A-99

Reprinted from

IMAGE COMMUNICATION

Signal Processing: *Image Communication* 14 (1999) 245–267

A strategy for satellite data archival. Low noise variable-rate vector quantization with application to AVHRR satellite images: A tutorial review

K.S. Thyagarajan[a,*], G. Bendak[a], E.R. Boer[b], V. Ramanathan[b]

[a] *Department of Electrical and Computer Engineering, San Diego State University, San Diego, CA 92182, USA*
[b] *Scripps Institute of Oceanography, University of California, San Diego, CA, USA*

Received 24 April 1997

ELSEVIER

ELSEVIER

Signal Processing: *Image Communication* 14 (1999) 245–267

SIGNAL PROCESSING: IMAGE COMMUNICATION

A strategy for satellite data archival. Low noise variable-rate vector quantization with application to AVHRR satellite images: A tutorial review

K.S. Thyagarajan[a,*], G. Bendak[a], E.R. Boer[b], V. Ramanathan[b]

[a] *Department of Electrical and Computer Engineering, San Diego State University, San Diego, CA 92182, USA*
[b] *Scripps Institute of Oceanography, University of California, San Diego, CA, USA*

Received 24 April 1997

Abstract

The volume of satellite data amassed by modern day weather and climate satellites is so enormous that it has become virtually impossible for researchers to access the original resolution data collected by the satellites. Typically, researchers are forced to deal with lower resolution reduced data (e.g. cloud cover or temperatures) and often at a resolution degraded by one to three orders of magnitude when compared with the resolution of the original data. The uncertainties in the reduced data are often unknown. This state of affairs will only get worse, since the data to be collected under the guidance of NASA's global change program may increase by several orders of magnitude in the coming decades.

We need to experiment with mathematically rigorous ways to tame the original data without significantly degrading the information content. Compression of original data by objective mathematical techniques is a promising approach. This study adopts recent compression techniques developed in the field of communications and applies them to weather satellite data. Typically, these techniques compress the raw data by factors ranging from 10 to 100. The compressed data can be decompressed to retrieve the near original data at the site of the user. The mean error in the compression–decompression process varies from a few percent to several percent. As a demonstration, we consider the advanced very high resolution radiometer (AVHRR) radiances with a nadir resolution of 1 km × 1 km. For this demonstration, we adopt a well understood compression technique which is the so-called vector quantization technique. Several vector quantization techniques (full-search, tree-search, pruned-balanced-tree, greedy-tree and pruned-greedy-tree) are compared in performance and ease of implementation. The discussion focuses on the pruned greedy tree-structured vector quantizer because it is highly suited to the compression of AVHRR satellite images. For the case considered here, visual and scientific reproducibility of the original high resolution images are very good. The rms error for roughly 95% of the pixels in a scene is within 2%, even at a 32:1 compression ratio. The error in spatially averaged fields is less than 0.1% for averaging scales in excess of 50 km × 50 km. Some important spatial structural information is lost, however. It is found that the same image when compressed using JPEG standard shows significant loss of numerical accuracy at the same compression ratio of 32:1. But improvements and developments in compression techniques can minimize these errors

* Corresponding author.

PII: S0923-5965(98)00012-5

and afford researchers the luxury of storing and working with high resolution data at roughly at about 0.03 of the space required by the original data.

1. Introduction

NASA is currently embarking on the ambitious earth observing system program which will make an unprecedented attempt to obtain simultaneous high resolution data from the physical, chemical and biological state of the land–ocean–atmosphere system [27]. The resulting data rate will exceed, by several orders of magnitude, the current data rate from instruments such as the Advanced Very High-Resolution Radiometer (AVHRR) aboard NOAA's polar orbiting satellites.

The following example illustrates the enormous magnitude of the data volume. Per day, the 5 channel AVHRR at full resolution (1 km × 1 km at nadir) generates about 1 billion samples (radiances) stored in 2 billion bytes. Typically, to answer a simple question concerning the response of clouds to the warming trend of the 1980s, we need to analyze at least 15 years of data. Because of the formidable nature of the problem, the global AVHRR data are not archived at full resolution. The archived data set is a sampled and averaged data with a spatial resolution coarser than the original by more than an order of magnitude. The volume of even this sampled data sets are too large for most of the academic research community, which resorts to using the reduced products, such as cloud fraction averaged over a 100 km × 100 km region. The full resolution data is either permanently lost or mostly inaccessible to the community at large. The problem with sampled or averaged data sets is the loss of spatial structure, which could prove to be a fatal loss for problems related to the spatial structure of clouds [2].

The scientific community faces the challenging task of taming this vast jungle of data. It is not clear how the community will face this challenge. Most likely, the data will be severely underutilized, as is the case with most of modern day satellite data. By under-utilization, we mean that the community will access only sampled or averaged data sets. For example, even in the highly successful earth radiation budget experiment (ERBE) data, only a percent or less of the total data (the so-called standard data products) has been analyzed or published. There are two distinctively different types of issues. First, in the case of high resolution data, lack of adequate satellite onboard data storage facilities necessitates sub-sampling or averaging, to reduce data flow rate. Second, a majority of research community lacks adequate computer and storage facilities to attack even the sampled data set.

It is conceivable that by the middle of the next century, improvements in computer technology will make it feasible for a majority of the research community to handle high resolution data. Clearly, to prevent further loss of valuable global data, we need to develop strategies to preserve the information content of the data without overstressing the storage capability. Compression of data by mathematical techniques, without significant loss of information, is one of the attractive alternatives to this dilemma. The purpose of this work is to explore whether compression techniques developed by the information theory researchers can meet the challenge facing the global change community.

For the purpose of this exploratory work, we adopt a state-of-the-art and well documented technique, namely, vector quantization methods (VQM) as a means of data compression. We apply VQM to AVHRR data collected by NOAA polar orbiters (weather satellites). The AVHRR image consists of two visible, one near-infrared and two far-infrared (so-called window) channels. The visible channels record the reflected solar radiance from the surface–atmosphere column and the infrared channels record the radiation emitted to space by the earth–atmosphere system.

2. What is image compression?

Digital image compression refers to any process or algorithm by which the amount of computer memory required to store the original raw data is

reduced. The amount of data reduction depends on the compression algorithm and the types of images under consideration. Existing compression techniques fall into two broad categories: lossy and lossless. The lossy techniques, by definition, introduce an amount of degradation which depends on compression ratio, whereas lossless techniques show no degradation and typically can achieve a compression ratio of 2:1. Such a low ratio is not sufficient for the type of images considered in this work and is therefore not discussed. For lossy techniques, image reproducibility (quality of the decompressed image) becomes, aside from compression ratio, a major factor in algorithm selection. During the course of this article, quality, unless specified otherwise, refers to visual reproducibility of the images in terms of sharpness, texture, edges, etc., as well as the numerical reproducibility in terms of statistical parameters and other performance criteria which will be discussed later on. Recently, some works have been reported in the area of lossy compression techniques for satellite images [15,21] with high compression ratios.

Existing lossy techniques fall into the classes of predictive coding and transform coding. The word coding is synonymous with compression. Predictive coding takes advantage of the dependency of a sample on neighboring pixels to compress the data. The determination of the optimal predictor coefficients can be achieved using efficient recursive techniques based on a set of images. In this case the predictor is fixed. Once a predictor is designed the compression consists of two stages: (1) a predicted image is created by replacing the value of each sample by the value predicted from previous samples, and (2) an error image is then formed as the difference between the actual and predicted image. Instead of storing the original image, the first sample in the traversal path is stored along with the parameters that define the predictor, and the quantized error image. The idea is that prediction tries to remove redundancy in the image data. Thus, the differential or error image has very little redundancy which is exhibited in its probability density function (pdf). The model for the pdf of the error image is typically Laplacian, as shown in Fig. 3 with much smaller variance than the image itself. It is easy to show that the average bit rate reduction due to prediction is logarithmically related to the ratio of variances of the original and error images. Hence compression is obtained. At the receiver the reverse process is applied, the predicted image is generated from the first traversal point and the received predictor error image is added to obtain the decompressed image. Acceptable compression ratios achievable with this method are around 3:1. Even though implementation is very simple, its application is limited to certain types of images [10,12]. Using adaptive predictive coders or compressors, higher compression can be achieved, however, at significantly increased complexity [10,12].

Transform coding, another lossy technique, reduces the redundancy of neighboring pixels by transforming a block of pixels into a block of coefficients such that the energy in the original block is packed into as few transform coefficients as possible. Then, only the coefficients with significant variances need to be quantized and stored, thus achieving compression. The Karhunen–Loeve transform (KLT) has been shown to be the optimum transform [10,22]. However, the KLT is computationally intensive and image dependent. Hence in practice, other suboptimal transforms such as the discrete cosine transform (DCT) are used which are machine efficient [6]. In practice, no single coding technique achieves the required high compression ratio and good quality. In fact, a combination of both lossless and lossy techniques are used to achieve as much compression as possible, example, the JPEG standard. Since transform coding is a memoryless scheme, unlike the above predictive coding, the noticeable degradation is usually in the form of blockiness in the reconstructed image. Transform coding techniques result in much higher compression than the predictive compression techniques for the same loss in information.

In the compression methods mentioned above, the image of interest is quantized using scalar quantizers (quantize one sample at a time). Quantization refers to a process of converting a signal value of arbitrary accuracy into a finite one from an allowed set of values. For example, if the allowed number of values equals 256, then quantization of a signal

value will result in an 8 bit number. The quantizers depend on the probability distribution of the input image for optimality, and their design is based on Lloyd's iterative algorithm which tries to minimize the overall mean square error (MSE) between the input and quantized image [18]. Tables of optimal decision and reconstruction levels can be found in [12,11]. Quantization is achieved by replacing a sample value with a reconstruction code that is dictated by the decision level closest to the sample value. According to Shannon, a quantizer operating on a block or vector of consecutive samples will always outperform a scalar quantizer in terms of signal power to noise power ratio (SNR), for the same bit rate [28]. This has given a large impetus to the development of efficient block or vector quantizer design algorithms [7]. Moreover, as will be shown, the vector quantizer (VQ) has a simple decoder (decoder is the same as the decompressor) that requires no computation.

For these reasons, we focus on vector quantization as a means to compress digital images. The next section gives a brief description of fixed-rate vector quantization methods followed by Section 4 which outlines a variable-rate, tree-structured vector quantizer and describes how to generate a codebook. Sections 5–7 deal with the issues in VQ codebook design and the criteria for choosing the right type of image database for training the VQs. Application of these VQ methods to AVHRR images will be discussed in Section 8 followed by concluding remarks in Section 9.

