessly reconstructed.

In the data archive and distribution setting, one needs to be able to browse through the available data to find the data most appropriate for a given application. With progressive compression, the browse data and archive data can be seamlessly integrated. The browse data (initial progressive rendition of the data) should be an accurate enough representation of the data so that it is possible to make a decision concerning whether or not the data is useful. If the decision is negative, the cost of the decision is just the cost of obtaining the browse data. If the decision is positive, there are two options: i) the user can either get the successive detail data adding to the initial browse data until the data precision is adequate and then terminate the transmission, ii) or the user can get all the remaining data and losslessly reconstruct the data.

C. Progressive Vector Quantization

Two main types of progressive VQ are reported in the literature. One type progressively refines the VQ index, and is usually based on tree-structured VQ. An extension of this idea to full search VQ is given in [7]. The second type uses one or more (multiple) stages or levels of VQ followed by lossless compression of the residual. Our approach is of this latter type. Much of the research on multiple stage VQ, also called residual vector quantization (RVQ), has focused on the joint optimal design of codebooks for each stage [8].

Wang and Goldberg [9] present a progressive image transmission scheme based on RVQ that is very similar to the one we describe here. However, there are some key differences. Wang and Goldberg assume very small codebooks (up to 16 codewords per codebook in their progressive examples), while our codebooks are quite large (the size of the MasPar MP-2, *i.e.*, 16,384 codewords per codebook). Thus, while Wang and Goldberg can send their small codebooks as overhead along with their encoded data, we cannot. Instead, we create appropriate standard codebooks from representative samples of the type of data we expect to encode, and make these codebooks available to the decoder in an off-line process which does not count against our

compression values. In addition, Wang and Goldberg use individualized codebooks for each residual level, while we use a single generic codebook for all residual levels.

II. Implementation of Progressive VQ

Our implementation of progressive VQ is shown in block schematic form in Figure 1. The input image data is first VQ encoded and the VQ indices are losslessly compressed, producing output stream q_1. This first level VQ encoding is then decoded and subtracted from the input image to produce the first residual image which is then subject to the same process performed on the input image data, producing output stream q_2. The VQ encoded first residual is then decoded and differenced with the original first residual to produce a second residual. This process can be repeated any number of times until the last (n-1th) residual is formed. In the last (n^{th}) stage, this last residual is *not* VQ encoded, but rather is losslessly compressed to produce output stream q_n.

The input image data is VQ encoded with a codebook designed to produce a relatively high compression ratio (*e.g.*, compression ratios of 30 to 50). In the second level of compression, the first residual image is VQ coded with a different codebook than used for the first level. A single codebook is used to code residual data at ensuing levels making it very easy to perform any number of levels of decomposition.

The two codebooks (one for the input image data, and the other for all levels of residual images) are generated using a training set which does not include the images that are being compressed. In the cases we have tested, the successive residual images from this decomposition have monotonically decreasing entropy, and appear to possess the stationarity properties that would make it possible for a single codebook to be adequate for all residual images. The final residual image and the VQ encoding indices from each level are losslessly compressed using either the Rice algorithm [10] or the Ziv-Lempel algorithm [11].

The progressive VQ decoding is exactly the reverse process (see Figure 2). The codebooks are assumed to be available at the receiving end. The first level browse image, I_1, is obtained by a lossless decompression and a VQ table look-up decoding of the first data received from

progressive the VQ scheme. Refinements can be obtained by decoding the successive levels of compressed residual image data and adding the result to the reconstructed image from the previous level. The decoding for the last level consists only of a lossless decompress. When all the levels of the progressive VQ are processed, the image is losslessly reconstructed as I_n.

III. Implementation of Full Search VQ on a SIMD Machine

The training and coding phases of full search VQ are computationally intensive. We have chosen to address the computational problem by implementing full search VQ on a Single Instruction, Multiple Data-Stream (SIMD) multiprocessor, the MasPar model MP-2.

The MasPar MP-2 at the NASA Goddard Space Flight Center has 16,384 4-bit Processing Elements (PEs), with 64 Kbytes of local memory at each PE. As shown in Figure 3, the input image (in this case, a multispectral image) is vectorized such that spatially adjacent pixels in contiguous spectral channels form the elements of a vector. In this example, a 4x4x2 data cube is extracted as a vector in which the first 16 elements of the vector come from the 4x4 window in the first spectral band, and next 16 elements are from the 4x4 window in the second spectral band.

The codebook (Y_1, Y_2, ..., Y_n) is stored in the local PE memory as shown in the Figure 3. The image is vectorized and each vector (X_j) is broadcast throughout the PE local memory. The Euclidean distances between each input vector and the code vectors are computed simultaneously by all the processors and the result is stored in a distance array (D_1, D_2 , ..., D_n). The time taken for computation of this distance array is proportional to the size of the vector in pixels. The closest match can be then found by finding the least value in the distance array. This can be done in a time equal to the product of number of bits per one element of distance array (*i.e.*, 32 - the number of bits for a real number) and the clock cycle of the MasPar. The vector in the codebook corresponding to the location of this minimum distance value is the closest match. In training, this vector (plus a set of neighboring vectors in case of SOFM) is updated. For coding, the address of the closest match is the code index, and is written to the

output file. This process is then repeated for all the vectors from the input image, forming one iteration of training or a complete coding of the input image.

For optimal performance, the codebook size must be equal to the size of the PE array. However, codebooks smaller or larger than PE array size can be easily handled. If the codebook size is smaller than the PE array size by 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 codevector can be split up into two smaller codevectors, each containing half the number of elements and loaded into the local memories of adjacent PEs. The input vector is similarly split into two smaller vectors, and propagated such that the first vector is propagated to PEs with even number address and the second one to PEs with odd number address. 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 using processor virtualization.

Our MasPar MP-2 implementation of progressive VQ includes options for generating codebooks using either GLA or SOFM algorithms, modules for coding and decoding image data, and several other utilities for combining browse with residual data for progressive build up.

