Predictive Vector Quantization of Images

HSUEH-MING HANG, MEMBER, IEEE, AND JOHN W. WOODS, SENIOR MEMBER, IEEE

Abstract—**The purpose of this paper is to present new image coding schemes based on a predictive vector quantization (PVQ) approach. The predictive part of the encoder is used to partially remove redundancy, and the VQ part further removes the residual redundancy and selects good quantization levels for the global waveform. Two implementations of this coding approach have been devised, namely, sliding block PVQ and block tree PVQ. Simulations on real images show significant improvement over the conventional DPCM and tree codes using these new techniques. The strong robustness property of these coding schemes is also experimentally demonstrated.**

I. INTRODUCTION

BY *predictive vector quantization* (PVQ) we will mean a predictive tree encoding in which the ordinary scalar quantizer is replaced by a vector quantizer (VQ). Because typical images have high correlation over neighboring pixels, they can be compressed by employing a predictive model such as DPCM and tree codes [1], [2]. However, since a real image is nonstationary in nature, a scalar quantizer together with a fixed structure coding filter can only condense pictures to a certain extent. Vector quantizers help improve the coding performance because they quantize a whole block of data and, thus, can match local image statistics better. The purpose of this paper is to present new image coding schemes based on the PVQ concept.

Most of the conventional image coding techniques fall into two categories: predictive coders and transform coders [3]–[5]. Conceptually, these coding schemes have two stages: 1) remove the source signal redundancy which does not provide useful information, and 2) select "good" representatives which contain the essential information for reproduction of the original signal. Both predictive and transform techniques try to achieve the first goal—removing signal redundancy. For continuous-amplitude waveforms, the selection of representatives is performed either by quantization or other coding techniques such as random tree codes. Our coding schemes will follow the direction of the second category: using an encoding filter to remove the predictable redundancy of the source signal.

Rate distortion theory indicates that a well-defined signal source can be compressed closely to the rate distortion bound, provided that the coding block length is large enough [6]. From this viewpoint, conventional DPCM has the drawback that its predictor only uses the past information to remove redundancy and its quantizer only operates on a single pixel. A predictive tree code is thus introduced by adding a delayed decision feature which makes use of the nearby future data [7], [8]. The tree code is then further improved by replacing the scalar quantizer with a vector quantizer, resulting in a predictive vector quantizer.

From the quantizer's point of view, PVQ can be regarded as an extension of vector quantization. This situation is analogous to the predictive quantizer—a different name for DPCM [1]. The quantization levels in a predictive quantizer are first biased by the prediction filter and then used to quantize the source waveforms. Or, equivalently, the prediction errors rather than the original signals are quantized.

Predictive tree coding for 1-D speech compression was successfully developed by Anderson and his coworkers [7], [8]. This approach has also been extended to image coding [2], [9], [10] where about 3 dB SNR improvement over DPCM was reported [2], [9], [10].

On the other hand, although the vector quantizer (block quantizer) concept has been around for a long time [11], the theory of VQ has advanced rapidly only in the past several years. The advantage of using VQ is evident, since the usual scalar quantizer is a special case of VQ with length 1. Experimentally, the superiority of VQ over the scalar quantizer was first reported by Buzo *et al.* for low-rate speech compression [12]. Along the same line, VQ has been recently employed to encode images [11]. For example, Gersho and Ramamurthi [13] proposed a method which classifies pixels in an image into several constituent groups and then applies VQ on each group. This approach was further refined and extended in [14] and [15]. Baker and Gray introduced a differential vector quantizer which removes the mean of each image block and then vector-quantizes the residual signals [16], [17].

Image encoding using PVQ [19] is not a straightforward extension of the ordinary vector quantization. A special implementation of 1-D PVQ has appeared for speech coding in Stewart *et al.* [18]. But the full potential of the general PVQ approach, especially its application to images, has not been explored. In order to construct a code tree on a compact 2-D region, we devised a 2-D decision order which provides an appropriate encoding sequence for 2-D tree codes. The details of this ordering can be found in [20], [26].

Several 2-D PVQ coding schemes are developed in this paper. The original VQ concept and predictive tree codes will be briefly reviewed in Section II. Two implementations of PVQ will then be presented in Section III. The significant coding improvement offered by these techniques is demonstrated by experiments in Section IV. Some related problems such as sensitivity to the VQ table are also discussed there.

II. BACKGROUND

In order to understand 2-D PVQ, some of the necessary background material including the concepts of predictive tree codes, vector quantizers, and 2-D search ordering are briefly reviewed in this section. Our main results will be presented in the next section.

A. Predictive Tree Codes

A predictive tree code is a special type of tree code in which the branches are generated by predictive filters with quantized inputs. In other words, the branch values in this tree are filter

Paper approved by the Editor for Signal Processing and Communication Electronics of the IEEE Communications Society. Manuscript received June 4, 1984; revised April 3, 1985. This paper was presented in part at the 18th Annual Conference on Information Science and Systems, Princeton University, Princeton, NJ, March 1984.

H.-M. Hang was with the Department of Electrical, Computer, and Systems Engineering, Rensselaer Polytechnic Institute, Troy, NY 12181. He is now with AT&T Bell Laboratories, Holmdel, NJ 07733.

J. W. Woods is with the Department of Electrical, Computer, and Systems Engineering, Rensselaer Polytechnic Institute, Troy, NY 12181.

0090-6778/85/1100-1208$01.00

outputs excited by the quantization levels. In practice, to locate good paths (codewords) in a tree, an instrumentable search (nonexhaustive) technique must be adopted. We have chosen the (M, L) search because of its reasonably good performance, synchronized transmission, and moderate computational complexity [21], [22]. The parameter M indicates the maximum number of paths retained at each tree depth and L is the delay or search length at which a decision on the released point is made.

It had been believed that tree coding performance would always improve by increasing the search length. On the contrary, we have found that when a conventional (M, L) search is used, increasing the search length can degrade coding performance due to the *near merging of paths* phenomenon [10]. Fortunately, this performance degradation for PVQ is much less than that of an ordinary predictive tree code. This will be shown in Section IV-A.

B. Vector Quantization and Clustering Design Algorithm

The ordinary-scalar, or one-dimensional, quantizer is an operator which converts a single sample of an analog signal to one of a finite number of representatives (quantization levels). In vector quantization, a block of J samples is mapped into one of a finite set of representative vectors. An index that identifies this representative is then sent out.

An N-level, J-dimensional quantizer can be modeled as a mapping $\mathbf{F}$ from the input J-dimensional space $\mathbf{X} = \{\boldsymbol{x} = (x_1, \cdots, x_J)\}$ to a finite reproduction set $\mathbf{Y} = \{y_i: i = 1, \cdots, N\}$. Each vector y_i in $\mathbf{Y}$ is a J-dimensional vector. The mapping $\mathbf{F}$ is completely characterized by the partition $\{P_i: i = 1, \cdots, N\}$ in the input space $\mathbf{X}$, which assigns an input vector $\boldsymbol{x}$ to the representative y_j if $\boldsymbol{x} \in P_j$. The reproduction set $\mathbf{Y}$ is also called the VQ lookup table since the index of y_i, which is transmitted over the channel, can be regarded as an address in a table.