3. Fixed rate vector quantization

In this section we discuss the application of vector quantization to 2-dimensional images (data sets). The original image X is first decomposed into rectangular blocks of L samples, say N total. An image vector X_n, $(X = \{X_n | n = 1, \ldots, N\})$, is formed by ordering the samples of a block sequentially. The vector quantizer Q maps these L-dimensional image vectors (elements of Euclidean space R^L) into a finite set C^L of M code vectors. Thus,

$Q: R^L \rightarrow C^L$,

where $C = \{Y_m | m = 1, 2, \ldots, M\}$. A codevector or codeword is a predetermined vector that is representative of some part of the image to be compressed. Vector quantization (encoding) is the process of assigning a codeword Y_m to each image vector X_n. Instead of storing the vector X_n, the index m of the 'best' matching codeword (will be defined later) is stored. In terms of storage requirements, each vector to be compressed is replaced by an index (an integer) of $\log_2 M$ bits corresponding to the best code vector, or equivalently, $(1/L)\log_2 M$ bits per pixel. For example, a VQ with a codebook (a codebook refers to a predetermined table of representative vectors) containing 256 16-dimensional code vectors ($M = 256$, $L = 16$) requires 8 bit numbers ($8 = \log_2 256$) to index a codeword, which translates to a storage requirement of 0.5 bit per original image sample, and a file size compression ratio of $I/0.5$, where I is the number of bits per sample in X, the original image (I is 16 for our AVHRR images). But to obtain the actual data compression ratio, we need to use $I = 10$, because the data has a dynamic range of only 10 bits. Since each block of the original image is now stored with the same number of bits in the compressed image, the quantizer or compressor is called a fixed-rate quantizer. One might be tempted to conclude that larger image and code vector dimensions would increase compression, however, the visual quality of the reconstructed image will be poorer because of blockiness and edge degradation [8,29]. A remedy is to increase the codebook size M, but on the other hand, this increases the storage requirements and computational load. This fundamental trade-off between compression ratio and image reproducibility will be addressed. Once the codebook is extracted from a carefully selected set of training images, image decompression (decoding) is simply the process of replacing the indices in the compressed image with the corresponding codewords.

The VQ as described above is called a full-search VQ because it has to search the entire codebook to find the 'best' matching code vector [9], whereas other tree-structured VQs allow for more efficient codebook searching (see below).

The most common measure for matching two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ of dimension n is the Euclidean

distance denoted by $d(x,y)$.

$$d(x,y) = \|x - y\|^2 = \sum_{i=1}^{n} (x_i - y_i)^2. \quad (1)$$

Once the optimal codebook is found, the best match for any vector (block) X_n in the input image is independent of the rest of the image. In other words, compression reduces to choosing a codeword Y_m such that the sum of all $d(X_n,Y_m)$ is minimized over the entire image.

Before compressing an input image, one has to design the optimal codebook from a set of representative images. This can be accomplished by using the generalized Lloyd's iterative algorithm, popularly known as the Linde, Buzo, Gray (LBG) algorithm, and is an unsupervised training procedure [17]. The algorithm iteratively increases the size of the optimal codebook in the manner described in the flow chart of Fig. 1. For more details on the design procedure, the interested reader is referred to [17].

In order to determine the centroids of the partitions which determine the optimum vectors of the respective partitions in the MSE sense, the underlying probability distribution must be known. However, in practice when the probability distribution is not known a priori, one uses the arithmetic mean. For details of the algorithm, the interested reader is referred to [7]. The algorithm is based on the superposition principle, that the distance of the whole image to the best codewords is minimized if the distance for each block is minimized. The resulting codebook is the one that minimizes the sum of all distances between image vectors and their representative codewords. The codeword that will replace the image vector in the compressed image is the one that lies closest in distance to the image vector. Once a codebook is defined, the distortion D due to compression of image X is defined as

$$D = \sum_{n=1}^{N} d(X_n, Q(X_n)),$$

where N is the total number of image vectors, X_n is an image vector and $Q(X_n)$ the codeword that represents X_n best in terms of Euclidean distance.

The full search VQ or simply FVQ, has no structure in the arrangement of the code vectors. In general, therefore, (1) has to be calculated for each input X_n and results in a time consuming search for large codebooks. However, it is possible to improve the search efficiency by exploiting the triangular inequality in the Euclidean space [13]. But with a tree structured VQ (TSVQ) a considerable reduction in computations can be achieved [9]. As an example, consider a binary tree structured VQ of depth 4 in which each node in the tree represents the centroid of a particular cluster of the input image. Starting at the root node, the tree is grown one layer at a time using the generalized Lloyd's algorithm. For a binary TSVQ of depth K, the number of vector searches required to obtain the best match can be shown to be $\log_2 K$ as opposed to K for an FVQ [7]. During the compression process, an input vector is compared with the two node vectors at level 1 and assigned a 0 or a 1 depending on whether the left or right node (centroid) was the closest. A similar operation is carried out at the next level and an additional bit is added to the path, till the terminal node is reached. The index (bit sequence) corresponding to the path is stored for each input vector instead of the actual vector, thus giving rise to compression. It is known that the TSVQ is suboptimal and performs poorer than the FVQ in root mean square (rms) error [7,24,3].

In FVQ and TSVQ, the same number of bits is allocated for all the vectors of an image. This results in a fixed rate VQ which implies that spatially coherent (shades) and spatially noncoherent (edges) blocks are encoded with the same number of bits irrespective of their probability of occurrence.

The visual effect of quantization (both scalar and vector) of an image is twofold: (a) edges appear jagged and (b) shade regions appear patchy. Due to the fact that our visual attention is naturally focussed on high contrast areas, the effect of edge degradation is more apparent. Therefore, by expending more bits to represent areas of high contrast, better visual quality for the same average bit rate of a fixed rate VQ can be achieved. Or conversely, for a given signal power to noise power ratio (SNR), higher compression ratios can

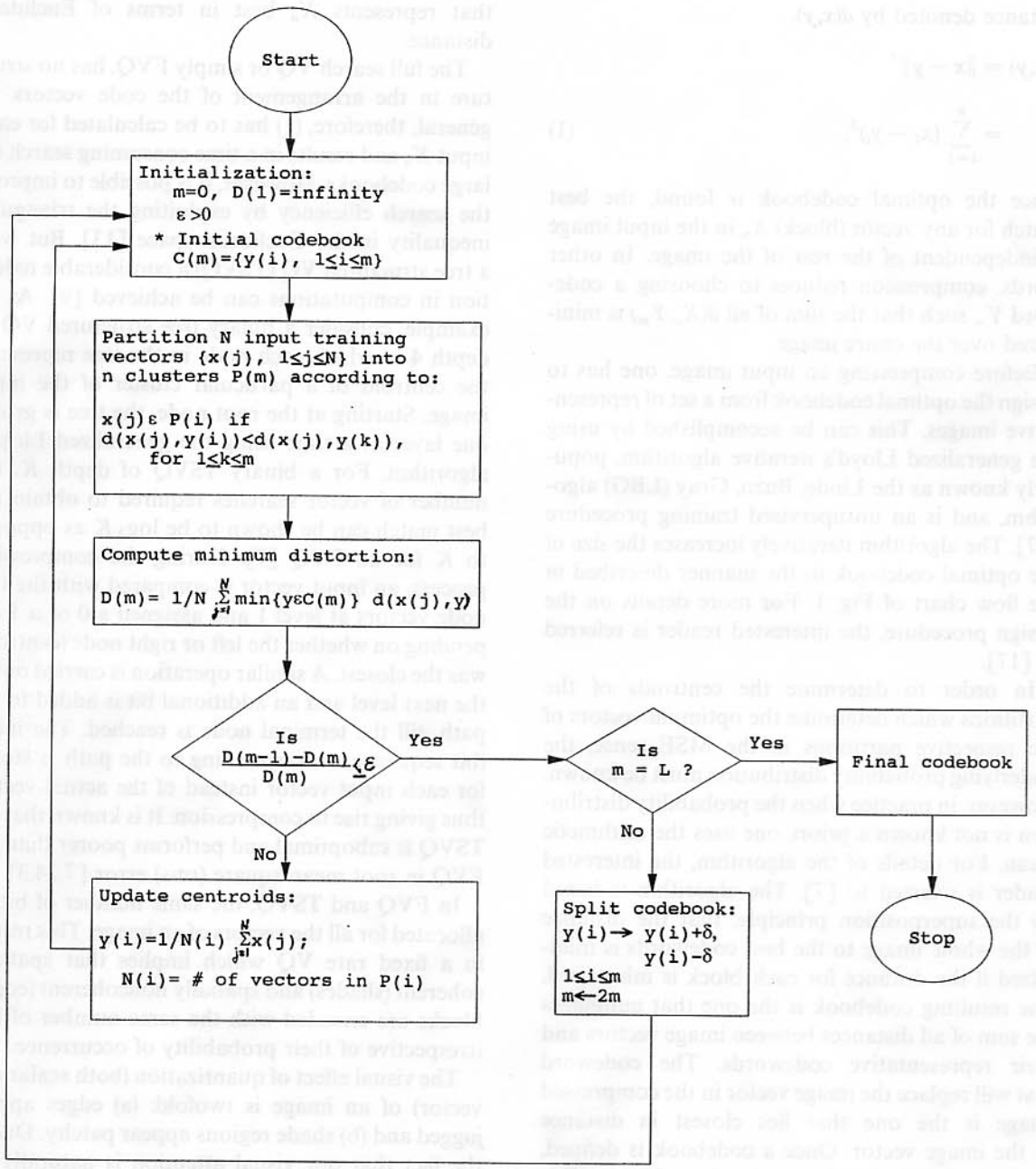

Fig. 1. Flow chart for the LBG VQ design algorithm.

be achieved from a variable rate VQ than from a fixed rate VQ. Note that this approach does not necessarily mean that numerical reproducibility is also enhanced.