IV. Progressive VQ Performance and Tradeoffs

GAC (Global Area Coverage) data from the Advanced Very High Resolution Radiometer (AVHRR) instrument was used in the first test of our progressive VQ implementation. The AVHRR test data consists 409-by-512 pixel sections of data with five spectral bands; the first two bands are shown in Figure 4a. (All images are non-linearly stretched for clearer visual display.) While the data are stored as 16 bit pixels, the radiometric resolution of the data is actually 10 bits/pixel. Two pairs of codebooks were generated using GLA and SOFM algorithms. The training set consists of AVHRR image data other than that used in the test.

In the example, the multispectral AVHRR image data was decomposed into three levels. Figure 4(b-f) shows the compression results for GLA training. Figure 4b is a VQ coded browse

quality image with compression ratio of 30.7† (0.52 bits/pixel) and MSE of 627.7. In Figure 4c, a VQ coded (compressed to approx. 30) and reconstructed first order error residual is shown. When these images are added the resulting refined image (Figure 4d) has a compression ratio of 15.4 (rate of 1.04 bits/pixel) and MSE of 229.9. In this three level decomposition, the lossless image data is obtained by adding the second order residual shown in Figure 4e, to obtain the final losslessly reconstructed image (Figure 4f). The image data has been decomposed to browse quality VQ coding, and first order residual VQ coded data and final error residual. Each of these were compressed further with the Ziv-Lempel compression algorithm. Similar results are shown in Figure 5 (a-d) using the SOFM algorithm.

As the number of levels increases, a better distortion performance can be obtained as shown in Figure 6 for both GLA and SOFM algorithm in the lossy mode. Similarly, the compression performance of these algorithms as a function of number of levels is given in Figure 7. The first level compression ratio of with SOFM training is better than that of GLA training for comparable vector and codebook sizes. The reason for this is that SOFM maps the vectors (points in k-dimensional Euclidean space) derived from the input image onto a codebook such that the vectors that are close in k-dimensional space are close in the index range of the codebook. This ensures correlations in the vector space are preserved in the index range, thus providing a good lossless compression on the address bits. However, the MSE performance of the SOFM is not as good as that of GLA. In our experiments we have found in more than one example the rate-distortion performance of the GLA is better than that obtained with SOFM. The rate distortion performance of the GLA and for SOFM are shown in Figure 8. We have also seen that SOFM training requires many more iterations than the GLA to get similar results.

Lossless progressive VQ decomposition can be obtained using any number of levels. In each case, last residual is losslessly compressed. The lossless compression ratio that can be obtained from such decompositions depends greatly on how well the given lossless compression technique performs on the last residual. We have seen that the Rice algorithm compresses the

† The compression ratios quoted here are measured against 16 bit/pixel - the way in which the data is normally stored. To obtain the compression ratio versus the 10 bit/pixel radiometric resolution of the data, multiply by 0.625.

image data better than the Ziv-Lempel technique. However, the performance of these techniques on the address bits of VQ quantized levels are not significantly different. In Figure 9, the lossless performance of progressive VQ using the GLA is shown with Ziv-Lempel and Rice lossless compression, and the results are compared with entropy for 8 levels of decomposition. The lossless performance of progressive VQ using SOFM is shown in Figure 10. Figures 9 and 10 show that the Rice algorithm yields compression ratios close to the rates based on first order entropy. As regards to number of decomposition levels, the best compression ratio can be obtained for 5 level decomposition with GLA/Ziv-Lempel, whereas just 2 levels of decomposition give the best compression for GLA/Rice. Similar results are obtained for SOFM/Ziv-Lempel and SOFM/Rice (see Figure 10).

The lossless compression ratio for level 0 in Figures 9 and 10 corresponds to straight application of Ziv-Lempel or Rice compression on the image data. From Figure 9, the level 0 lossless compression ratio for the GLA/Ziv-Lempel combination is 1.44 (rate = bits per pixel/CR = 16/1.44 = 10.83). Using 2 levels of decomposition the compression ratio is improved to 1.82 (rate = 8.79). Similarly, for the GLA/Rice combination, the level 0 lossless compression ratio is improved from 2.1 to 2.31 using 2 levels of decomposition (rates 7.62 and 6.93, respectively in Figure 9). Similar improvements with number of levels can be seen for SOFM based lossless decomposition in Figure 10.

With increasing number of decomposition levels we see an increase in the compression performance until an optimum point is reached and the compression performance starts to decrease. The number of levels that gives the best compression depends on the type of lossless compression used. In this case, the best performance with Ziv-Lempel occurs at 5 levels, while the best performance with Rice occurs at 2 levels. We note, however, that as the number of levels increase the reconstruction costs are higher. Therefore, the optimum number of levels of decomposition is a tradeoff between compression ratio and computational burdens in reconstructing the data. We see that after certain number of decomposition levels, the computational burden outweighs the compression savings. We have found that 2 to 3 level

decomposition is usually adequate for progressive refinement and best compression without excessive computational burden.

Progressive VQ was also tested on a 512-by-512 pixel section of bands 2 and 3 of Thematic Mapper (TM) image of Washington, DC. Two level progressive VQ improved the overall lossless compression ratio from 2.05 to 2.35 compared to Lempel-Ziv compression and one level progressive VQ improved the overall lossless compression ratio slightly from 2.31 to 2.34 compared to Rice compression. Similar results were also obtained using a subset of Coastal Zone Color Scanner (CZCS) data.

This correspondence described a progressive VQ compression approach which is implemented on a massively parallel machine, on which training and coding can be performed in reasonable processing times. Reconstruction is a table look-up process that can be easily performed on a sequential machine. The performances of the two training techniques, namely GLA and SOFM, differ marginally, with GLA scoring slightly higher on the test data. In both cases, two level progressive VQ produced a higher or comparable overall lossless compression than a non-progressive application of lossless compression. The advantages of progressive transmission combined with this improvement in compression outweigh the marginal computational problems associated with reconstructing the data from progressive VQ.

REFERENCES

[1] J. C. Tilton, M. Manohar and J. A. Newcomer, "Earth science data compression issues and activities," *Remote Sensing Reviews*, Vol. 9, pp. 271-298, 1994.