Recently, a simple and effective VQ design algorithm was proposed by Linde *et al.* [23] as an extension of a scalar quantizer design scheme due to Lloyd. One attractive feature of this approach is that it does not require knowledge of the source statistics. The parameters in the coding scheme are obtained by applying an iterative algorithm on training sequences. Additional details on this VQ design method including convergence properties can be found in [23], [24]. This coder design algorithm will be extended to the design of two-dimensional PVQ in Section III.

C. 2-D Search Order

A new problem for 2-D tree coding which does not appear in 1-D is to decide the tree generating sequence, *search order*, inside a 2-D region. In 1-D tree coding, there is only one natural order for tree searching, but this is not the case for 2-D signals. As discussed in [20] and [26], where search order is called *decision order* for broader use, we may define several eligible search orders on a 2-D region. The line-by-line scan order is generally permissible and is called *1-D-like search* for its similarity to 1-D tree search order. However, since our search is over a 2-D spatial region, a 2-D search order which attemps to minimize the redundancy in the code tree will yield better results, as will be shown in Section IV-B.

The construction of this 2-D order depends on both the filter support and the search region geometry. In fact, the shape of the search region also restricts the choice of filter support for a causal coding process. Although this 2-D order will be used in Section III-B, because of limited space the details of 2-D decision ordering are relegated to [20] and [26].

III. Predictive Vector Quantization

The basic idea of predictive vector quantization (PVQ) is to use a predictive filter to remove the predictable redundancy in the data and then use a VQ to encode the prediction error. We will show two implementations of PVQ in this section, namely, *sliding block PVQ* and *block tree PVQ*.

A. Sliding Block Implementation

Fig. 1 represents an ordinary sliding block decoder in which the u_i's are the inputs to the shift register, the q_i's are the outputs of the decoder, and $\mathbf{F}$ is a time-invariant mapping which specifies the output value q_i. Suppose the shift register is binary with length J; then the total number of possible states of this machine is 2^J, i.e., the mapping $\mathbf{F}$ has 2^J entries. This mapping $\mathbf{F}$ can thus be viewed as a lookup table, with the shift register acting as an address selector which picks entries in the table to form the outputs. In this way, the current output q_i is determined by the vector $\boldsymbol{U}_i = (u_i, u_{i-1}, \cdots, u_{i-J+1})$ which is the state of the shift register, where u_i is the current input and $u_{i-1}, u_{i-2}, \cdots, u_{i-J+1}$ are the $J-1$ previous inputs. Hence, the information contained in the previous data can be utilized to select the best current output value q_i.

We can view $\boldsymbol{U}_i$ as an index to the vector quantizer. At the time J, this index corresponds to the representation vector $\boldsymbol{Q}_J = (q_J, q_{J-1}, \cdots, q_1)$. For $i > J$, we simply slide the block to the right; hence the name "sliding block" for this type of VQ. To quantize a sampled waveform, the source signal is compared against all the quantization levels specified by the shift register of which the latest input has two possible values; the one with least distortion is then selected. The ordinary scalar quantizer can be viewed as the special case of this machine which only contains one element in the shift register. Therefore, if we choose the mapping $\mathbf{F}$ properly, the performance of VQ will always be better than that of a scalar quantizer.

We can also adopt a sliding block structure to implement PVQ, which we call *sliding block PVQ* (SBPVQ). The block diagram of a 2-D SBPVQ decoder is shown in Fig. 2. The encoding filter in this decoder is a recursive difference equation,

$$\hat{s}(m, n) = \sum_{i,j} c_{ij} \cdot \hat{s}(m-i, n-j) + q(m, n)$$
$$\equiv \boldsymbol{c} \circledast \hat{\boldsymbol{s}}_{\text{old}} + q(m, n),$$

i.e., the reproduced signals $\{\hat{s}(\cdot, \cdot)\}$ are filter outputs driven by the selected PVQ levels. One of the problems in applying this scheme to a 2-D image is the selection of a register support for the mapping $\mathbf{F}$. Since an image pixel is highly correlated with its neighbors, naturally we would choose a compact region around the current point to be our register support. For example, the causal region of Fig. 3 could be the support of the register in Fig. 2. As we slide the region of Fig. 3 horizontally across the image, the current quantization level (input to the filter) is determined by the contents of the register, i.e., the previous and current path map symbols. The encoding filter then uses this quantization level to generate a reproducing pixel, $\hat{s}(m, n)$.

We now turn to the encoder of SBPVQ. Usually the encoding process is tedious and requires a large amount of computation. An SBPVQ encoder can be thought of as running several decoders simultaneously. Then it makes a delayed decision on the released path map. This procedure is best illustrated by an example using a small test image as in Fig. 4. Fig. 5 shows the code tree generated by this example. We start with pixel 1, where all the pixels before it are assumed to be previously released pixels or boundary values. A 1 bit/pixel quantizer will be employed.

Comment 1: Except for the current pixel (with index 7), all the elements in the register support are fixed boundaries. Since the current pixel can have two possible path values (two-level quantizer), two branches stem out at pixel 1, as seen in Fig. 5.

Comment 2: If the encoder does not make a decision at

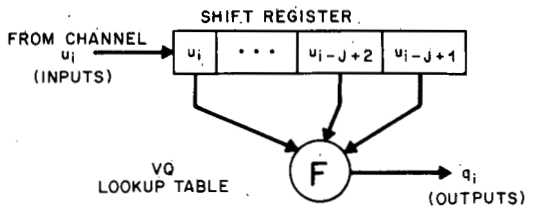


Fig. 1. A sliding block channel decoder.

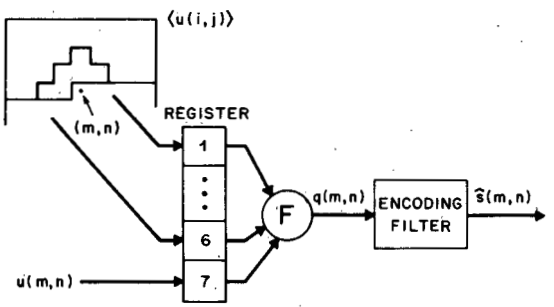


Fig. 2. A 2-D SBPVQ decoder.

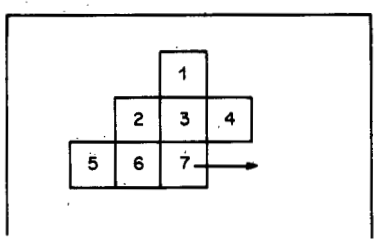


Fig. 3. A causal region for the register in SBPVQ.

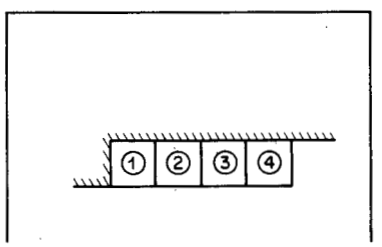


Fig. 4. A test image for SBPVQ.