4. Variable rate vector quantization

As mentioned above, areas with high details, such as edges, can be coded more accurately (with

more bits) than areas with low details so that the overall mse is reduced. Since each original image vector (block) is now replaced in the compressed image by an index (an integer) whose length varies according to the type of block (edge or shade), the VQ is called a variable-rate VQ. Since highly detailed regions are relatively fewer than low activity regions in most practical images, the variable rate coding will result in an average bit rate lower than a fixed rate scheme. A number of variable rate VQs can be found in [7,19,5]. One example is where the output of a full-search VQ is encoded into a variable length code using the Huffman coding scheme which is similar to the UNIX "compact" command. The Huffman method assigns larger codewords for the indices that occur less frequently, i.e., for edge blocks, than for the indices that occur more frequently (shades). This is suboptimal. Alternative approaches to the design of variable rate VQs do exist and can be categorized into *constrained*, *adaptive* and *finite-state* VQs. Among these the most important variable rate VQs are extensions of TSVQs. Here, one starts with a large TSVQ and optimally prune back branches to obtain a variable rate VQ using the pruning algorithm due to Breiman, Friedman, Olshen and Stone (BFOS) [4]. The optimality is in the sense of distortion-rate. This is called a pruned tree-structured VQ (PTSVQ). An obvious variation of this scheme is to use the BFOS pruning procedure to trade off average distortion with average entropy. This type of VQ is called the entropy-pruned TSVQ. In yet another method, one can jointly optimize the VQ and the variable rate codes. This is known as the entropy-constrained VQ. On the other hand, one can grow an unbalanced tree one node at a time in such a way that the node that results in the greatest decrease in distortion for a given rate increase is split. Naturally this results in a variable rate TSVQ but is suboptimal. We will only describe this suboptimal variable rate VQ in the following. Details on other constrained variable rate VQs can be found in [7].

4.1. Algorithm for designing a variable-rate VQ

In this paper, we implement an algorithm to design a VQ with a tree structure that has a variable number of nodes and to prune back the tree to achieve the lowest mse for a given compression ratio. As mentioned above, this type of VQ will use indices of different lengths, corresponding to the different nodes in the tree, to represent each block of the input image and is shown to outperform VQs that use fixed-length indices [25,26]. Usually, a TSVQ is designed one layer at a time using the generalized Lloyd's algorithm (GLA) [7]. The root node is the centroid of all vectors from the training images. Then the first layer is grown by splitting the root node into two initial nodes – by perturbing the root node slightly, and then these two nodes are optimized using the GLA. Each new layer of the tree is thus designed by splitting each intermediate node of the previous level into two 'perturbed' levels followed by the GLA. This tree is a balanced tree and results in a fixed rate TSVQ because the depth to every terminal node is the same.

Instead of designing (growing) one layer at a time, Makhoul et al. [20] proposed an alternative design method that splits the node that will result in the greatest distortion reduction. This algorithm is, therefore, 'greedy' because it splits each node without looking ahead, i.e. without taking into account its effect on the future performance. The resulting tree is unbalanced, having different depth to different terminal nodes. This will, therefore, require variable wordlength addresses to different terminal nodes and hence is a variable-rate VQ.

Riskin and Gray have proposed another method of growing an unbalanced greedy tree [25,26]. As in Makhoul's method, the tree is grown one node at a time based on the slope of the rate-distortion curve. In order to determine whether a node is to be split, a 'goodness' of split is defined as the decrease in distortion for increase in average rate. The node with the highest goodness of split is always split.

4.1.1. Tree functionals

A few definitions and terminologies pertaining to a TSVQ will be useful. For more details on tree definitions, one can refer to [14]. A tree denoted by T is a set of nodes $\{t_0, t_1, \dots\}$. The set of terminal nodes or leaves is denoted by $\tilde{T}$. The root node of a tree is usually denoted by t_0. A subtree S of T is a tree whose root coincides with that of T. A subtree is formally denoted by $S \preceq T$. A tree functional

is a real-valued function on a tree and its subtrees. Examples of tree functionals are the average length $l(S)$ and average distortion $\delta(S)$. Note that $l(S)$ increases monotonically as the tree size increases and that $\delta(S)$ decreases as the tree grows. With this preamble, we can describe the idea behind the GTSVQ design.

4.1.2. Greedy tree algorithm

Consider a tree T with its terminal nodes (leaves) denoted by $\tilde{T}$. Suppose that we split a node t of $\tilde{T}$ into left and right terminal nodes t_L and t_R, respectively. Let the average distortion and rate measured by the tree T be D and R, respectively, before splitting t and let the corresponding quantities be $\hat{D}$ and $\hat{R}$ after the node t is split. Each node gives rise to a certain average encoding rate and a corresponding distortion. If we increase the depth of a certain node by splitting it further, then the average encoding rate will increase with a corresponding decrease in distortion. Let the changes in distortion and rate due to splitting be represented, respectively, by ΔD and ΔR. In order to grow the tree, one should split the nodes at different depths. Then the rationale for splitting this node should be the largest magnitude of the slope of distortion rate curve, because we want to incur as little rate increase as possible with as large a decrease in distortion as possible. We must therefore calculate this slope at each node splitting. This magnitude is expressed as

$$\alpha = -\Delta D/\Delta R.$$

Note that ΔR is positive but ΔD is negative and hence the negative of this distortion rate change ratio is the magnitude of the slope. Since the average distortion for the tree is the sum of the node distortions, we can write

$$D = \sum_{j\in\tilde{T},\, j\neq t} p(j)d(j) + p(t)d(t), \tag{2}$$

$$R = \sum_{j\in\tilde{T},\, j\neq t} p(j)l(j) + p(t)l(t). \tag{3}$$

In the above equations, we have separated the effect of the node being split from the rest of the nodes. After splitting the node t, the distortion and rate are given by

$$\hat{D} = \sum_{j\in\tilde{T},\, j\neq t_L,t_R} p(j)d(j) + p(t_L)d(t_L) + p(t_R)d(t_R), \tag{4}$$

$$\hat{R} = \sum_{j\in\tilde{T},\, j\neq t_L,t_R} p(j)l(j) + p(t_L)l(t_L) + p(t_R)l(t_R). \tag{5}$$

Hence the changes in D and R are expressed by

$$\Delta D = \hat{D} - D = p(t_L)d(t_L) + p(t_R)d(t_R) - p(t)d(t), \tag{6}$$

$$\Delta R = \hat{R} - R = p(t_L)l(t_L) + p(t_R)l(t_R) - p(t)l(t). \tag{7}$$

Using the fact that

$$p(t_L) + p(t_R) = p(t), \tag{8}$$

$$l(t_L) = l(t_R) = l(t) + 1, \tag{9}$$

we can rewrite ΔD as

$$\Delta D = -p(t)\left[d(t) - \frac{p(t_L)}{p(t)}d(t_L) - \frac{p(t_R)}{p(t)}d(t_R)\right], \tag{10}$$

which can be simplified to

$$\Delta D = -p(t)[d(t) - p_L d(t_L) - p_R d(t_R)], \tag{11}$$

where we have used the fact that $p_L = p(t_L)/p(t)$ and $p_R = p(t_R)/p(t)$. Note that p_L and p_R respectively represent the proportion of vectors that are assigned to the left and right children of the node t. We can similarly write

$$\Delta R = p(t_L) + p(t_R) = p(t). \tag{12}$$

Thus the magnitude of the slope of the distortion rate function is given by

$$\alpha = -\frac{\Delta D}{\Delta R} = d(t) - p_L d(t_L) - p_R d(t_R). \tag{13}$$

Based on the above argument, we can design a variable rate VQ by designing a TSVQ, but one node at a time. For illustrative purposes, we only describe a binary TSVQ. First we have the root node which represents the mean value of the input training vectors. It is then split into two nodes using the LBG algorithm. At this point, the tree has a depth of one with an average rate of 1 bit/vector. Next we have to decide which of the two nodes to split. For this purpose, first the left node t_L is split using the LBG algorithm and its α_L is calculated. Similarly, the right node t_R is split into its two

children and its α_R is calculated. The slopes are then entered into a stack. Then the node to be actually split is the one with the largest α, and the corresponding node is denoted by t_{split}. Once the node has been chosen for splitting, its corresponding α is removed from the stack. The process is continued until the desired average encoding rate is reached. For details on the flowchart of the algorithm, one may refer to [25].

In this paper, we have designed greedy TSVQs and compared to other VQs in performance as applied to AVHRR satellite images.

5. Criteria for selecting a set of training images

The heart of block data compression methods is the codebook, which is basically the alphabet of the data set under consideration. In other words, it contains a representation of all the intensity patterns (spatial patterns) that can occur in any fixed size block (variable size blocks are possible, but not discussed here). We choose 4×4 samples as our block size for all the different algorithms. Based on this definition of a codebook, the training images need to be chosen such that all possible blocks that can occur in a given data set (e.g., all AVHRR data sets) are present in the training images. If for instance, the training images in the AVHRR case are selected to contain mainly clear sky areas, then cloudy regions in the test images will not be represented very well. This bias is beneficial if it is known beforehand that the compressed images will only be used for clear sky studies, because the clear sky regions will be represented by many codewords, hence lower noise levels in the clear sky regions than in the cloudy regions. Inducing a bias in selecting the training images makes the compressed images application specific, something that needs consideration in selecting the training images.

Theoretically, for an image source that is ergodic, an infinite number of images is required to train the vector quantizers. However, it is shown that a practical quantizer will approach the performance of a true quantizer with a finite number of images [7]. To generate a codebook that would represent all areas with both good visual quality and high numerical accuracy requires generating a codebook with several (5–10) images and many codewords, thus resulting in a low compression ratio. Therefore the importance of knowing ahead of time the class of images to be quantized and not compressing any other class with the same codebook cannot be over stressed.

Compare this process with a compression method that represents each letter by a number instead of a complex pattern of pixels. If we train it only on English text and then try to use the resulting codebook to compress Greek text, we will get illegible text back after compression, because the encoder replaced each Greek symbol with the best English substitute. So in order to make the codebook as general as possible, we need to feed it with text from many languages. By doing so, the number of codewords increases and the compression ratio decreases. This tradeoff between applicability of the codebook and compression ratio can be bypassed to some extent by carefully determining the applications and creating a different codebook for each of them. For example, if the compression is generally used on text of one specific language, it is more advantageous to use a separate codebook for every language plus maybe some combinations, such as English and Greek for scientific text. Another approach is one in which the scientist decides that he/she is not interested in Greek symbols in an English text, and decides to use an English codebook. The different approaches should be clear from this metaphor.

6. Comparison of complexity of VQ algorithms

6.1. Codebook generation

Machine capabilities and time constraints are important criteria for selecting which algorithm to implement. In generating codebooks, the FVQ can be designed in such a way that the final codebook of a given size can be generated by starting with a single codeword and progressing as a power of two until the final codebook size is obtained. Since the intermediate codewords need not be stored, the FVQ takes the least amount of machine memory but the greatest amount of time. FVQ has to find

the best match for each input vector by looking at every codeword in the codebook for each iteration of the GLA. Since codebook size increases as a power of 2 with each new bit, the computational load increases exponentially.

Tree-structured VQs (TSVQ) are computationally more efficient due to reduced search time. For each GLA iteration there are $2\log_2 M$ searches as opposed to M, where M increases as 2^l (l is the number of bits required for indexing and M is the current number of codewords). From this it can be seen that the computational load for TSVQs increases linearly with each increase in codebook size. The disadvantage is the amount of memory required to store a tree which is nearly twice as much. For FVQ the codebook requires $L*M$ floating point locations for storage, whereas TSVQ codebooks require $L*(2*M-1)$ floating point locations plus extra bits to denote terminal nodes, where L is the vector dimension.