[2] R. M. Gray, "Vector quantization", *IEEE ASSP Magazine*, pp. 4-28, April 1984.

[3] S. Gupta and A. Gersho, "Feature predictive vector quantization of multispectral images," *IEEE Trans. on Geoscience and Remote Sensing*, Vol. 30, No. 3, pp. 491-501, May 1992.

[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, Jan. 1980.

[5] T. Kohonen, "The self-organizing map," *Proc. of the IEEE*, Vol. 78, pp. 1464-1480, 1990.

[6] M. Manohar and J. C. Tilton, "Compression of remotely sensed images using self organizing feature maps," in *Neural Networks for Human and Machine Perception*, H. Wechsler, ed., Academic Press, In., San Diego, CA, 1991.

[7] E. A. Riskin, R. Ladner, R.-Y. Wang, and L. E. Atlas, "Index assignment for progressive transmission of full search vector quantization," submitted to *IEEE Trans. on Image Processing.*

[8] C. F. Barnes and R. L. Frost, "Vector quantizers with direct sum codebooks," *IEEE Trans. on Information Theory*, Vol. 2, No. 2, pp. 565-580, March 1993.

[9] L. Wang and M. Goldberg, "Lossless progressive image transmission by residual error vector quantization", *IEE Proceedings*, Vol. 135, part F, pp. 421-430, 1988.

[10] R. F. Rice, P.-S. Yeh and W. Miller, "Algorithms for a very high speed universal noiseless coding module," *Jet Propulsion Laboratory Publication 91-1*, Feb. 15, 1991.

[11] J. Ziv and A. Lempel, "A universal algorithm for sequential data compression, " *IEEE Trans. Information Theory*, Vol. IT-23, pp. 337-343, 1977.

Figure Captions

Figure 1. Block schematic of progressive vector quantization (VQ) encoding. The same codebook (codebook2) is used for encoding all residual images. The browse level approximation is q_1, while q_2, ..., q_n are quantization residuals.

Figure 2. Block schematic of progressive VQ decoding. I_1 is the browse level reconstruction, I_2,... I_{n-1} are intermediate level reconstructions, and I_n is the lossless reconstruction.

Figure 3. Vector quantization of multispectral data on a SIMD (single instruction, multiple data) massively parallel machine.

Figure 4. Results of 2 level progressive VQ on bands 1 and 2 of the AVHRR test data set using the GLA for codebook generation. a) Band 1 and 2 AVHRR data. b) Browse quality (CR=30). c) Vector quantized residual. d) Refined image obtained by adding images in b and c. e) The final residual (which is losslessly compressed). f) Losslessly decompressed AVHRR data.

Figure 5. Results of 2 level progressive VQ on bands 1 and 2 of the AVHRR test data set using the SOFM algorithm for codebook generation. a) Band 1 and 2 AVHRR data. b) Browse quality (CR=40). c) Vector quantized residual. d) Refined image obtained by adding images in b and c. e) The final residual (which is losslessly compressed). f) Losslessly decompressed AVHRR data.

Figure 6. Distortion as a function of the number of levels of progressive VQ (prior to the final lossless step).

Figure 7. Compression performance as a function of the number of levels of progressive VQ (prior to the final lossless step).

Figure 8. Rate distortion for progressive VQ (prior to the final lossless step) for both the GLA and SOFM codebook training algorithms.

Figure 9. Lossless compression performance in bits/pixel of progressive VQ with GLA training as a function of number of levels. Two lossless compression techniques (Ziv-Lempel and Rice) are compared with the entropy rate.

Figure 10. Lossless compression performance in bits/pixel of progressive VQ with SOFM training as a function of number of levels. Two lossless compression techniques (Ziv-Lempel and Rice) are compared with the entropy rate.