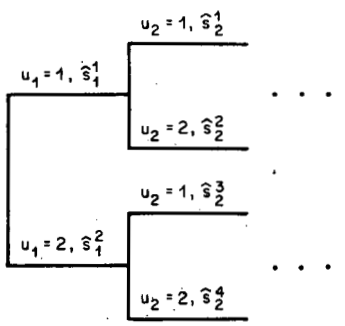


Fig. 5. A tree generated by an SBPVQ.

pixel 1, it moves the shift register support to the next pixel. There are four branches at this pixel. The first two are extended from the first branch at pixel 1 by varying the most recent input to the register. Similarly, the other two branches come from the second branch at pixel 1. Thus, a code tree is constructed. A search algorithm such as the (M, L) algorithm can then be used to locate good paths in the tree.

Essentially, an SBPVQ requires about the same amount of computation as a tree code but needs an extra register and a VQ table. The complexities of a few predictive coding algorithms are compared in Table I. One important issue in the design of an SBPVQ is how to determine a "good" VQ table **F**. We propose an iterative approach based on the clustering algorithm of [23]. In order to describe our design algorithm, we need to define two more terms. In the encoding process, releasing a data pixel is equivalent to selecting an entry in the VQ table for that pixel. The index of the selected entry will be called the *partition index associated with this pixel*. Also the unquantized prediction error (i.e., $e(m, n) = s(m, n) - c \circledast \hat{s}_{\text{old}}$) will be called the *prediction error associated with the released pixel*. The SBPVQ design algorithm can then be described.

SBPVQ Design Algorithm:

Step 1: Initialization: Start with some initial value for **F**. For example, use the scalar quantization levels derived from a predictive tree code (so-called product VQ codes in [11]).

Step 2: Coding: Apply the above encoding procedure to the training data, i.e., introduce a minimum distortion partition $\{P_1, \cdots, P_N\}$ on the test image. Store the prediction error $e(m, n)$ and the partition index of each pixel. The partition index associated with a data point is, equivalently, the contents of the register used to encode that pixel.

Step 3: Updating **F**: Since the squared-error is used, the new quantization level of index j is the average of all the prediction errors of partition index j, i.e.,

$$q_j = \frac{1}{|P_j|} \sum_{(m,n) \in P_j} e(m, n)$$

where $|P_j|$ denotes the number of training vectors in partition P_j.

Step 4: Compute the distortion and compare it to the previous distortion. Stop if the distortion decrement is less than a prespecified value. Otherwise, go to Step 2.

B. Block Tree Implementation

The block tree implementation of PVQ is easy to appreciate in concept. A test image is first partitioned into small blocks, and then predictive tree coding is performed on each block. The difference between block tree PVQ and a tree code is that the quantization levels in the former are vectors.

Initially, we considered ideal block PVQ (full-searched PVQ) which has a full size VQ table and requires an exhaustive search. Due to computational consideration, this scheme was deemed impractical. Then we imposed a tree structure on the VQ table, calling the new algorithm *block tree* PVQ. The idea of tree-structured vector quantizers was first proposed by Buzo *et al.* for linear predictive coding (LPC) of speech [12]. However, the tree search technique used here is different. The tree-searched VQ table in [12] and [25] is a list of vectors organized by a tree-like framework, and the search is basically an address locating procedure. A node in that tree is the representative for all the nodes (or branches) extending from that node. Only the ultimate leaves (nodes or branches without successors) of the tree are used as code vectors. On the other hand, we follow the traditional sequential tree coding approach to construct the VQ table. Every tree branch is a part of a code vector. A complete code vector is formed by concatenating the branch symbols along any path in the tree.

The structure of an ideal block PVQ decoder is shown in Fig. 6. The path map $\boldsymbol{u}$ from the channel is a vector containing an address in the VQ lookup table. An entry in the VQ lookup table is another vector which is a sequence of quantization levels used to drive the encoding filter. As a simple decoding example, consider the test image in Fig. 7. At the receiver, the

TABLE I
COMPUTATION AND STORAGE OF CODING SCHEMES

	DPCM	Tree Code	SBPVQ	BTPVQ
Number of Encoding Filters Calculated (per pixel)	1	M	M	M
Storage for Delay Decision	0	$M \cdot L$	$M \cdot L$	$M \cdot L$
Storage for VQ Table	0	0	b^J	b^{BLK}

b = average number of branches stemmed from one node in the tree
L = number of delayed elements in the (M,L) search
M = number of paths retained in the (M,L) search
J = size of the shift register
BLK = elements of one block

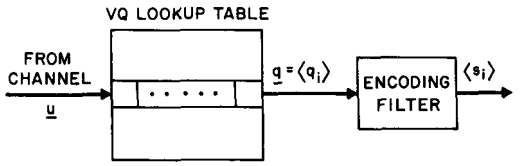


Fig. 6. An ideal block PVQ decoder.

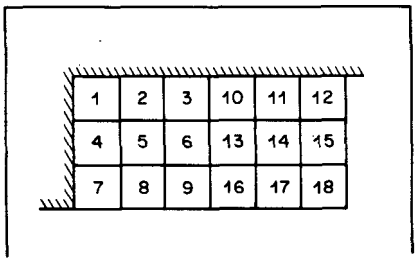


Fig. 7. A test image for BTPVQ.

quantization vector $q = \{q_1, q_2, \cdots, q_9\}$ of a 3 × 3 block is selected by the path map symbol u. Then, each element q_i passes through the encoding filter and yields the reconstructed signals $\hat{s}_i$ sequentially.

Since the block of this PVQ is a compact 2-D region, the search order of elements inside a block should follow the 2-D search ordering defined in [20] and [26], there called decision ordering. Indeed, the 1-D-like search would yield a less satisfactory result, as will be seen in Section IV-B. The importance of search order becomes apparent when a full-searched PVQ is replaced by a tree-searched PVQ. The 2-D search region also limits the geometric shape of the encoding filter so that the decoder is causally realizable. For instance, a nonsymmetric half-plane filter cannot be used with a rectangular block search region.

At the transmitting end, in order to get the best quantization vector for a given block, ideally we should try all the entries in the VQ table and send out the one with the least distortion. If the number of entries in the VQ table is N, we would need N decoder operations to locate the best quantization level. For a typical image, this may be impractical. For instance, a VQ table for a 3 × 3 block and 1 bit/pixel rate code contains 512 vectors. To find the optimum quantization vector, all the vectors in this VQ table have to pass through the encoding filter and then all the reproduced vectors must be compared against the source signal. This search would require 512 decoding computations for each pixel.

The computational problem of an ideal block PVQ can be greatly eased by imposing a tree structure on the VQ table, as

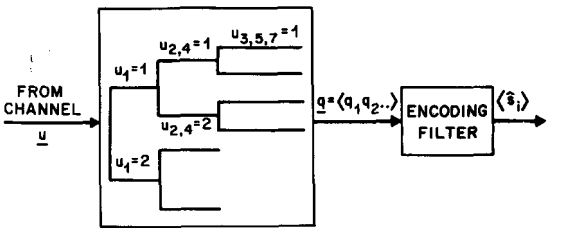


Fig. 8. The structure of BTPVQ.

mentioned above, and applying the (M, L) search algorithm to the code tree. We call this new scheme *block tree PVQ* (BTPVQ). As illustrated by Fig. 8, the VQ lookup table now has a tree structure and a path in the tree is the quantization vector identified by the path map symbol u. If we apply an (M, L) search with $M = 8$ on the test image of Fig. 7, the encoder only conducts 2 × 8 or fewer decoder operations per pixel, which is much smaller than the 512 operations of the ideal block PVQ.