A greedy tree is grown one node at a time as opposed to a layer at a time and will have many more codewords than an FVQ of equivalent rate. Therefore, we may expect the codebook generation time to be comparable to that of an FVQ. The greedy TSVQ requires an amount of memory greater than other TS methods. Even though the codebook is of the same average rate, it contains more than three times the number of codewords, hence requiring extra space.

6.2. Encoding

Encoding speed should be considered as a criterion as well, especially if there are many images to be transmitted from one location to another within a short amount of time. The time it takes to encode an image is strongly dependent on the VQ method – FVQ requires more number of vector searches than the TSVQ – just as codebook generation does. The argument about the number of calculations per vector applies here also. The slowest method would be FVQ due to the fact that it has to search the codebook exhaustively to encode each vector (hence it is inherently the least efficient, searchwise). The tree-structured methods (variable rate VQs) all have similar encoding times and are approximately three times faster than the full search (experimentally observed).

6.3. Decoding

For decoding purposes, all the discussed methods involve replacing the codeword index with the codeword itself. The amount of time used to determine the codeword assigned to an index is solely a factor of how many bits the index contains and how fast address computations can be carried out. So decoding speed between the discussed VQs at a certain rate is the same. The real difference between the methods is codebook design and encoding speed. Other overhead includes the time it takes to read in the encoded image and the corresponding codebook, but this would be similar for all methods.

7. Judging image compression based on visual and scientific reproducibility

Whether a compressed image will be used for visual or scientific purposes dictates the compression ratio and to some extent the compression method. For visual reproducibility, slight contrast enhancing methods are preferred because the human tends to focus mainly on areas with high contrast. This adds spatially nonuniform noise power to the original image, since only the high contrast areas are affected. For scientific reproducibility, the noise power should be spatially uniform and close to the noise power of the scanner. In terms of the vector quantizers, high compression ratios ($\geqslant 64$) can be used for visualization and lower values ($\leqslant 32$) for scientific reproducibility.

Visual performance is highly subjective and depends on the a priori knowledge about the content of an image. An example in the case of AVHRR images is the gradual and small variations in reflectivity (channel 1) for clear sky regions which are often suppressed during compression and might be noticed by a knowledgeable person even though the visual reproducibility for a layman seems very high. The ultimate goal in terms of scientific

reproducibility is achieved when the compressed images are identical to the originals except for a noise pattern, with a level close to the scanner's noise level. This type of constraint is not feasible for any significant ($\geqslant 8$) compression ratio, unless the image is of deterministic character. Acceptable scientific performance levels depend on future applications of the compressed images.

Data used for scientific purposes other than looking at the color coded or gray scaled images requires more than just visual reproducibility. The main goal for visual reproducibility is recognition. One application for highly compressed, visually acceptable images is their use in browsing through a large data set in search for the right region (e.g., convective clouds, hurricanes). Scientific reproducibility (numerical accuracy) plays in many areas of research, such as climate model verification, a much larger role than visual reproducibility and puts therefore tighter constraints on the applied compression techniques. Scientists have different intentions with the data, which places different constraints on the criteria used for scientific reproducibility. For example, clear sky analysis of AVHRR images requires accurate representation of low reflection (low contrast) areas, whereas cloudy samples surrounded by other cloudy pixels do not need to be represented with higher accuracy as long as they are not merged with clear sky samples. Research focussed on the distribution of optically thick clouds does not require accurate reproduction of clear sky samples. The final decision depends on: future usage, storage limitations, scientific value of the data set, etc. For demonstration purposes we selected a compression ratio of 32:1 which shows high visual reproducibility whereas the numerical reproducibility is slightly better towards high intensity, high contrast areas in the image. Suggestions for modification of the rate and methods to bypass these problems completely will be discussed.

8. Application to AVHRR satellite data sets

Four different vector quantization methods as described in the previous section are applied to two large AVHRR data sets. Comparison is based on quantitative as well as qualitative performance measures which will be described below.

8.1. Specifications of the AVHRR data sets

Advanced Very High Resolution Radiometer (AVHRR) was first flown on TIROS-N in 1978, and since then on the NOAA satellite series. The AVHRR data has a resolution of 1.1 km at nadir, a 10-bit dynamic range, a ground swath of 2700 km, and contains intensities from five spectral bands (channels) with respective bandwidths: band1 0.58–0.68 μm, band2 0.725–1.1 μm, band3 3.55–3.93 μm, band4 10.3–11.3 μm, band5 11.5–12.5 μm [16]. The measurements from the radiometers were calibrated, converted to the physical units, multiplied by 100, truncated to whole numbers and stored in two byte integers (16 bits per sample). This process worked well because of the 1% noise level of the instruments and the final values for all channels fell between 0 and $2^{16}/100 = 655$. Noise propagation will not be considered but can play a very important role in setting the data compression parameters.

Two data sets (2048×1024 samples) were used in the data compression experiments and divided into four quadrants of 1024×512 samples, hence resulting in eight data sets (images) per channel. Quadrants 1 and 3 were used in the training set to generate the codebooks of the various fixed and variable-rate vector quantizers, whereas the other four images from quadrants 2 and 4 were used as test data in the comparison between the different compression methods. Because of similar patterns in all the channels, it was decided to focus on channel 1.

8.2. Applied compression methods

The pruned greedy tree-structured vector quantizer (PGTSVQ) was applied to spectral channels 1, 2, 4 and 5 using Riskin's method, generating a separate codebook for each channel. Each VQ used 4×4 pixel blocks resulting in a vector dimension of 16 ($L = 16$) which was a tradeoff between blocking effect and compression ratio. The codebooks were

grown to an average rate of 8 bits/vector which is equivalent to 0.5 bit/pixel or a compression of 32:1. For pruning the trees, the procedure is to grow the tree to an average rate greater than the final rate. In general, the greater the initial bit rate the better the performance will be after pruning. This is due to the fact that a tree with a larger number of depths will have a greater flexibility in pruning to achieve a desired target rate than the one with lesser depths. In this work, the trees were grown to an average rate of 12 bits/vector and then pruned back to the required rate of 8 using the BFOS algorithm [4], under the constraint of a maximum depth of 16 which is based on a compromise between storage and performance. In order to compare the performance of the PGTSVQ with other VQs, four other VQs, namely full-search VQ (FVQ), tree-structured VQ (TSVQ), pruned-tree-structured VQ (PTSVQ), greedy-tree-structured VQ (GTSVQ) were applied to the same set of images with the same parameters. All the different VQs were designed for an average rate of 8 (i.e. 8 bits per codeword or a compression ratio of 32:1). A GTSVQ (i.e., a greedy TSVQ without pruning) was also designed for an average rate of 15 and compared with PGTSVQ for rate 8 to demonstrate how certain problems could be overcome.

8.3. Data analysis

Two mathematically tractable performance measures which are all based on the original image (X), the decoded image (Y) and the difference or error image ($E = Y - X$) are used in the comparisons. They are: (1) the mean square error (MSE)

$$\mathrm{MSE} = \frac{1}{N}\sum_{n=1}^{N} E_n^2 = \frac{1}{NL}\sum_{n=1}^{N}\sum_{l=1}^{L} E_{n,l}^2 = \frac{1}{NL}\sum_{n=1}^{N} E_n^{\mathrm{T}} E_n, \tag{14}$$

and (2) the peak signal to noise ratio (PSNR)

$$\mathrm{PSNR} = 10\log_{10}((\max(X))^2/\mathrm{MSE}),$$

where $\max(X)$ is the maximum value in X. For some figures we need a third measure, namely, the standard deviation which is defined as

$$\mathrm{STD}(X) = \sqrt{\frac{1}{NL-1}\sum_{n=1}^{N}(X_n - \overline{X})^{\mathrm{T}}(X_n - \overline{X})}.$$

In addition to the above mentioned measures of performance, we use scatter plots to evaluate the quality of a compression scheme. Scatter plots of mean and standard deviation values for the original and compressed images based on 4×4 pixel blocks give insight into possible degradations of edges and shades. Besides the scatter plots, error histograms were calculated to assess the measure of performance of a quantizer. Ideally, the distribution of the error between the original and the compressed sample values must be unbiased and random.

The absolute error measures were based on the maximum value in the image, 100.0 for channel 1 and 115.0 for channel 4. Some care needs to be taken when blindly finding the maximum value because of possible bad samples or in the case of satellites, bad scan lines. It is advised to use the physical upper limit as a reference for error measures. For example, the values found in channel 1 or 2 cannot exceed 100%, because no more than 100% of the sunlight can be reflected back into space.

8.4. Results

All graphical results are from the variable-rate tree-structured VQ called the pruned greedy tree-structured VQ (PGTSVQ) for a compression ratio of 32:1, except where specified otherwise. Note that this compression ratio is for the file size since the original pixels are stored as two byte integers. Visual comparisons can be made from Fig. 2, where the original channel 1 data (ACH1_3) is shown (1024×512 samples) along with the compressed versions using the fixed and variable-rate VQs (TSVQ and PGTSVQ). In Fig. 2, the gray scale resolution is 16 bits/pixel, though the actual value occupies only 10 bits, the average bit rate is 0.5 bit/pixel with a PSNR of 41.2 dB. The visual quality of the original and the worst performer, namely the fixed-rate tree-structured VQ (TSVQ), is almost identical in the normal resolution images. To notice