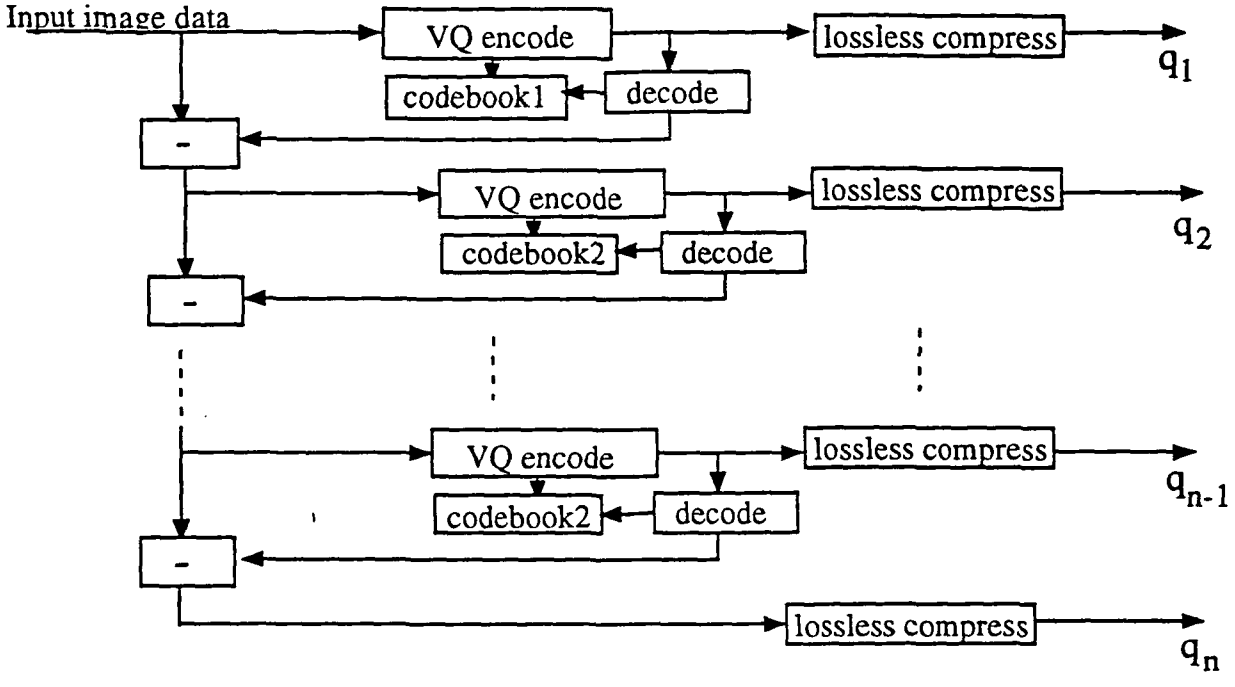

Figure 1

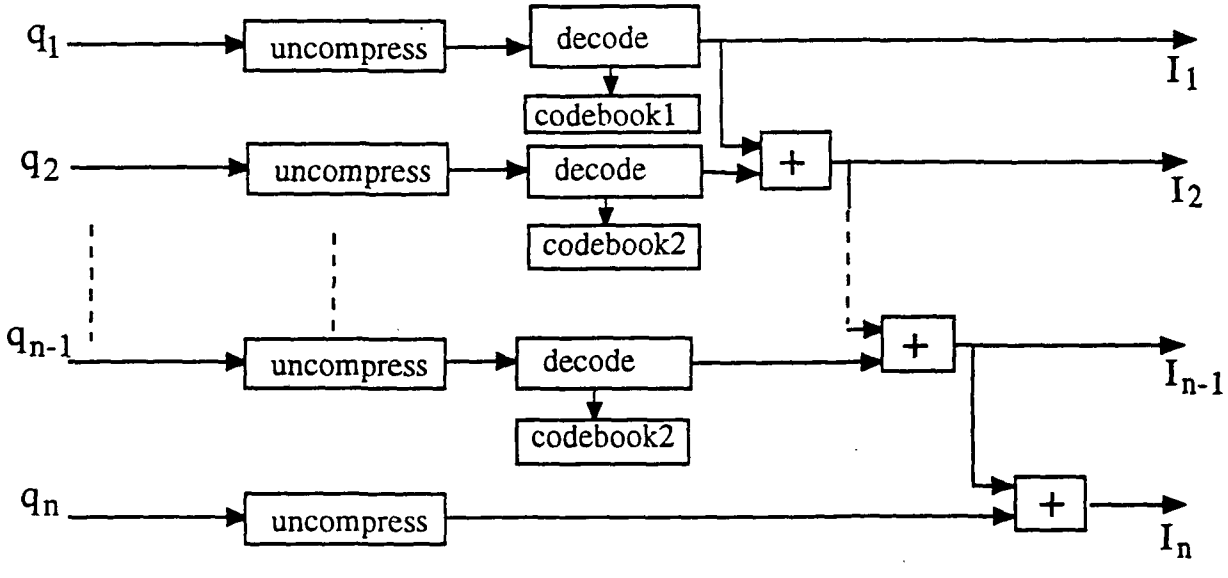

Figure 2

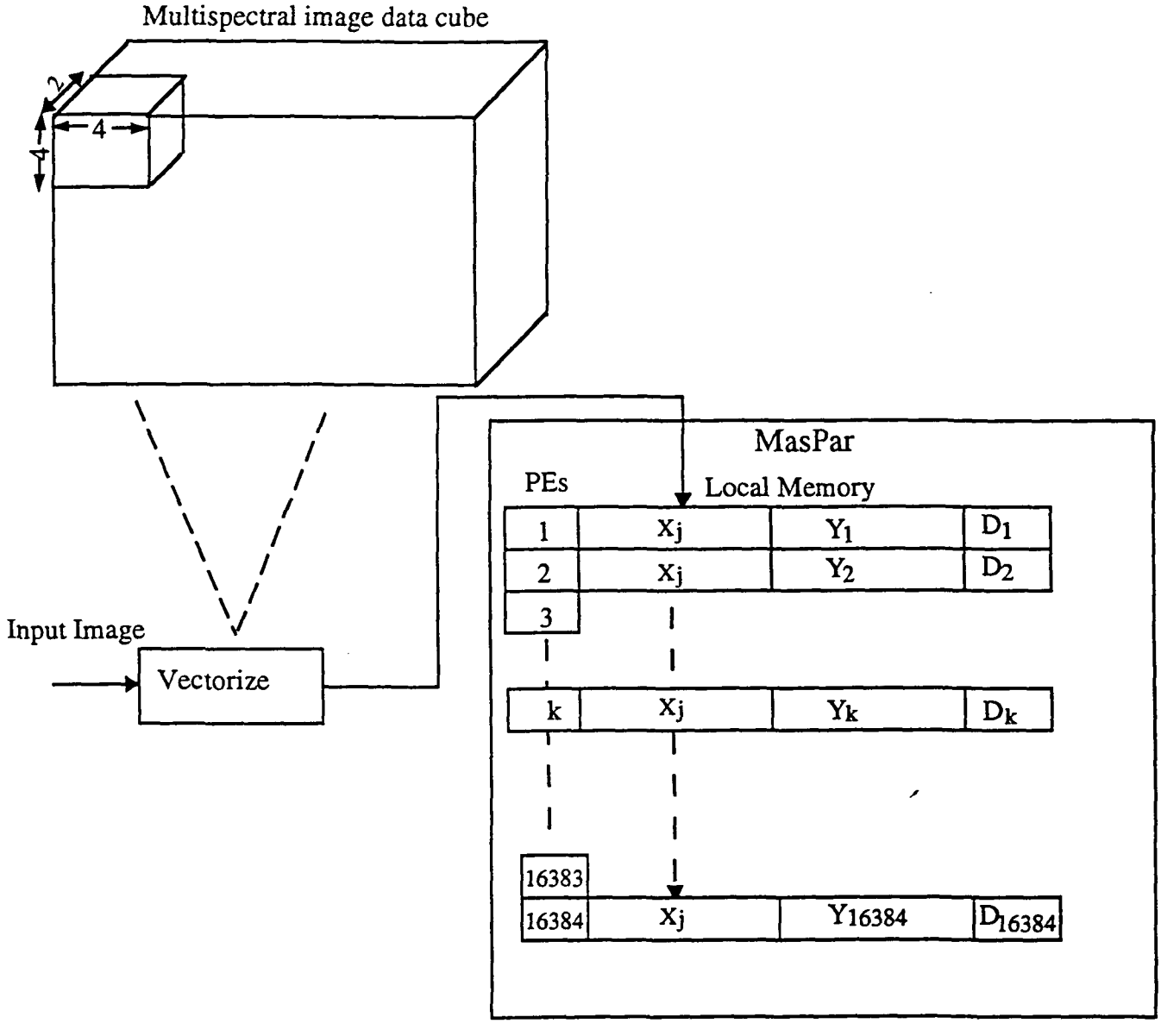
Multispectral image data cube
4
4
Input Image
Vectorize
MasPar
PEs
Local Memory
1
2
3
k
16383
16384
Xj
Xj
Xj
Xj
Y1
Y2
Yk
Y16384
D1
D2
Dk
D16384