Additionally, the encoder does not have to make a decision immediately at the end of a block. Instead, it can delay its decision-making and thus take advantage of the dependence between successive blocks. For example, the code tree in the first block of Fig. 7 can be extended to the second block, and the encoder would then release the first block after reaching the end of the second block. In other words, the tree structure inside one block would act as a substitute for a full-searched table, and the delayed decision feature can be brought in by allowing the tree to grow continuously over several blocks.

The overall delayed BTPVQ coding process for Fig. 7 with a 1 × 1-order quarter plane filter would proceed as follows.

Step 1: The level-1 elements in the tree-structured VQ table of Fig. 8 are used to produce image pixel 1. In this example, there are two branches at tree depth 1; hence, two corresponding $\hat{s}_1$'s are generated. When the (M, L) algorithm is used, only the best M paths are retained for the next step.

Step 2: Image pixels 2 and 4 are *co-order points,* i.e., they are encoded simultaneously. Hence, all the branches at tree depth 2 of this VQ table are associated with two quantization levels, one for pixel 2 and the other for pixel 4. Although there are four entries at tree depth 2, only the first two are permissible if $u_1 = 1$ was the choice at level 1. These permissible paths are then sorted and retained by the (M, L) algorithm.

Step 3: The above procedure continues until reaching pixel 9. Then it starts again using the level-1 entries in the VQ table to encode pixel 10, while the newly generated branches are still attached to the surviving paths at pixel 9. A delayed decision will finally be made on the first block (pixels 1–9) at pixel 18.

The entire encoding procedure for the BTPVQ is thus quite similar to that of an ordinary predictive tree code. The only change is that a path is released block by block rather than point by point. In fact, the computation and storage complexities of a BTPVQ are about the same as those of an ordinary tree code plus the additional VQ table. These calculational requirements of BTPVQ and some other schemes are given in Table I. From this point of view, the ordinary one-bit predictive tree coder can be regarded as a special case of BTPVQ for which the block size is 1 × 1 and the table size is 2 × 1.

It remains to describe the design of the VQ table for BTPVQ. Again, an iterative approach similar to the one used for SBPVQ was formulated. The notation used in Section III-A is followed here.

BTPVQ Design Algorithm:

Step 1: Initialization: Suppose that the initial VQ table is known, e.g., use the scalar quantization levels.

Step 2: Coding: Apply BTPVQ to the training image,

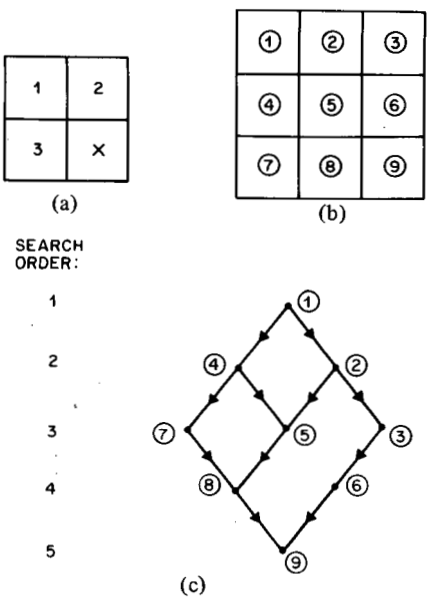


Fig. 9. (a) A 1 × 1-order quarter-plane filter. (b) A local 2-D search region. (c) Search order of (b).

resulting in a minimum distortion partition $\{P_1, \cdots, P_N\}$ of the training sequence. The partition index associated with each data block is the path map in the code tree used for that block.

Step 3: Updating the VQ table: The new quantization level of a branch in the VQ tree is the average value of all the prediction errors associated with that branch.

Step 4: Compute the distortion and compare it to the previous distortion. Stop if no further improvement. Otherwise, go to Step 2.

Since the VQ operation is performed on the whole block rather than on a single point, the number of branches per node in the tree can be arbitrarily chosen. In the example of Fig. 7 the 2-D search order of elements inside a block would appear as shown in Fig. 9(c). The code tree of this VQ table has depth 5, since the total search order of one block is 5. If the desired average bit rate is 1 bit/pixel, then 9 bits are available to index the tree. One possible arrangement is (1, 2, 3, 2, 1) bits per node, or equivalently (2, 4, 8, 4, 2) branches per node. By this notation we mean two branches per node at depth 1, four branches per node at depth 2, and so on. An alternative arrangement is (2, 2, 2, 2, 1) bits per node. It seems appropriate to put more freedom at the beginning of a tree, because the later part of the tree search will be constrained by the initial selections. Of course, other bit allocation patterns are permissible. More generally, we can also have noninteger numbers, say 1.5 bits (three branches) per node.

IV. Simulations

All the following simulations were run on 256 × 256 images with 8 bits/pixel gray level. Two images were used: a lady's face and a man's face. The channel transmission rate was set to 1 bit/pixel. In fact, the entropy of the path map derived from the coding schemes was usually somewhat less than 1 bit/pixel. Therefore, these coding algorithms can be realized on a constant rate channel with a bit rate equal to or less than 1 bit/pixel. The following conventions have been adopted: a) the global mean of the test image is removed before encoding and then is added back after decoding; b) all the boundary values used in the coding process are set to zero; and c) the SNR of the coded image is defined to be the ratio of the peak-to-peak signal (255) to the root mean squared-error for these density-domain images [5].

A. SBPVQ

The encoding filter for this coder is a 1 × 1-order nonsymmetric half-plane (NSHP) filter, with coefficient support as shown in Fig. 10. The filter coefficients were obtained by least-squares parameter estimation on the test image and are listed in Table II. The rate of the path map (the input to the sliding block register) was 1 bit/pixel. This sliding block register contains 7 bits in total, of which the last one comes from the channel and the rest are picked from the previous path map in a local region defined in Fig. 3. The initial values of the mapping **F** in the iterative design procedure were obtained from the best ordinary predictive tree code with the same model but a scalar quantizer.

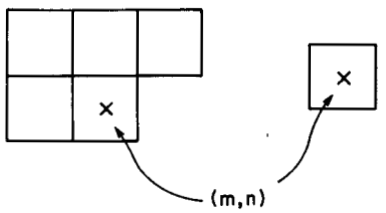


Fig. 10. A 1 × 1-order NSHP model coefficient support.