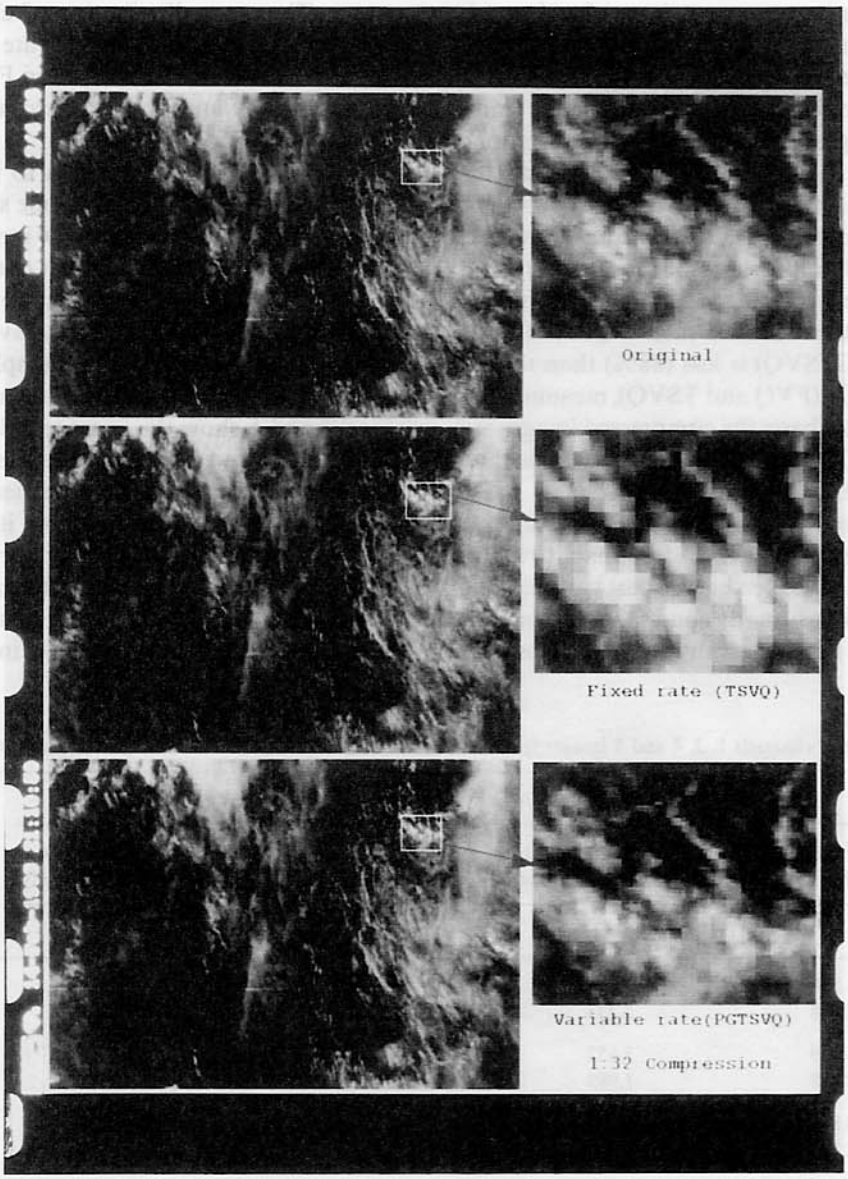

Fig. 2. Visual comparisons of channel 1 original and compressed images (image size = 512 × 512 pixels @ 10 bits/pixel) on two scales for TSVQ and PGTSVQ (32:1 compression) methods. The full resolution images are on the left and the zooms of the marked square region on the right side; PSNR = 41.2 dB for the PGTSVQ and 38.01 dB for the TSVQ; in both cases the bit average bit rate is 0.5 bit/pixel.

the differences one has to zoom into the images. Note that the 4 × 4 codewords of the fixed-rate VQ (TSVQ) become visible in the blown up inserts. The jagged edges and patchiness which often appear around edges and high contrast areas in vector quantized images can be overcome by increasing the number of available codewords or by using the variable-rate VQ (PGTSVQ) which appears almost as smooth as the original while maintaining the same average rate as the fixed-rate VQ (TSVQ).

The root mean square errors (rmse) for the various compression methods applied to channels 1, 2, 4 and 5 are listed in Table 1. The first half of the table gives the rmse for the images inside the training set (i.e., images used in codebook generation), and the second half for the images outside the training set (test images which were not part of the training set). Note that since quadrants 1 and 3 were used for training the codebooks, they will have, on average, a lower rmse than the test images. The rmse of the variable-rate VQs (PTSVQ, GTSVQ and PGTSVQ) is less (88%) than that of the fixed-rate VQs (FVQ and TSVQ), meaning that on a pixel by pixel basis the compressed images via variable-rate VQs were closer to the originals than those due to the fixed rate VQs.

The percentage of pixels with an absolute error ranging from less than 1% to less than 10% are given in Table 2. As expected, the best variable-rate VQ (PGTSVQ) consistently shows the highest percentage of pixels having less than specified absolute errors. The error distribution of channel 1 image (ACH1_3) for the best variable-rate VQ (PGTSVQ) column of Table 1 is displayed in Fig. 3 and shows approximately an unbiased, zero-mean, Laplacian distribution.

The previous results pertain to individual pixel value comparisons. However, one should also consider average properties to draw meaningful statistical conclusions about the effect of the applied data compression methods. To assure correct reproducibility of average properties, the average and variance of 4×4 and 32×32 sample blocks were calculated on the original and compressed images. Figs. 4 and 5 show the average reflectance properties for 4×4 and 32×32 blocks, respectively, while Fig. 6 shows the variance properties for 32×32 size blocks. From these scatter plots it is evident that the compressed images have preserved the average properties very well. Note the horizontal stripes in Fig. 4 which are indicative of the same codewords being used to represent a range of intensities. This is

Table 1
Comparison of rmse of channels 1, 2, 4 and 5 images for fixed and variable-rate VQs for an average compression of 32:1

	RMS errors in W/m² for training and test images (1:32 compression)				
	Fixed rate (full search)	Variable rate (pruned greedy)	Fixed rate (full search)	Variable rate (pruned greedy)	
Images	Training (RMSE)		Test (RMSE)		Images
Ach1_1	1.860	1.759	1.982	1.928	Ach1_2
Ach1_3	2.658	2.233	1.911	1.844	Ach2_4
Ach2_1	1.729	1.652	1.804	1.764	Ach2_2
Ach2_3	2.328	1.989	1.676	1.654	Ach2_4
Ach4_1	0.801	0.9865	2.126	1.967	Ach4_2
Ach4_3	2.707	1.968	1.488	1.395	Ach4_4
Ach5_1	0.795	1.036	2.307	2.155	Ach5_2
Ach5_3	2.879	2.062	1.549	1.491	Ach5_4
Bch1_1	2.006	1.843	2.481	2.372	Bch1_2
Bch1_3	2.648	2.300	2.039	1.974	Bch1_4
Bch2_1	1.974	1.807	2.366	2.262	Bch2_2
Bch2_3	2.456	2.136	1.974	1.898	Bch2_4
Bch4_1	1.263	1.319	1.341	1.561	Bch4_2
Bch4_3	2.563	2.143	2.047	2.247	Bch4_4
Bch5_1	1.271	1.360	1.414	1.638	Bch5_2
Bch5_3	2.766	2.318	2.117	2.220	Bch5_4

Table 2
Pixel error statistics for variable rate VQ (PGTSVQ) channel 1 (1 : 32 compression). Percentage of scene within 1, 2, 5 and 10% abs. error (abs. error is with respect to maximum value in the scene)

Images	⩽ 1% Error	⩽ 2% Error	⩽ 5% Error	⩽ 10% Error
Training				
Ach1_1	46.44%	65.82%	92.73%	99.35%
Ach1_3	67.16%	83.42%	97.23%	99.73%
Bch1_1	54.06%	71.14%	93.42%	99.44%
Bch1_3	54.92%	72.21%	93.10%	99.00%
Test				
Ach1_2	57.78%	75.53%	94.66%	99.34%
Ach1_4	54.46%	74.91%	95.75%	99.61%
Bch1_2	53.17%	69.51%	90.09%	98.20%
Bch1_4	54.15%	73.61%	94.25%	99.11%

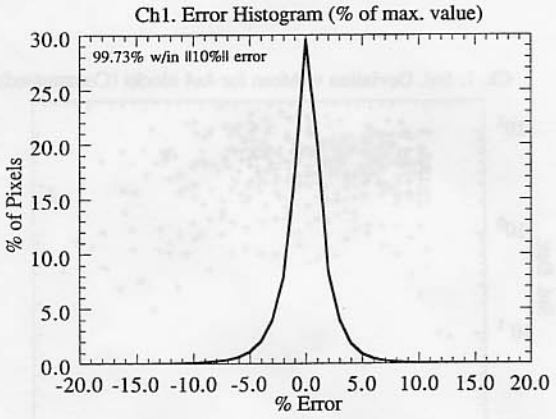

Fig. 3. Error histogram of the PGTSVQ (32 : 1 compression) on ACH1_3.

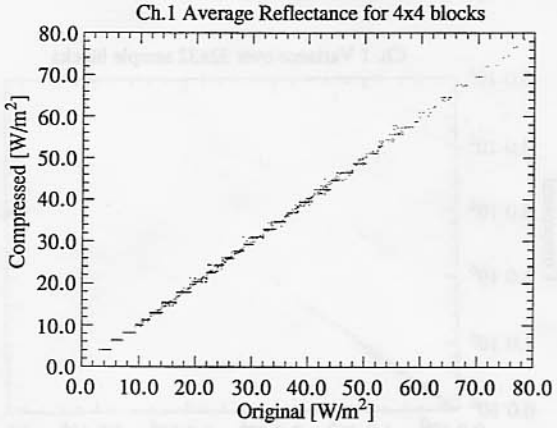

Fig. 4. Average reflectance over 4 × 4 samples of the PGTSVQ (32 : 1 compression) on ACH1_3.

a tell tale sign which is especially prevalent in low contrast regions which are under-represented in the codebook, but nevertheless contribute very little to the overall rmse. The variance of these sample blocks was also calculated to provide insight into the average reproducibility of high contrast areas. These properties are very important for classification algorithms which tend to be highly threshold sensitive. A comparison between the variance of the original and compressed channel 1 data (ACH1_3) is shown in Fig. 6. There is generally a good agreement as would be expected even though the variance for the compressed image is slightly less. This occurs because the edges in the reconstruction are not as sharp due to the 4 × 4 blocks used for quantization and because of the limit on the number of codewords.

Scatter plots of the standard deviation versus mean for the 4 × 4 blocks of the original and compressed channel 1 images are shown in Figs. 7 and 8, respectively. The compressed image corresponds to the variable-rate PGTSVQ with a 32 : 1 compression. Similar plots are shown for a lower compression ratio of 17 : 1 in Fig. 9. It is clear that the low intensity samples for 32 : 1 compression are only represented by a very few codewords, which is solved by a higher rate (less compression, Fig. 9) VQ. We also see that in Fig. 9, the high intensity

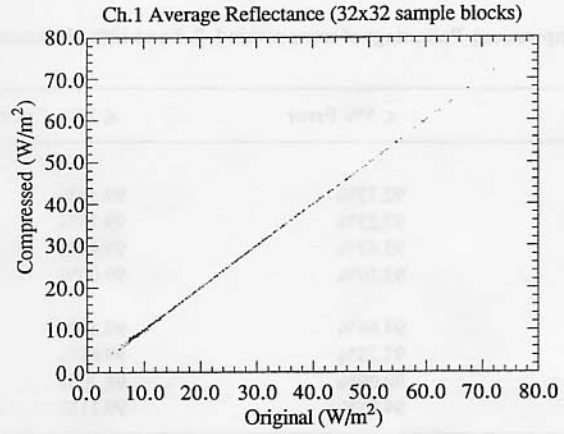

Fig. 5. Average reflectance over 32×32 samples of the PGTSVQ (32:1 compression) on ACH1_3.

