Variable Fanout Trimmed Tree-Structured Vector Quantization for Multirate Channels[1]

Shawn Herman and Kenneth Zeger
Coordinated Science Laboratory, Dept. of Electrical Engineering, University of Illinois
1308 W. Main St., Urbana, IL 61801 USA.
Email: {herman, zeger}@fire.csl.uiuc.edu

Abstract — **We introduce a generalized pruning technique called trimming which can improve upon the performance of pruned tree-structured vector quantization. The algorithm is used to optimize the tree structure with respect to a multirate channel. Experimental results are supplied which demonstrate this performance.**

I. Introduction

Pruned tree-structured vector quantization (PTSVQ) is a useful technique for progressive image transmission due to its successive approximation character. PTSVQ has also been shown experimentally to have rate-distortion performance close to full search vector quantization (FSVQ) over a substantial range of bit rates [1]. In applications where variable rate coding is acceptable, PTSVQ's reduced encoding complexity makes it an attractive alternative to FSVQ. However, in a packet network environment where bandwidth availability fluctuates in time, it is of interest to have a data compression system that works well when averaged over the available transmission rates. This problem has not previously been addressed with PTSVQ.

In this paper, we introduce a generalized pruning technique called "trimming" which improves upon the performance of PTSVQ. Our algorithm achieves a lower mean-square encoding error than PTSVQ while retaining a tree structure for efficient encoding and progressive transmission. Moreover, the tree structure is optimized with regard to a given histogram of available transmission rates on the channel. The interior nodes of our encoding tree are allowed to have fanouts which are greater than two. For a fixed number of bits spent per input vector, this technique can improve the signal-to-noise ratio at the cost of less flexible progressivity. The main idea is to first grow a large balanced tree with variable fanouts and then trim it back to create useful subtrees.

The idea of using variable fanout trees is not new [2], and, in fact, the process of locating all pruned subtrees which lie on the lower convex hull of the operational distortion-rate function for nonbinary TSVQs was also previously studied in [1]. However, our method of trimming, when applied to a variable fanout TSVQ, generates additional trees (i.e., rate-distortion pairs), some of which often lie below the lower convex hull as found by the generalized BFOS algorithm.

[1]This work was supported in part by the National Science Foundation.

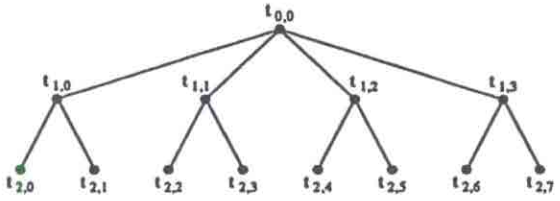


Fig. 1: Example of a variable fanout TSVQ; $t_{i,j}$ denotes the j^{th} node at depth-i within the tree.

II. The Variable Fanout Trimmed TSVQ Algorithm

Although the formulation of the generalized BFOS algorithm presented in [1] assumed a binary tree structure, it was noted that the algorithm did not, in general, require this. By applying the generalized BFOS algorithm to a broader class of tree structures, we are able to exploit a new tree operator which finds an improved collection of subtrees. We define a *variable fanout TSVQ* to be a TSVQ with the property that all nodes at the same depth have the same number of children. Between different tree depths, however, the number of children may vary, but all node fanouts are assumed to be integer powers of two. Because of these constraints, variable fanout TSVQs are special cases of nonbinary TSVQs.

Let T be a variable fanout TSVQ (e.g., see Figure 1). As with binary TSVQ, the binary sequence specifying the path from root to leaf node is an instantaneous code. Since T is balanced, this code is also fixed-rate (all paths beginning at the root of T and ending at a leaf node are described by the same number of bits). The variable fanout tree T is completely specified by the number of bits used to address children of nodes at each depth, represented by the *fanout vector*, $\mathbf{b} = \langle b_0, b_1, \ldots, b_{L-1} \rangle$, where L is the maximum depth of the tree and "depth-0" refers to the root. Thus, each interior node at depth-i has 2^{b_i} children. As an example, the variable fanout TSVQ shown in Figure 1 has $L = 2$ and the fanout vector $\langle b_0 = 2, b_1 = 1 \rangle$.