TABLE II
MODELS OF TEST IMAGES

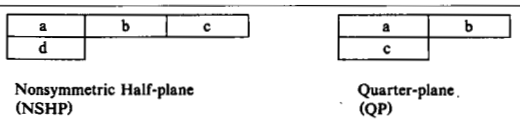


Image	Model	Coefficients (a,b,c,d)	Estimation error variance
Lady's face	QP, separable	-.8813, .96, .918	176.6
	QP, least-squares	-.313, .809, .485	129.0
	NSHP, least-squares	-.237, .54, .232, .463	113.2
Man's face	QP, separable	-.9506, .973, .977	63.3
	QP, least-squares	-.5306, .79, .739	51.2

TABLE III
THE DESIGN OF AN SBPVQ (1 BIT/PEL)

Iteration	Search parameters	SNR (dB)
1	M= 4, L=10	26.3
2	M= 4, L=10	28.9
3	M= 4, L=10	29.7
4	M= 4, L=10	30.1
5	M= 4, L=10	30.4
6	M= 8, L=20	30.9
7	M= 8, L=20	30.9
8	M=16, L=40	30.9

The "lady's face" is the training image in this experiment. At the beginning of the design process, the (M, L) search parameters were $M = 4$ and $L = 10$ to save computation. However, as the SNR of the coded image increased, higher search intensities were found necessary. At the second phase of the design, M was thus set to 8 and L to 20. Finally, an (M, L) search with $M = 16$ and $L = 40$ was used without significant further improvement. These results are listed in Table III.

Another parameter involved in the tree search is the path separation threshold factor THF incorporated to prevent the near merging described in [10]. In the above iterative procedure, this THF was 0.1, i.e., the branches retained in the search must be THF × Q apart, where Q is the initial quantization step size. If this path separation technique is not used, the performance degrades by a certain amount. For example, if the same VQ table of iteration 6 was used without path separation, the resultant SNR would be 0.6 dB lower.

A comprehensive comparison of several coding schemes is summarized in Table IV. There are three different models in this table, namely, the quarter-plane separable model, the quarter-plane least-squares model, and the nonsymmetric half-plane least-squares model. All the model coefficients were

TABLE IV
CODING PERFORMANCE ON "LADY'S FACE" (1 BIT/PEL)

Coding Scheme	Model	SNR (dB)
DPCM	QP, Separable	24.9
	QP, least-squares	25.2
	NSHP, least-squares	25.5
Predictive Tree Code $M=8$, $L=20$	QP, Separable	27.5
	QP, least-squares	26.3
	NSHP, least-squares	26.3
SBPVQ $M=8$; $L=20$	NSHP, least-squares	30.9
BTPVQ $M=8$, $L=18$	QP, least-squares	30.9

Fig. 11. Test and coded images. (a) Original "lady's face." (b) DPCM: SNR = 24.9 dB. (c) Tree code: $M = 8$, $L = 20$, SNR = 27.5 dB. (d) SBPVQ: $M = 8$, $L = 20$, SNR = 30.9 dB. All coders use 1 bit/pixel quantizers.

obtained using least-squares parameter estimation, except for the separable model where the horizontal and the vertical correlation coefficients were identified separately. These coefficients are summarized in Table II. It is interesting that the best DPCM coding filter may not be the best tree coding filter. Further discussion on the selection of an encoding filter may be found in [26].

We conclude from Table IV that the predictive tree code with a scalar quantizer can provide about 2 dB SNR improvement over DPCM. An additional 3.4 dB improvement is achievable using SBPVQ with a 256 × 1 VQ table. Furthermore, this improvement is visibly striking as shown in Fig. 11 and especially in Fig. 12, an enlarged version of Fig. 11. Since SBPVQ is computationally comparable with a tree

Fig. 12. Enlarged images of Fig. 11. (a)-(d) Same as in Fig. 11.

code, the coding performance is doubled only at the price of an extra VQ table and some additional memory.

B. BTPVQ

This subsection presents the results of three separate simulations. First we present a 1 bit/pixel BTPVQ encoding using the 2-D search order that has been discussed in Section III-B. Second, in order to justify the usefulness of 2-D search, a block tree PVQ encoding with 1-D-like search is presented for comparison. Third, two 0.5 bit/pixel BTPVQ encodings with different block sizes are also presented.

Again, the "lady's face" image was the training data. In the 1 bit/pixel cases, the block size was chosen to be 3×3. Of course, one may want to have larger size blocks to take fuller advantage of the dependence between nearby pixels. But the VQ table size grows exponentially with the block size. For instance, a VQ table for a 3×3 block contains 512 ($= 2^{3\times3}$) vectors and the one for a 4×4 block contains 65536 ($= 2^{4\times4}$) vectors. Hence, we did not try a block size larger than 3×3 at the 1 bit rate.

Because of the rectangular search region, the encoding filter support is restricted to quarter-plane. The model coefficients in all cases are the so-called least-squares, quarter-plane filter which has been given in Table II. As usual, the optimum scalar tree code provided the initial values for the VQ table. The delay length for tree coding was 2 blocks or 18 pixels; i.e., the encoder releases the first block when it reaches the end of the second block.

The 2-D search ordering defined in Fig. 9 was used for the first experiment. The bit allocation pattern for this 1 bit/pixel code was (2, 2, 2, 2, 1) bits per node. Table V shows the SNR results at each iteration in the design phase. We started with $M = 4$ and then increased the search intensity. At iteration 10 with $M = 16$, the coding performance almost saturates. The coded image SNR was compared with that of other schemes in Table IV. This coding result is subjectively and numerically similar to those we have obtained using SBPVQ. This also implies a 3.4 dB improvement over tree codes and a 5.4 dB improvement over DPCM. The coded image is shown in Fig. 13 and is comparable to the one obtained with SBPVQ. Fig. 14 shows enlarged portions of the images of Fig. 13. In comparison with SBPVQ in Section IV-A, the VQ table of this BTPVQ has a larger size with 1252 ($= 4 + 16 \times 2 + 64 \times 3 + 256 \times 2 + 512$) members. But it does not need a register and does not have to load the register for every pixel. The entropy of this code is 7.03 bits for a 3×3 block or 0.78 bits per pixel on average. Hence, we could further reduce the transmission rate by using some entropy coding techniques.

TABLE V
A 1 BIT BTPVQ DESIGN (2-D SEARCH)

Iteration	Search parameters	SNR (dB)
1	M= 4, L=2×3×3	24.3
2	M= 4, L=2×3×3	25.8
3	M= 4, L=2×3×3	28.1
4	M= 4, L=2×3×3	29.4
5	M= 4, L=2×3×3	29.8
6	M= 4, L=2×3×3	30.0
7	M= 4, L=2×3×3	30.3
8	M= 8, L=2×3×3	30.8
9	M= 8, L=2×3×3	30.9
10	M=16, L=2×3×3	31.1
11	M=16, L=2×3×3	31.2

Fig. 13. Coded images. (a) 0.5 bit BTPVQ: $M = 8, L = 2 \times 4 \times 4, 4 \times 4$ block, SNR = 27.5 dB. (b) 1 bit BTPVQ: $M = 8, L = 2 \times 3 \times 3, 3 \times 3$ block, SNR = 30.9 dB.