Figure 3

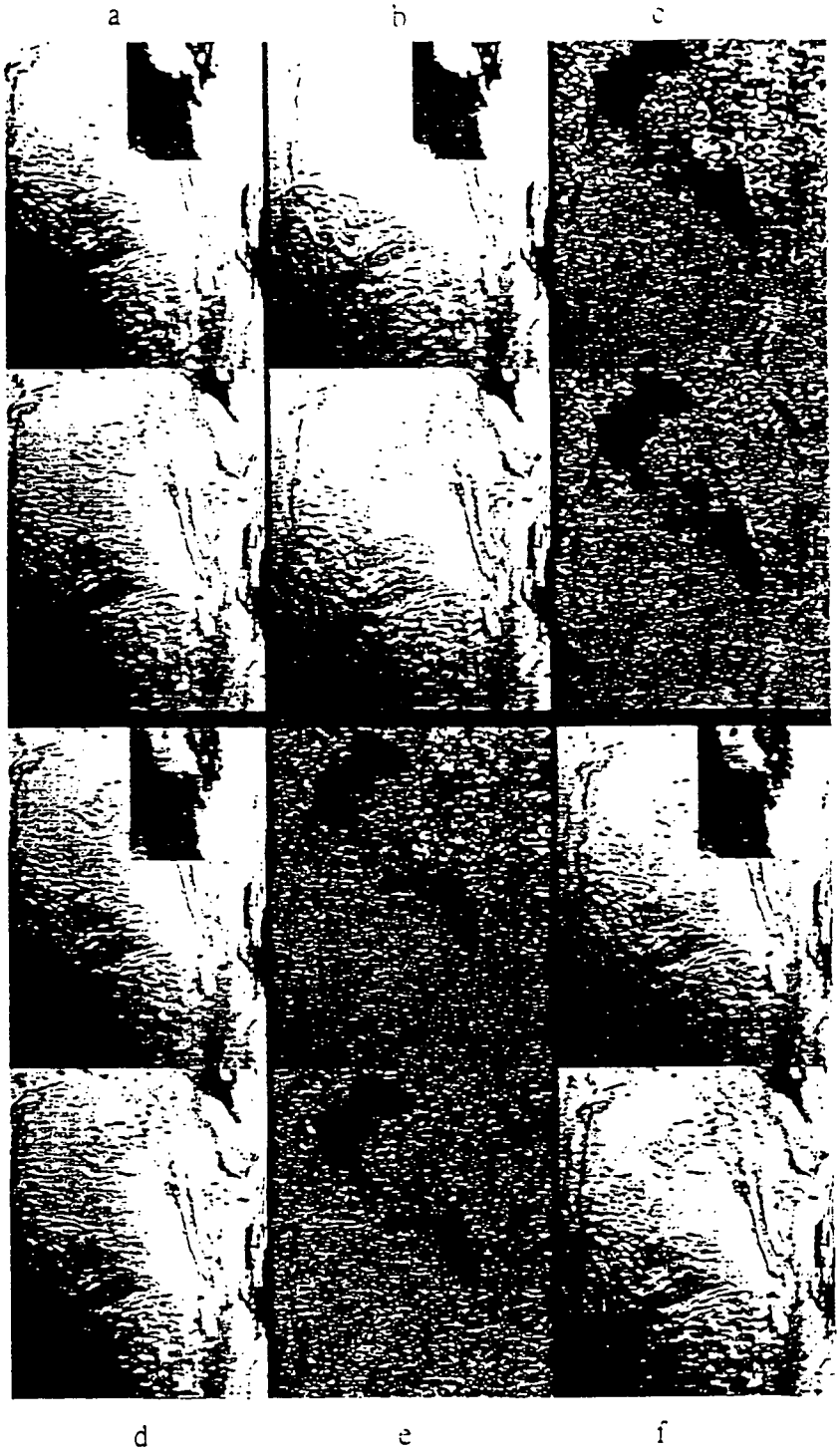

Figure 4

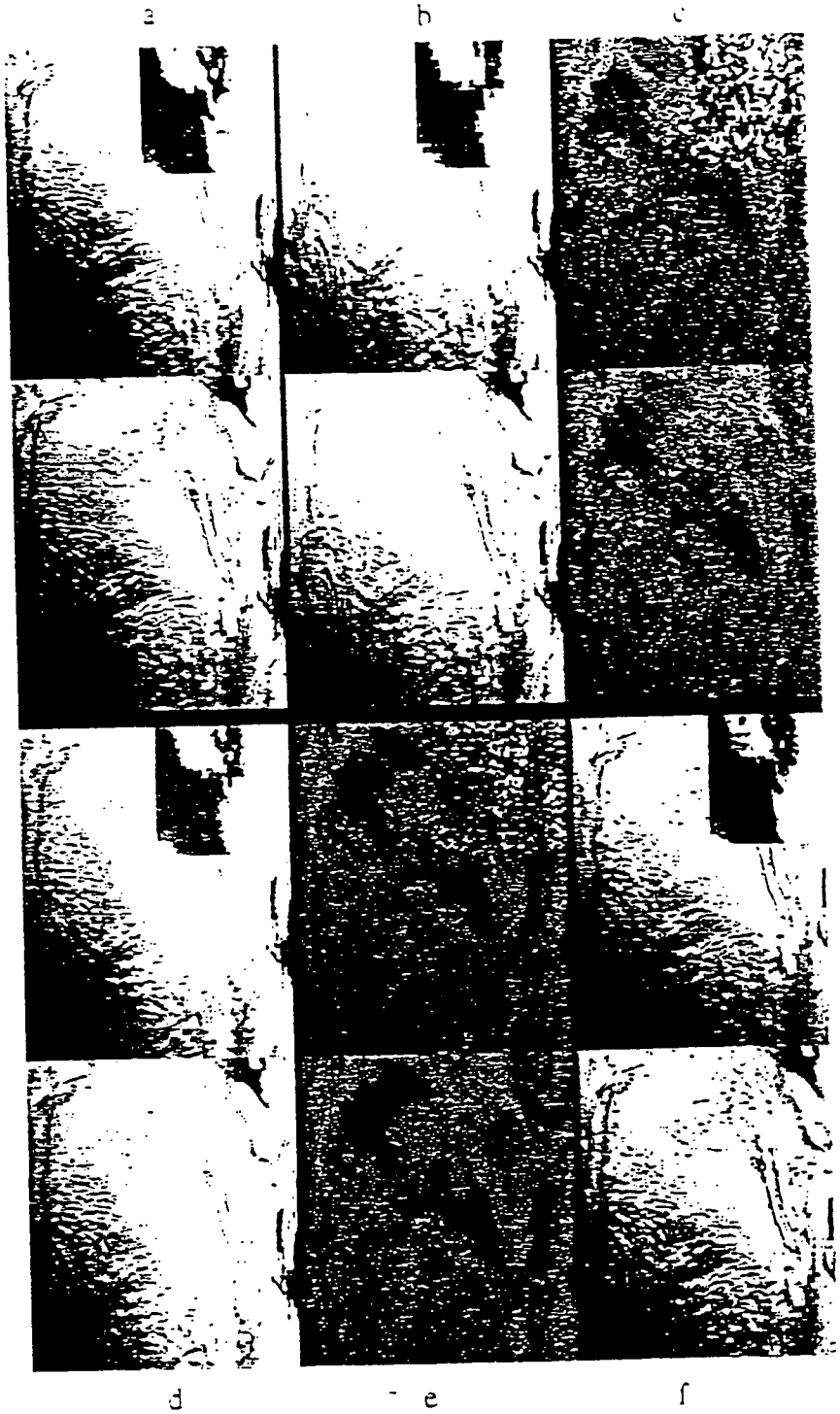

Figure 5