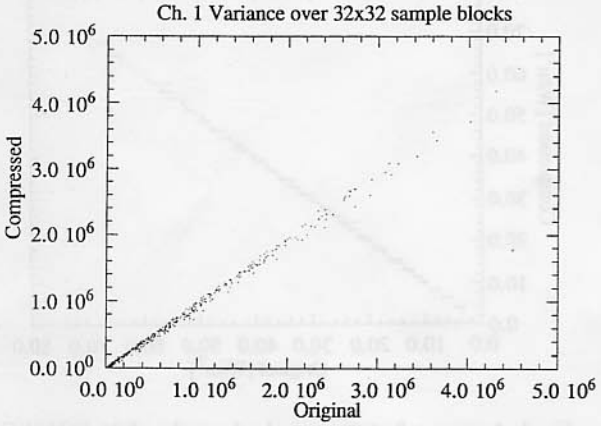

Fig. 6. Variance of reflectance over 32×32 samples of the compressed ACH1_3 image using PGTSVQ (32:1 compression).

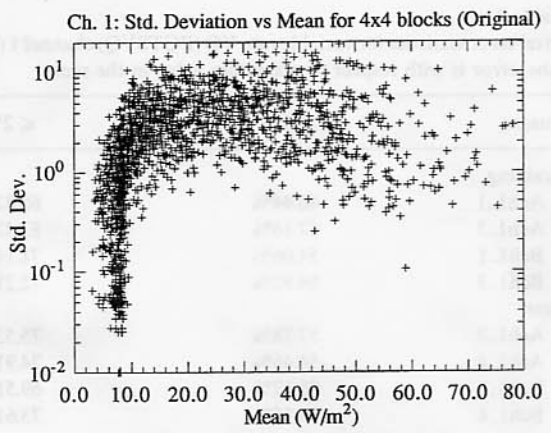

Fig. 7. Mean versus standard deviation of reflectance over 4×4 samples of the original ACH1_3 image.

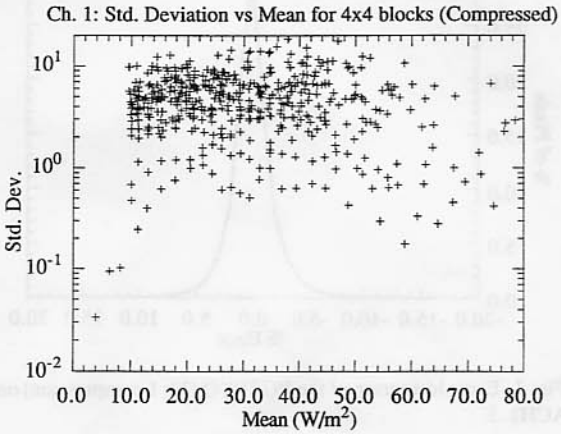

Fig. 8. Mean versus standard deviation of reflectance over 4×4 samples of the compressed ACH1_3 using PGTSVQ (32:1 compression).

samples are represented very well. The variable-rate VQ (PGTSVQ) method tends to represent high intensity regions better than the low intensity regions. Fig. 10 shows the low intensity region of the cumulative distribution function of channel 1 data (ACH1_3). The zoomed version of it is shown in Fig. 11. A similar result as described in the previous paragraph is observed here.

8.5. Comparison with JPEG standard

For compression of still pictures, the ISO has recently adopted a standard known as JPEG (Joint Photographer Expert's Group) which basically combines DCT, scalar quantization, DPCM and Huffman coding to achieve high compression at good quality. The typical compression achievable by the JPEG algorithm is around 20:1. Generally, the algorithm determines the compression ratio on the basis of a desired quality factor. It is not possible to exactly match the compression ratio with the VQ, for instance. In this work, we ran the JPEG algorithm with a quality factor of 23 which amounted to a compression of approximately 32:1,

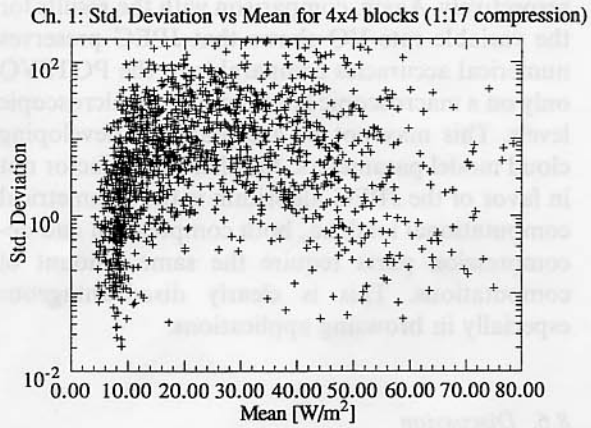

Fig. 9. Mean versus standard deviation of reflectance over 4×4 samples of the compressed ACH1_3 using PGTSVQ (17:1 compression).

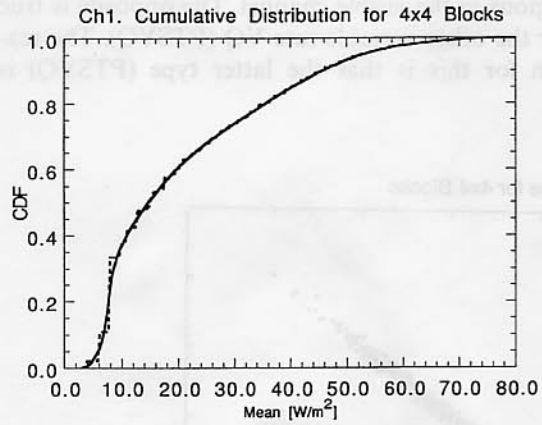

Fig. 10. Cumulative pixel distribution, the original (solid) and compressed (dashed) ACH1_3 using PGTSVQ (32:1 compression).

thereby establishing a base for comparison. Fig. 12 shows the original and the compressed versions. As can be seen, the visual quality of JPEG compressed image is comparable to the variable rate PGTSVQ (Fig. 2) at 32:1 compression. When using the JPEG algorithm, the compression ratio is calculated as the original image file size ($512 \times 512 \times 16 = 524\,288$ bytes; pixels in the original image occupy two bytes) divided by the compressed file size (16 449 bytes). This gives an average bit rate of 0.25 bit/pixel for the JPEG scheme

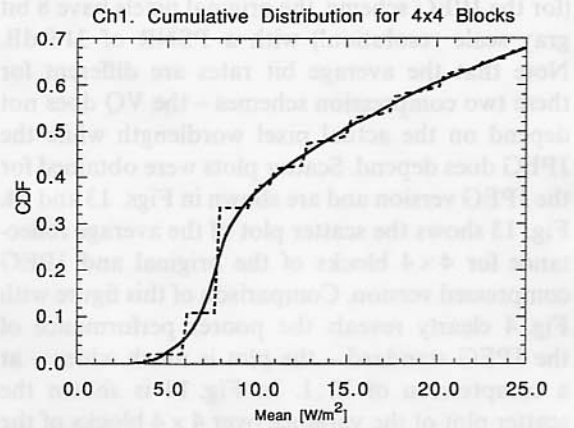

Fig. 11. Zoom of cumulative pixel distribution, the original (solid) and the compressed (dashed) ACH1_3 using PGTSVQ (32:1 compression).

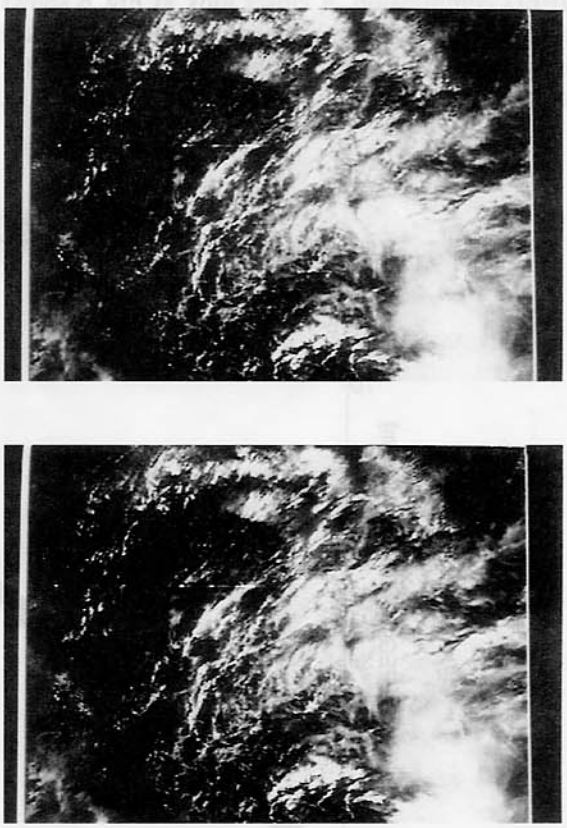

Fig. 12. Picture of channel 1 satellite image compressed using JPEG standard with a quality factor of 23 (approximately 32:1 compression): image size = 512×512 pixels @ 8 bits/pixel, PSNR = 31.9 dB with 0.25 bit/pixel coding rate. (a) Original, (b) compressed.

(for the JPEG scheme, the original pixels have 8 bit gray scale resolution!) with a PSNR of 31.9 dB. Note that the average bit rates are different for these two compression schemes – the VQ does not depend on the actual pixel wordlength while the JPEG does depend. Scatter plots were obtained for the JPEG version and are shown in Figs. 13 and 14. Fig. 13 shows the scatter plot of the average reflectance for 4×4 blocks of the original and JPEG compressed version. Comparison of this figure with Fig. 4 clearly reveals the poorer performance of the JPEG standard – the plot is much wider – at a compression of 32:1. In Fig. 14 is shown the scatter plot of the variance over 4×4 blocks of the original versus JPEG compressed versions. The scatter plot of the variance shows the increase in the mean square values (bulging of the plot) due to JPEG compression. A closer look at Fig. 6 shows that the PGTSVQ outperforms the JPEG standard by a significant margin. Similar scatter plots for block sizes of 32×32 are shown in Figs. 15 and 16, respectively. Again, comparison with the results for the variable rate VQ shows that JPEG preserves numerical accuracies comparable to the PGTSVQ only on a macroscopic level but not on microscopic levels. This may not be acceptable in developing cloud model parameters. Another major factor not in favor of the JPEG algorithm is the symmetrical computational load, i.e., both compression and decompression parts require the same amount of computations. This is clearly disadvantageous especially in browsing applications.