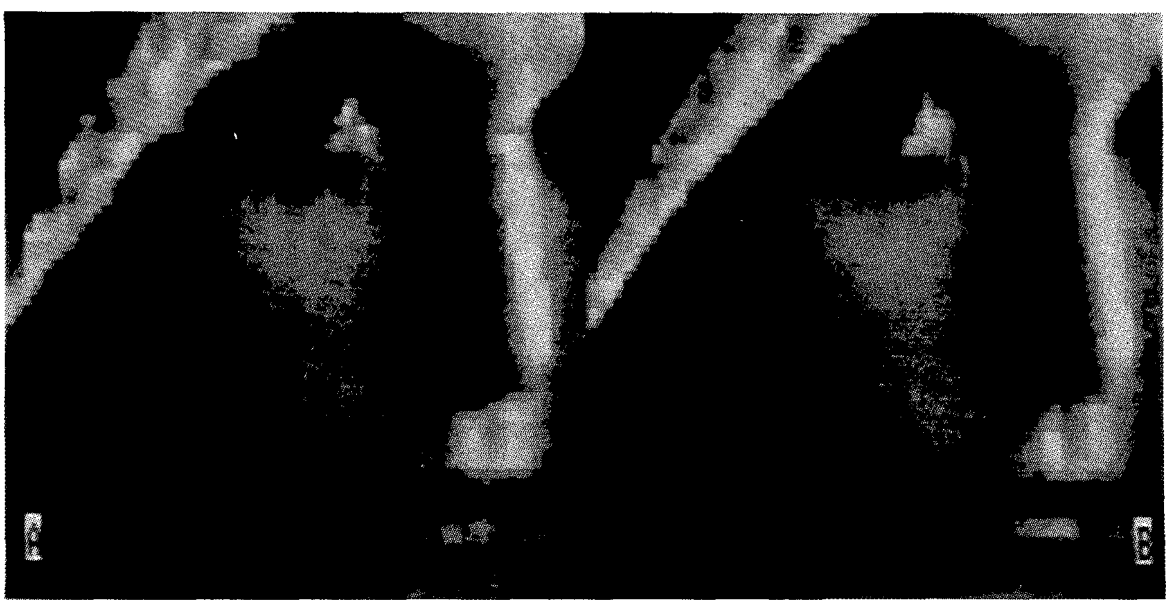


Fig. 14. Enlarged images of Fig. 13. (a), (b) Same as in Fig. 13.

TABLE VI
A 1-BIT BTPVQ DESIGN (1-D-LIKE SEARCH)

Iteration	Search parameters	SNR (dB)
1	M= 4, L=2×3×3	26.2
2	M= 4, L=2×3×3	27.8
3	M= 4, L=2×3×3	28.5
4	M= 4, L=2×3×3	28.7
5	M= 8, L=2×3×3	29.4
6	M= 8, L=2×3×3	29.6
7	M= 8, L=2×3×3	29.8
8	M= 8, L=2×3×3	29.8

TABLE VII
A 0.5 BIT BTPVQ DESIGN (3 × 3 BLOCK)

Iteration	Search parameters	SNR (dB)
1	M= 4, L=2×3×3	23.6
2	M= 4, L=2×3×3	25.4
3	M= 4, L=2×3×3	25.9
4	M= 8, L=2×3×3	26.9
5	M= 8, L=2×3×3	27.1
6	M= 8, L=2×3×3	27.2
7	M=16, L=2×3×3	27.2

As a second simulation we replaced the 2-D search order with the conventional 1-D-like search; i.e., inside each block the coder scans from left to right horizontally and then advances one row after completing the current row. Thus, the total search order for the 3 × 3 block is 9 (rather than 5 as in the 2-D search case). A reasonable bit allocation pattern for this case is 1 bit per node for all tree depths. All the other parameters in this code are the same as in the previous simulation.

The iterative design results are listed in Table VI. We observe that the best performance is 28.7 dB for $M = 4$ and 29.8 dB for $M = 8$. These values are lower than those of the 2-D search (Table V) by 1.6 dB and 1.1 dB for the cases $M = 4$ and $M = 8$, respectively. This agrees with the analysis in [20] and [26]. For a 2-D search region, the redundancy introduced by a 1-D-like search degrades the coding performance especially at very low search intensities.

As a final simulation, we tried to lower the bit rate further. Two 0.5 bit/pixel BTPVQ's were conducted with block size 3 × 3 and 4 × 4, respectively. Since the 1-D-like search is inferior for BTPVQ, we used 2-D search schemes. The coding procedure in this experiment is exactly the same as in the first experiment. The only change is the bit allocation pattern of the code tree. It is (1, 1, 1.5, 1, 0) bits per node for 3 × 3 blocks and (1, 2, 2, 2, 1, 0, 0) bits per node for 4 × 4 blocks. In other words, the VQ table would contain 24 vectors for 3 × 3 blocks and 256 vectors for 4 × 4 blocks, or equivalently, 0.51 bit/pixel and 0.5 bit/pixel on average for these codes.

The numerical coding results of these two nearly 0.5 bit BTPVQ are shown in Tables VII and VIII. The encoded image with 4 × 4 blocks was displayed in Fig. 13. An enlarged version is in Fig. 14. Numerically, a 0.5 bit BTPVQ is slightly better than a 1 bit ordinary tree code. Subjectively, the coded image of the BTPVQ is smoother, but it introduces block-type defects into the reproduced picture.

C. Robustness

One potential concern is the sensitivity of the VQ lookup table used in the coding. If a picture other than the training one is encoded, can we still obtain the same improvement? One may expect that a VQ table designed for smooth images may not work well on a picture filled with fine texture. But a good codebook should work reasonably well on a large variety of pictures with similar spatial characteristics such as human faces and buildings. To test the robustness of PVQ, we applied the coder designed for the "lady's face" to an image of a man's face. These two pictures have quite different statistics as listed in Table IX. Their probability densities (histograms), plotted in Fig. 15, are also quite different, and both are quite non-Gaussian. Furthermore, although they are pictures of

TABLE VIII
A 0.5 BIT BTPVQ DESIGN (4 × 4 BLOCK)

Iteration	Search parameters	SNR (dB)
1	M= 4, L=2×4×4	23.4
2	M= 4, L=2×4×4	25.6
4	M= 4, L=2×4×4	26.4
4	M= 4, L=2×4×4	26.7
5	M= 8, L=2×4×4	27.4
6	M= 8, L=2×4×4	27.5
7	M=16, L=2×4×4	27.8