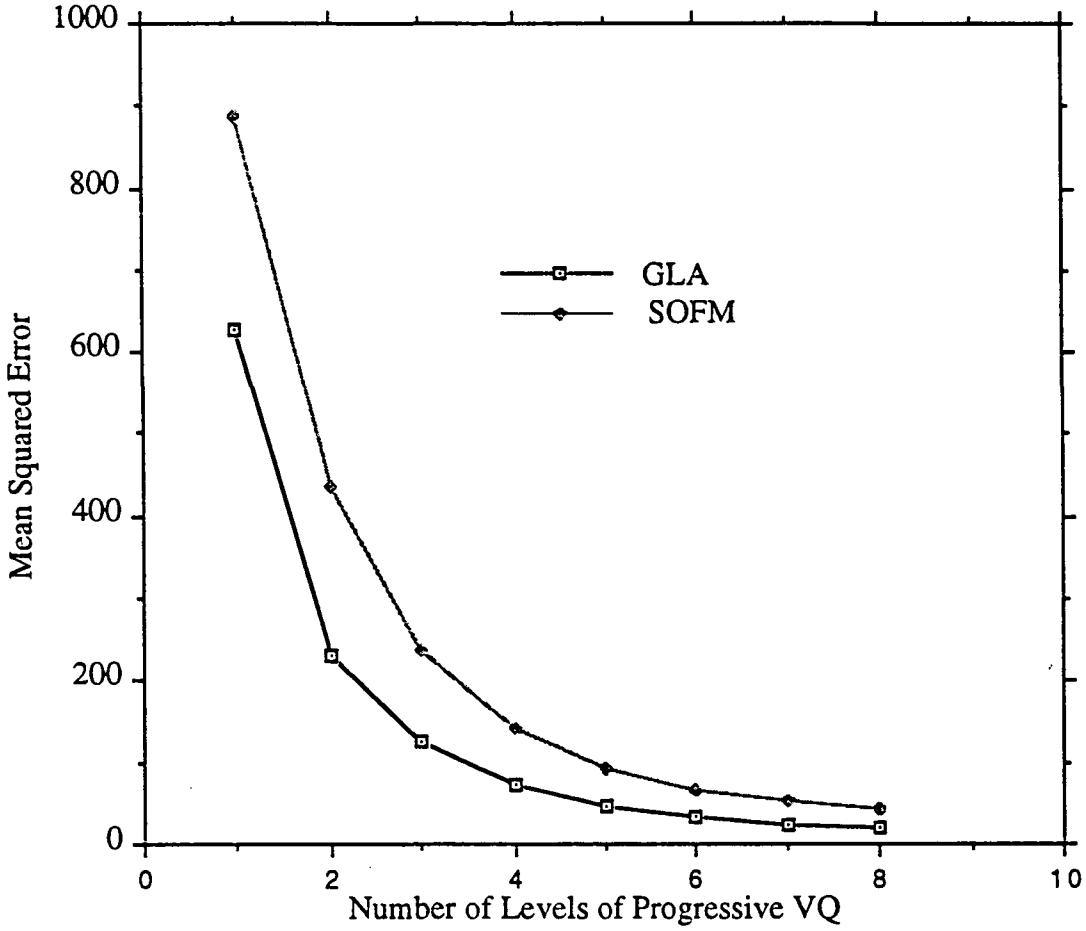
1000
800
600
400
200
0
Mean Squared Error
GLA
SOFM
0
2
4
6
8
10
Number of Levels of Progressive VQ