8.6. Discussion

From the scatter plots we find that the variable-rate VQ (PGTSVQ) does well at high intensity regions (cloudy), but poorly at dark (clear sky) regions in the visible channel. The opposite is true for the other variable-rate VQ (PTSVQ). The reason for this is that the latter type (PTSVQ) is

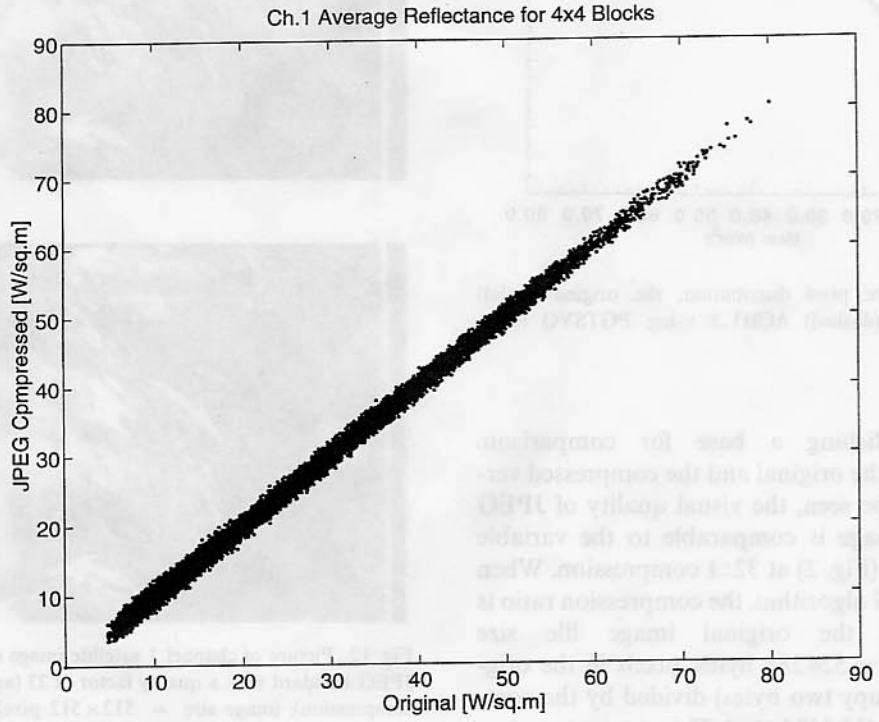

Fig. 13. Average reflectance over 4×4 samples of the JPEG compressed image (approximately 32:1 compression) on ACH1_3.

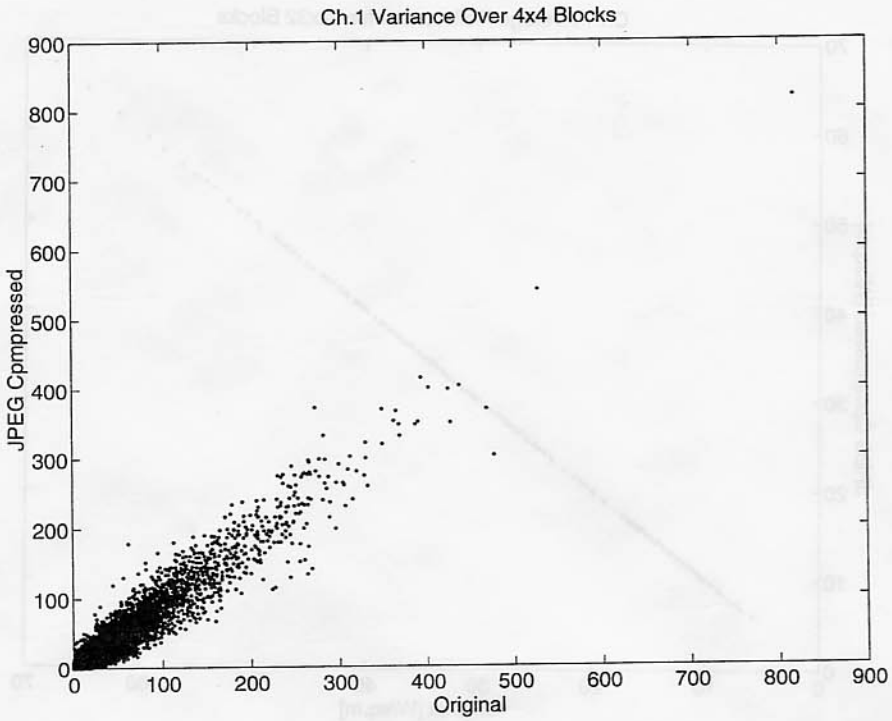

Fig. 14. Variance of reflectance over 4×4 samples of the compressed ACH1_3 image using JPEG standard (approximately 32:1 compression).

initially a balanced tree and is pruned back to the required rate and therefore has some characteristics of the fixed-rate VQs which represents all regions in the same manner. Therefore, the nodes of this VQ (PTSVQ) split more evenly towards high intensity areas (clouds) while nodes corresponding to darker areas have empty cells. The overall effect is a poorer quality of compression for the PTSVQ than the PGTSVQ which, by definition, is an unbalanced tree that grows according to the slope of the distortion-rate curve. In the case of infrared images for instance, the nodes corresponding to high pixel intensities will have a small slope and, therefore, may not grow before the nodes representing darker pixels grow. Nevertheless, the PGTSVQ will still perform poorly in the clear sky regions due to the fact that the overall distortion does not increase in the same order as in cloudy areas because of low intensity values. Hence the tree does not develop completely in the dark regions. Moreover, the empty cell problem (an empty cell represents a node with zero population) in the variable-rate VQ (greedy TSVQ without pruning) can be avoided entirely by adding a check to the algorithm and terminating those branches that do not have enough population to be split.

Even though the variable-rate VQ (PGTSVQ) lacks in performance in the clear sky region, the overall visual quality is better than that of other variable-rate VQs, namely the pruned tree-structured VQ. This may be argued by noting that the eye is sensitive to high contrasts. Since the contrast is very low in the clear sky regions, high distortion in those areas is not readily perceived by the human eyes. Whereas distortions in the cloud regions are easily perceived due to high contrast. This explains why the latter type VQ (PTSVQ) performs poorer in terms of visual quality compared to the former type (PGTSVQ). However, low numerical accuracy in the clear sky areas can have severe consequences in, for example, sea surface temperature calculations. The decompressed images can be made to

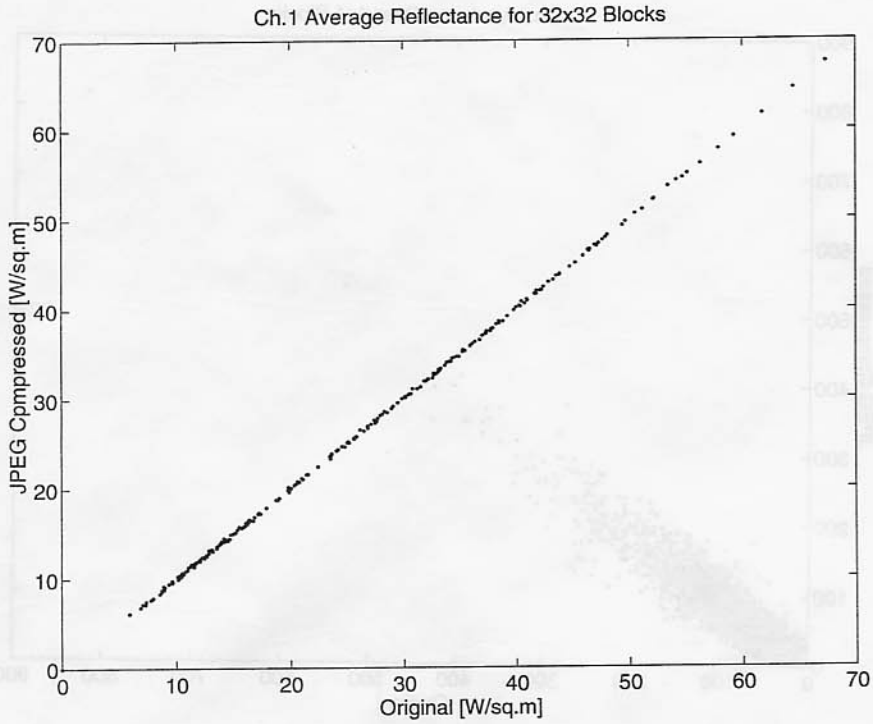

Fig. 15. Average reflectance over 32×32 samples of the JPEG compressed image (approximately 32:1 compression) on ACH1_3.

appear smoother and sharper using postprocessing techniques. For example, the noise due to the limited codebook size (called the quantization noise) can be somewhat smoothed either by dithering or by using spatially varying filters [23]. Other postprocessing techniques such as adaptive contrast enhancement and pseudocoloring can also be used to improve the quality (visual and numerical accuracy) of the decompressed image.

Even though pixel by pixel comparisons are crucial for scientific reproduceability, we must pay careful attention to the comparison of the statistical properties such as the mean and variance as well. Scatter plots showing the mean and variance of 4×4 and 32×32 pixel blocks of the original versus compressed images reveal general agreements between the two. This is an encouraging result, since many studies depend upon average properties. The poorer performance of the PGTSVQ in the clear sky regions is again revealed by the scatter plot of the intensity average. There it is seen from the horizontal stripes that the PGTSVQ uses a few codewords to represent clear sky regions as opposed to cloud regions. This is further exemplified by the cumulative distribution function (cdf) plot. The blown-up figure of the cdf plot, Fig. 11, shows that the quantizer has denser representations above the intensity level of approximately 8. As mentioned before, the low intensity values are used in segmenting an image into cloudy and clear sky regions, which may be altered by the limited number of low intensity codewords. This shortcoming is overcome when lower compression ratios are used, say 21:1. This clearly demonstrates the versatility of the PGTSVQ and its ability to tradeoff compression versus quality without affecting other factors, like the computational complexity.