TABLE IX
STATISTICS OF TEST IMAGES

Image	Mean	Variance
Lady's face	99.18	2734.3
Man's face	113.53	4924.3

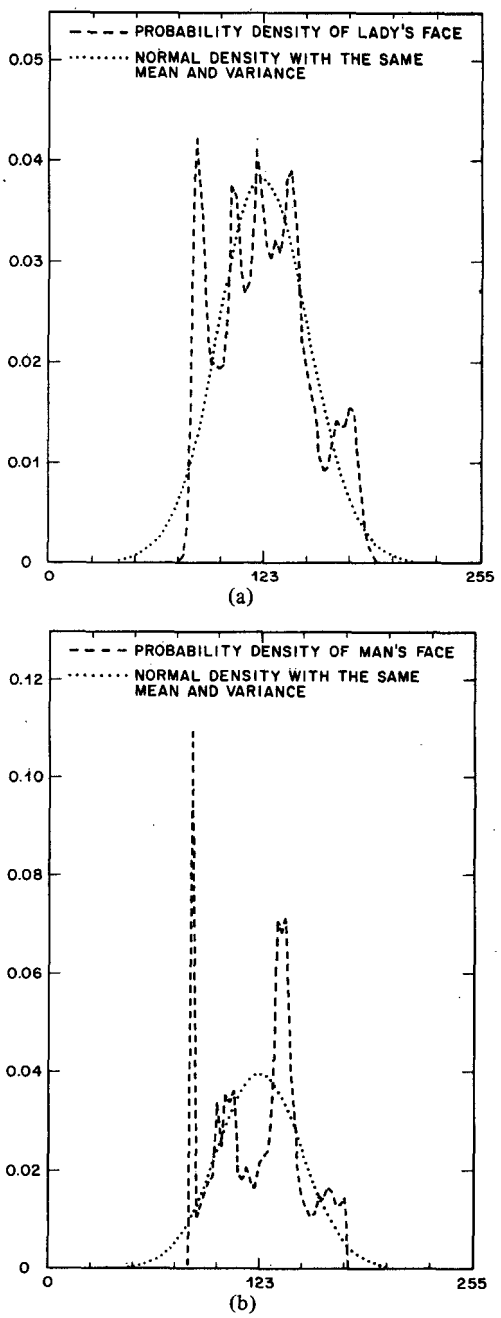


Fig. 15. (a) Normalized histogram of "lady's face" image (density domain). (b) Normalized histogram of "man's face" image (density domain).

TABLE X
CODING PERFORMANCE ON "MAN'S FACE" (1 BIT/PEL)

Coding Scheme	Model	SNR (dB)
DPCM	QP, Separable	26.9
	QP, least-squares	27.3
Predictive Tree Code M=8, L=20	QP, Separable	30.4
	QP, least-squares	29.1
BTPVQ M=8, L=18	QP, least-squares	33.7
SBPVQ M=8, L=20 (from lady's)	NSHP, least-squares	32.3
BTPVQ M=8, L=18 (from lady's)	QP, least-squares	32.5

Fig. 16. Test and coded images. (a) Original "man's face." (b) DPCM: SNR = 26.9 dB. (c) Tree code: $M = 8$, $L = 20$, SNR = 30.4 dB. (d) BTPVQ: $M = 8$, $L = 2 \times 3 \times 3$, SNR = 32.5 dB. All coders use 1 bit/pixel quantizers.

human faces, the man's one has stronger spatial correlation and is relatively easy to compress. The SNR of its ordinary predictive tree coded image is as high as 30 dB, which is hard to achieve for the "lady's face" image.

Both the SBPVQ and the BTPVQ derived from the "lady's face" were applied to the "man's face." We still obtained 1.9 dB and 2.1 SNR improvements in these cases over the tree codes that were specifically designed for them. These values were only about 1 dB lower than the result of a BTPVQ designed exclusively for the "man's face" image. The coding performances of several schemes on the "man's face" image are listed in Table X. Fig. 16 shows the coded images of the man's face. Substantial improvements are visible in Fig. 17, which is the enlarged version of Fig. 16.

Fig. 17. Enlarged images of Fig. 16. (a)-(d) Same as in Fig. 16.

V. Discussion

We have presented several coding schemes which combine predictive tree codes and vector quantizers. Simulations on real images showed significant improvement using these new techniques. Roughly speaking, a predictive VQ can double the coding performance of a tree code at the 1 bit rate, or achieve about the same numerical result with a half-bit rate. From an implementation viewpoint, PVQ has about the same computational complexity as an ordinary tree code. Although PVQ needs some extra storage, due to the advances of VLSI memory devices, this factor would have a minor impact on the overall cost. Because of the use of the (M, L) algorithm, this scheme can be implemented with the same basic building blocks operating in parallel. This identical block structure is most desirable for VLSI circuit layout.

It has been shown by Gray *et al.* [24] that the clustering design algorithm of VQ converges to, at least, a local optimum point; however, the convergence property of PVQ is difficult to analyze because the VQ lies inside the coder loop and the quantization errors are thus correlated with the VQ table. If the correlation between the VQ table and prediction errors is negligible and if an image can be truly modeled by a white noise driven autoregressive source, the prediction errors are then approximately stationary and ergodic when the quantization errors are quite small. This returns to the case in [24]—block ergodic and stationary sources quantized by vector quantizers. Of course, the above argument is heuristic. A rigorous proof of the convergence of the PVQ design scheme has not been obtained. Nevertheless, our simulations indicate that the PVQ iterative design algorithms are well-behaved on real-world image data.

A VQ coding process is basically a pattern matching technique as pointed out by Gersho and Cuperman [27]. The source vector is compared with all the codewords or patterns in the VQ table, and hopefully one of these patterns can match the source vector well. Therefore, if the source vectors of all the test images have similar patterns, then a "good" VQ table which contains almost all these patterns should be robust. Indeed, this is true for PVQ. When the low frequency components of images are removed by the coding filter, the residual images have similar local features. In other words, if the residual images are partitioned into blocks with proper

sizes, these blocks can be grouped into a few representative patterns which represent flat regions and edges with various orientations, positions, and widths. This may explain the robustness of a well-designed PVQ that we obtained in our simulations.

The coding algorithms described in the previous sections can be further modified by using adaptive predictors. Since a real-world image may be modeled as combinations of local edge and nonedge regions, a space varying model seems appropriate for picture coding. The concept of multiple-model for pictures is borrowed from [28], [29]. This multiple-model concept has been found useful in image filtering [28] and adaptive prediction DPCM coding [30]. In a future paper we will report on improved PVQ performance by employing multiple-model prediction.

REFERENCES

[1] J. B. O'Neal, Jr., "Predictive quantizing systems (differential pulse code modulation) for the transmission of television signals," *Bell Syst. Tech. J.*, vol. 45, pp. 689–721, May–June 1966.

[2] J. W. Modestino, V. Bhaskaran, and J. B. Anderson, "Tree encoding of images in the presence of channel errors," *IEEE Trans. Inform. Theory*, vol. IT-27, pp. 677–697, Nov. 1981.

[3] W. K. Pratt, Ed., *Image Transmission Techniques*. New York: Academic, 1979.

[4] A. N. Netravali and J. O. Limb, "Picture coding: A review," *Proc. IEEE*, vol. 68, pp. 366–406, Mar. 1980.

[5] A. K. Jain, "Image data compression: A review," *Proc. IEEE*, vol. 69, pp. 349–389, Mar. 1981.

[6] T. Berger, *Rate Distortion Theory: A Mathematical Basis for Data Compression*. Englewood Cliffs, NJ: Prentice-Hall, 1971.

[7] J. B. Anderson and J. B. Bodie, "Tree encoding of speech," *IEEE Trans. Inform. Theory*, vol. IT-21, pp. 379–387, July 1975.