Figure 6

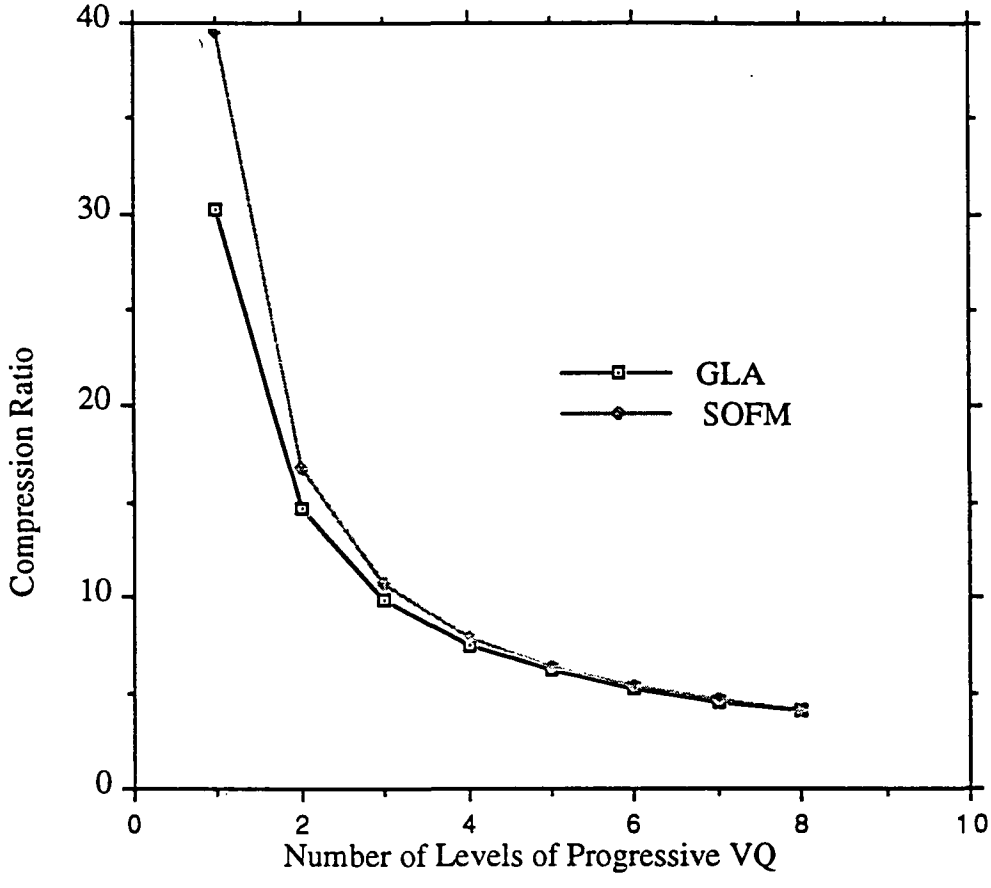
40
30
20
10
0
Compression Ratio
GLA
SOFM
0
2
4
6
8
10
Number of Levels of Progressive VQ

Figure 7

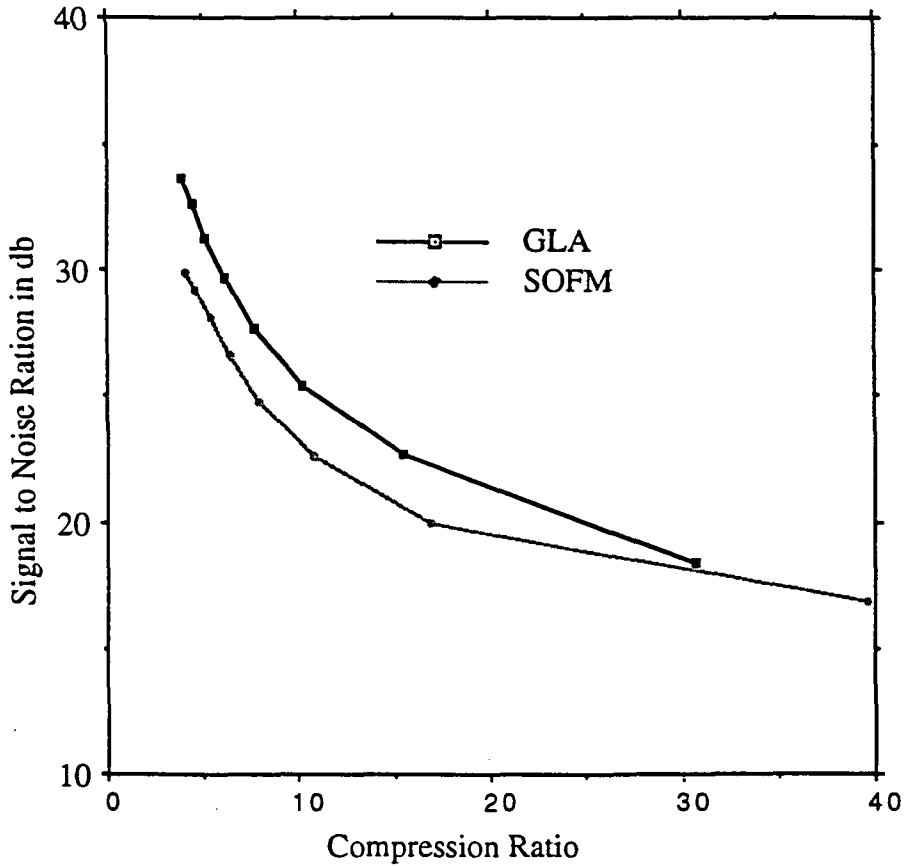
40
30
20
10
0
10
20
30
40
Signal to Noise Ration in db
Compression Ratio
GLA
SOFM