High numerical reproducibility can be achieved with compression ratios in the order of 8:1 to 16:1 and even higher if the images are of deterministic origin. Close interaction between scientists who use the compressed images and those who develop

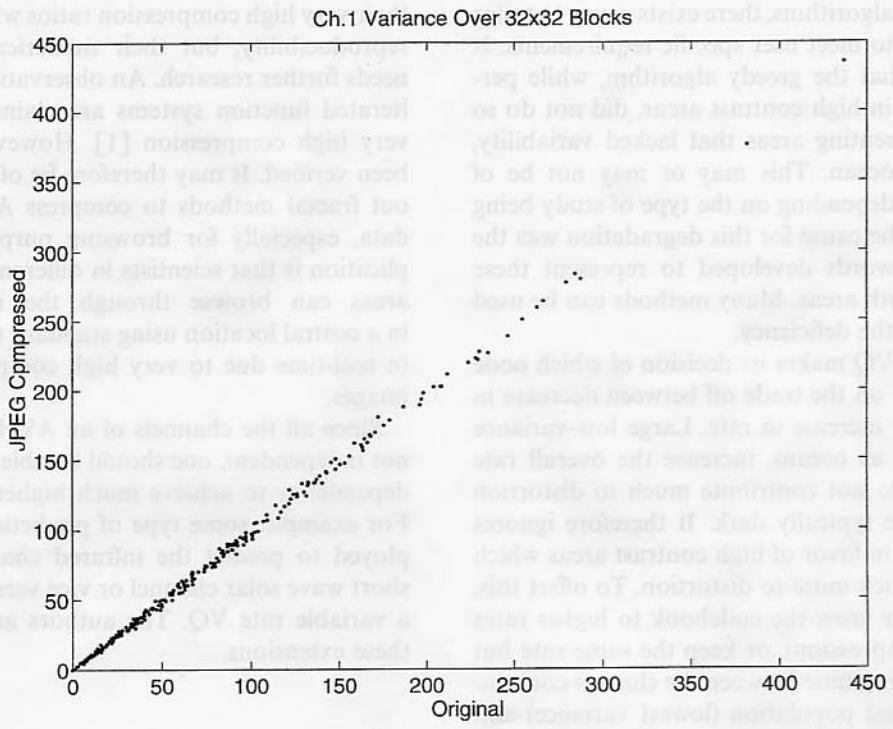

Fig. 16. Variance of reflectance over 32×32 samples of the compressed ACH1_3 image using JPEG standard (approximately 32:1 compression).

compression algorithms is of crucial importance. Since there are different cloud images with variations in structure and compositions, it may be necessary to generate different codebooks for each cloud category. During compression, one then has to determine the type of each block or a region of the image to access the proper codebook. This may involve a higher computational cost and the decision to use such a scheme depends on the application at hand.

9. Concluding remarks

In this paper, the applicability of variable-rate vector quantization as a vehicle to compress data sets like the AVHRR is investigated. Both fixed and variable-rate VQs were designed for the visible and infrared channels. In the case of the variable-rate schemes, the trees were grown to a depth of 12 (average compression of 21:1) and then pruned back to the required average rate. It is found that even at a compression ratio of 32:1, the image quality of a PGTSVQ is excellent, and that it performs consistently better than the fixed-rate VQs and the PTSVQ. The superior performance of the PGTSVQ is exemplified in the scatter plots and the error histograms. Moreover, a comparison of performance with the JPEG standard algorithm shows that JPEG compression is inferior to the PGTSVQ at a compression of 32:1 (PSNR of 41.2 dB for the PGTSVQ versus 31.9 dB for the JPEG) in preserving the numerical accuracy. It is important to note that the VQ, in general, does not depend on the word length of pixel intensities, such as 8 or 16 bits/pixel, whereas JPEG does depend on the word length. This is an important factor to be reckoned with since wordlength limitation, as in the JPEG standard, severely degrades the numerical integrity. Overall, the PGTSVQ seems to be a suitable image compression method for data sets such as the AVHRR.

As with all algorithms, there exists a way to tailor performance to meet user specific requirements. It was shown that the greedy algorithm, while performing well in high contrast areas, did not do so well in representing areas that lacked variability, such as the ocean. This may or may not be of consequence depending on the type of study being conducted. The cause for this degradation was the lack of codewords developed to represent these visually smooth areas. Many methods can be used to overcome the deficiency.

The PGTSVQ makes its decision of which node to split based on the trade off between decrease in distortion for increase in rate. Large low-variance regions, such as oceans, increase the overall rate rapidly but do not contribute much to distortion since they are typically dark. It therefore ignores these regions in favor of high contrast areas which contribute much more to distortion. To offset this, one can either grow the codebook to higher rates (decrease compression), or keep the same rate but split the training time between the clusters containing the greatest population (lowest variance) and the clusters containing the highest variance. Another method would be to weight the distortion term in the slope equation with the inverse of the range of intensities within the cluster. The above modifications will yield a decrease in visual quality or compression but may maintain a greater scientific accuracy within the image.

More complicated schemes may also be considered that would classify the images into categories, say of oceans and clouds in the case of AVHRR images, and develop a codebook for each category [23]. This would increase the complexity of the encoder/decoder and would double the number of codebooks, but would yield a more even treatment for both low contrast and high contrast areas while maintaining the same compression ratio.

Further compression without sacrificing the quality (visual and numerical) can be achieved using more advanced techniques such as the wavelet transform [30] or fractal methods of which the latter uses iteraterative function systems. These techniques can also be combined with the PGTSVQ to achieve high quality and very high compression. These alternative techniques are highly suited for browsing purposes because of their very high compression ratios with good visual reproducibility, but their numerical quality still needs further research. An observation here is that iterated function systems are claimed to achieve very high compression [1]. However, it has not been verified. It may therefore be of interest to try out fractal methods to compress AVHRR image data, especially for browsing purposes. The implication is that scientists in different geographical areas can browse through the images stored in a central location using standard telephone lines in real-time due to very high compression of the images.

Since all the channels of an AVHRR image are not independent, one should be able to exploit this dependency to achieve much higher compression. For example, some type of prediction can be employed to predict the infrared channel from the short wave solar channel or vice versa, followed by a variable rate VQ. The authors are working on these extensions.

Acknowledgements

This work was funded by the Alderson chair funds and the Zetleson foundation. The authors wish to thank the anonymous reviewers for their constructive criticisms.

References

[1] M.F. Barnsley, A. Jacqui, Application of recurrent iterated function systems to images, SPIE Vol. 1001, Vis. Comm. and Image Processing 1988, pp. 122–131.

[2] B.A. Baum, B.A. Wielicki, M. Patrick, Cloud-property retrieval using merged HIRS and AVHRR data, J. Appl. Meteorol. 31 (April 1992) 351–369.

[3] H. Bheda, K.S. Thyagarajan, H. Abut, A fast matrix quantizer for image encoding, in: Proc. IEEE ICASSP-85, March 1985, pp. 4.7.1–4.7.4.

[4] L. Breiman, J.H. Friedman, R.A. Olshen, C.J. Stone, Classification and Regression Trees, Wadsworth, Belmont, CA, 1984.

[5] P.A. Chou, T. Lookabaugh, R.M. Gray, Optimal pruning with applications to tree-structured source coding and modeling, IEEE Trans. Inform. Theory 35 (2) (March 1989) 299–315.

[6] D.F. Elliot, K.R. Rao, Fast Transforms, Algorithms and Applications, Academic Press, New York, 1983.

[7] A. Gersho, R.M. Gray, Vector Quantization and Signal Compression, Kluwer Academic Publishers, Dordrecht, 1992.
[8] A. Gersho, Ramamurthy, Image coding using vector quantization, in: Proc. IEEE ICASSP-82, April 1982, pp. 428–431.
[9] R.M. Gray, Vector quantization, IEEE ASSP Magazine (April 1984) 4–29.
[10] A.K. Jain, Image data compression: A review, Proc. IEEE 69 (March 1981) 349–389.
[11] A.K. Jain, Fundamentals of Digital Image Processing, Prentice-Hall, Englewood Cliffs, NJ, 1989.
[12] N.S. Jayant, P. Noll, Digital Coding of Waveforms, Prentice-Hall, Englewood Cliffs, NJ, 1984.
[13] M.H. Johnson et al., Fast nearest neighbor search for ECVQ and other modified distortion measures, ICIP96 3 (1996) 423–426.
[14] D.E. Knuth, The Art of Computer Programming, Vol. 1, Addison-Wesley, Reading, MA, 1973.
[15] F. Kossentini et al., Low bit rate coding of earth science images, in: Data Compression Conf., DCC-93, March 1993, 371–80.
[16] L. Lauritson, G.J. Nelson, F.W. Porto, Data extraction and calibration of TIROS-N/NOAA radiometers, NOAA Technical Memorandum NESS-107, Washington, DC, November 1979, Reprinted October 1985.
[17] Y. Linde, A. Buzo, R.M. Gray, An algorithm for vector quantizer design, IEEE Trans. Commun. COM-28 (January 1980) 84–95.
[18] S.P. Lloyd, Least squares quantization in PCM, Unpublished Bell Laboratories Technical note. Portions presented at the Institute of Mathematical Statistics Meeting Atlantic City New Jersey September 1957. Published in the March 1982 special issue on quantization of the IEEE Trans. Inform. Theory (1957).
[19] T. Lookabaugh, E.A. Riskin, P.A. Chou, R.M. Gray, Variable rate vector quantization for speech, image, and video compression, IEEE Trans. Commun. 1991.
[20] J. Makhoul, S. Roucos, H. Gish, Vector quantization in speech coding, Proc. IEEE 73 (11) (November 1985) 1551–1587.
[21] T. Markas, J. Reif, Multispectral image compression algorithms, in: Data Compression Conf. DCC-93, March 1993, pp. 391–400.
[22] W.K. Pratt, Digital Image Processing, Wiley, New York, 2nd ed., 1991.
[23] B. Ramamurthy, A. Gersho, Classified vector quantization of images, IEEE Trans. Commun. COM-34 (November 1986) 1105–1115.
[24] B. Ramamurthy, A. Gersho, A. Sekey, Low-rate image coding using vector quantization, in: Conf. Record, GLOBECOM, October 1983.
[25] E.A. Riskin, E.M. Daly, R.M. Gray, Pruned tree-structured vector quantization in image coding, in: Proc. ICASSP, 1989, pp. 1735–1738.
[26] E.A. Riskin, R.M. Gray, A greedy tree growing algorithm for the design of variable rate vector quantizers, IEEE Trans. Signal Process. 39 (11) (November 1991) 2500–2509.
[27] M. Schwaller, R. Price, J. Dalton, Science data plan for the EOS data and information system, covering EOSDIS version 0 and beyond, Document Version 2.0, NASA Goddard Space Flight Center, Greenbelt, MD, June 1993.
[28] C.E. Shannon, Coding theorems for a discrete source with a fidelity criterion, in: IRE National Convention Record, Part 4, 1959, pp. 142–163.
[29] K.S. Thyagarajan, M. Viswanathan, Matrix quantization of homomorphically processed images, in: Proc. MILCOM'86, October 1986.
[30] J.W. Woods, Subband Image Coding, Kluwer Academic Publishers, Dordrecht, 1991.