[8] J. B. Anderson and C. W. Law, "Real-number convolutional codes for speech-like quasi-stationary source," *IEEE Trans. Inform. Theory*, vol. IT-23, pp. 778–782, Nov. 1977.

[9] J. W. Modestino and V. Bhaskaran, "Robust two-dimensional tree encoding of images," *IEEE Trans. Commun.*, vol. COM-29, pp. 1786–1798, Dec. 1981.

[10] H.-M. Hang and J. W. Woods, "Near merging of the paths in suboptimal tree searching," *IEEE Trans. Inform. Theory*, vol. IT-30, pp. 567–573, May 1984.

[11] R. M. Gray, "Vector quantization," *IEEE ASSP Mag.*, pp. 4–29, Apr. 1984.

[12] A. Buzo, A. Gray, Jr., R. M. Gray, and J. D. Markel, "Speech coding based upon vector quantization," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-28, pp. 562–574, Oct. 1980.

[13] A. Gersho and B. Ramamurthi, "Image coding using vector quantization," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, Paris, France, Apr. 1982, pp. 428–431.

[14] B. Ramamurthi, A. Gersho, and A. Sekey, "Low-rate image coding using vector quantization," in *IEEE Global Telecommun. Conf. Rec.*, Nov. 1983, pp. 184–187.

[15] B. Ramamurthi and A. Gersho, "Image vector quantization with a perceptually-based cell classifier," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, Mar. 1984, pp. 32.10.1–32.10.4.

[16] R. L. Baker and R. M. Gray, "Image compression using non-adaptive spatial vector quantization," in *Proc. 16th Asilomar Conf. Circuits, Syst., Comput.*, Pacific Grove, CA, Oct. 1982.

[17] ——, "Differential vector quantization of achromatic imagery," in *Proc. Int. Picture Coding Symp.*, Mar. 1983.

[18] L. C. Stewart, R. M. Gray, and Y. Linde, "The design of trellis waveform coders," *IEEE Trans. Commun.*, vol. COM-30, pp. 702–710, Apr. 1982.

[19] H.-M. Hang and J. W. Woods, "Predictive vector quantization of images," presented at 18th Annu. Conf. Inform. Sci. Syst., Princeton, NJ, Mar. 1984.

[20] H.-M. Hang, "Two-dimensional sequential decision ordering," submitted for publication.

[21] F. Jelinek and J. B. Anderson, "Instrumentable tree coding of information sources," *IEEE Trans. Inform. Theory*, vol. IT-17, pp. 118–119, Jan. 1971.

[22] J. B. Anderson and S. Mohan, "Sequential coding algorithms: A survey and cost analysis," *IEEE Trans. Commun.*, vol. COM-32, pp. 169–176, Feb. 1984.

[23] Y. Linde, A. Buzo, and R. M. Gray, "An algorithm for vector quantizer design," *IEEE Trans. Commun.*, vol. COM-28, pp. 84–95, Jan. 1980.

[24] R. M. Gray, J. C. Kieffer, and Y. Linde, "Locally optimal block quantizer design," *J. Inform. Contr.*, vol. 45, pp. 178–198, May 1980.

[25] R. M. Gray and Y. Linde, "Vector quantizers and predictive quantizers for Gauss-Markov sources," *IEEE Trans. Commun.*, vol. COM-30, pp. 381–389, Feb. 1982.

[26] H.-M. Hang, "Predictive coding of images," Doctoral thesis, Dep. Elec., Comput., Syst. Eng., Rensselaer Polytech. Inst., Troy, NY, July 1984.

[27] A. Gersho and V. Cuperman, "Vector quantization: A pattern-matching technique for speech coding," *IEEE Commun. Mag.*, pp. 15–21, Dec. 1983.

[28] H. Kaufman, J. W. Woods, V. K. Ingle, R. Mediavilla, and A. Radpour, "Recursive image estimation: A multiple model approach," in *Proc. IEEE Conf. Decision Contr.*, Dec. 1979, pp. 812–814.

[29] J. W. Woods, "Kalman filtering in two dimensions," in *Two-Dimensional Digital Signal Processing*, vol. I, T. S. Huang, Ed. New York: Springer-Verlag, 1981.

[30] J. W. Woods and I. Paul, "Adaptive predictive DPCM coding of images," in *Proc. Int. Conf. Commun.*, Seattle, WA, June 1980, pp. 31.8.1–31.8.5.

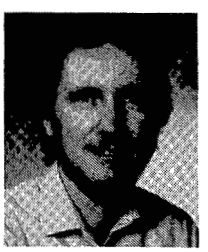
★

Hsueh-Ming Hang (S'79–M'85) was born in Taipei, Taiwan, in 1956. He received the B.S. degree in control engineering and the M.S. degree in electronics from National Chiao-Tung University, Hsinchu, Taiwan, in 1978 and 1980, respectively, and the Ph.D. degree in electrical engineering from Rensselaer Polytechnic Institute, Troy, NY, in 1984.

From 1980 to 1983 he was a Graduate Assistant in the Department of Electrical, Computer, and Systems Engineering, Rensselaer Polytechnic Institute. During 1983–1984 he was an Instructor in the same department, teaching courses in linear and communication systems. Since 1984 he has been with the Visual Communications Research Department, AT&T Bell Laboratories, Holmdel, NJ. His research interests include data compression, information theory, signal processing, and stochastic processes.

Dr. Hang is a member of Sigma Xi.

★



John W. Woods (S'67–M'70–SM'83) was born in Washington, DC, on December 5, 1943. He received the B.S., M.S., E.E., and Ph.D. degrees in electrical engineering from the Massachusetts Institute of Technology, Cambridge, in 1965, 1967, 1967, and 1970, respectively.

From 1970 to 1973, he was at the VELA Seismological Center, Alexandria, VA, working on array processing of digital seismic data. From 1973 to 1976, he was at the Lawrence Livermore Laboratory, University of California, Livermore, working on two-dimensional digital signal processing. Since 1976 he has been with the Department of Electrical, Computer, and Systems Engineering, Rensselaer Polytechnic Institute, Troy, NY, where he is currently Professor. He has taught courses in digital signal processing, probability and stochastic processes, information theory, and communication systems. His research interests include estimation/restoration, detection, recursive digital filtering, and data compression of images and other multidimensional data. He has authored or coauthored over 40 papers in these fields.

Dr. Woods was co-recipient of the 1976 Senior Award of the IEEE ASSP Society. He is a former member of the Digital Signal Processing Committee. He was Chairman of the Schenectady Joint ASSP/Communications Society Chapter in 1977–1978. He is a former Associate Editor for Signal Processing of the IEEE TRANSACTIONS ON ACOUSTICS, SPEECH, AND SIGNAL PROCESSING. He was Co-Chairman of the Third ASSP Workshop on Multidimensional Signal Processing held at Lake Tahoe, CA, October 1983. He is currently Chairman of the ASSP Technical Committee on Multidimensional Signal Processing. He is a member of Sigma Xi, Tau Beta Pi, Eta Kappa Nu, and the American Association for the Advancement of Science.