Figure 8

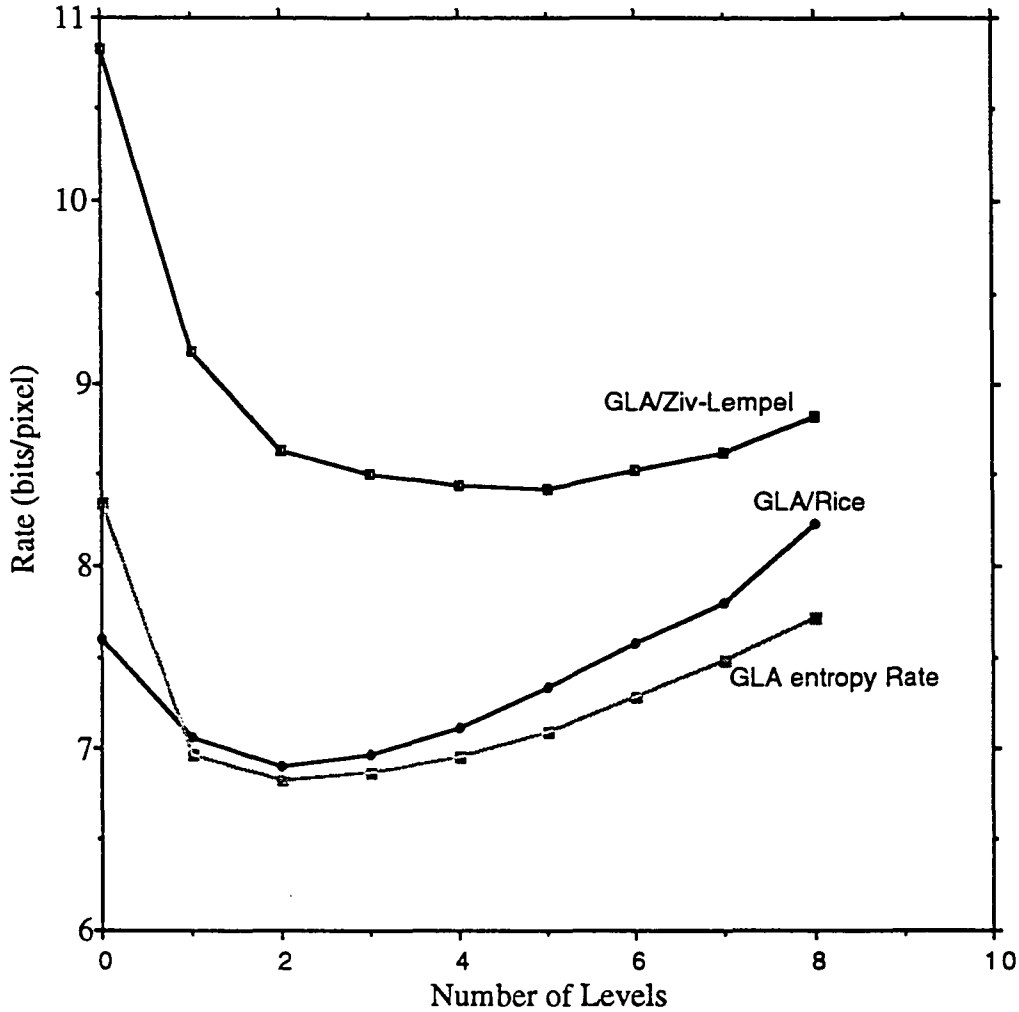
11
10
9
8
7
6
Rate (bits/pixel)
0
2
4
6
8
10
Number of Levels
GLA/Ziv-Lempel
GLA/Rice
GLA entropy Rate

Figure 9

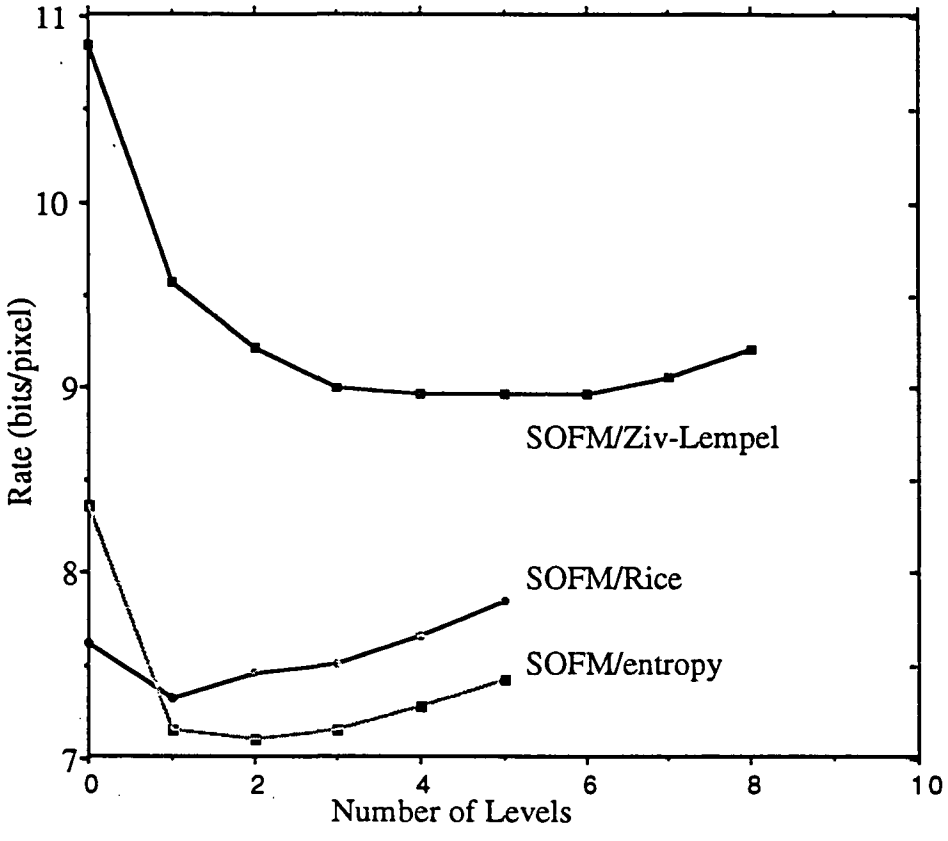
11
10
9
8
7
Rate (bits/pixel)
0
2
4
6
8
10
Number of Levels
SOFM/Ziv-Lempel
SOFM/Rice
SOFM/entropy

Figure 10