Let $\phi(R)$ denote the minimum average distortion attainable at rate no greater than R by pruning a given TSVQ. Let C be a random variable representing the

available transmission rate for a channel connecting a PTSVQ encoder to its decoder. Let f_C be the density function of available rates for the channel being used. One measure of performance of a source coder for this channel is the *MSE averaged over available transmission rates*, that is

$$\bar{D} = \int_0^B \phi(r)\, f_C(r)\, dr \qquad (1)$$

where time-sharing between codebooks is allowed. We note that $\bar{D}$ depends on the statistics of both the source and the channel.

A. Design of the Initial Variable Fanout TSVQ

Our method for producing the initial variable fanout TSVQ generalizes the design procedure for balanced binary TSVQ (i.e., "splitting"). For now, we assume a prescribed fanout vector $\mathbf{b} = \langle b_0, b_1, \ldots, b_{L-1} \rangle$. Let $C_N = \{y_0, y_1, \ldots, y_{N-1}\}$ denote an N-point, k-dimensional vector codebook generated by the generalized Lloyd algorithm (GLA) using a training set $\mathcal{V}$. If $t_{i,j}$ is the j^{th} node at depth-i within T, then let $\mathcal{V}^{(t_{i,j})}$ denote all those vectors in $\mathcal{V}$ which are closer to $t_{i,j}$ than any other node at depth-i. Using this notation, $C_n^{(t_{i,j})}$ denotes a size-n codebook designed for node $t_{i,j}$ using $\mathcal{V}^{(t_{i,j})}$.

Design Algorithm

STEP ZERO: Design $C_1^{(t_{0,0})}$ for the root node by finding the centroid of $\mathcal{V}$ (initialize counters $n \leftarrow 1$ and $t_{i,j} \leftarrow t_{0,0}$).

STEP ONE: Split the vectors in $C_n^{(t_{i,j})}$ and design $C_{2n}^{(t_{i,j})}$ by running the GLA with $\mathcal{V}^{(t_{i,j})}$ ($n \leftarrow 2n$).

STEP TWO: If $n < 2^{b_i}$, the current set of vectors split again (go to step one), else assign the codevectors in $C_n^{(t_{i,j})}$ to the $n = 1$ codebooks of each of $t_{i,j}$'s children.

STEP THREE: If all nodes at depth-i have been designed, go to step four, else move laterally to next node ($n \leftarrow 1$, $t_{i,j} \leftarrow t_{i,j+1}$), determine the new set $\mathcal{V}^{(t_{i,j})}$, and go to step one.

STEP FOUR: If $i+1$ is the terminal level of T, then quit, else descend one level in T ($n \leftarrow 1$ and $t_{i,j} \leftarrow t_{i+1,0}$), determine $\mathcal{V}^{(t_{i,j})}$, and go to step one.

The algorithm described above begins by creating a 2^{b_0}-point FSVQ using the GLA. It then repeatedly designs a single 2^{b_i}-point FSVQ for each of the encoding cells generated during the previous iteration ($i = 1, 2, \ldots, L-1$). The procedure for generating balanced binary TSVQs is identical except that 2-point FSVQs are generated at each iteration (i.e., the *else* directive in STEP TWO is always executed). However, greedily grown binary TSVQs do not follow such a procedure since there is no constraint on the number of nodes. Instead, the tree is grown by iteratively splitting the node which yields the maximum ratio of decrease in MSE to increase in average rate [3] [4].

For variable fanout trimmed TSVQ, all intermediate (or additional) codebooks $\{C_n^{(t_{i,j})} : n = 2, 4, \ldots, 2^{b_i - 1} \text{ and } t_{i,j} \notin \tilde{T}\}$ are used for the trimming of T. Note that $\{C_1^{(t_{i,j})} : t_{i,j} \in T\}$ represents the set of codevectors used by the initial tree T, and as such, these are not considered additional codebooks. It can be shown that the storage needed for the variable fanout trimmed TSVQ decoder is identical to that of balanced binary TSVQ at the same maximum coding rate B. Thus, a decoder using either of these TSVQs will need to store the same number of vectors. For greedily grown binary TSVQ, the resulting tree has experimentally been observed to require considerably more storage space at the decoder than variable fanout trimmed TSVQ since the number of nodes is not constrained [4].

B. Trimming of Variable Fanout TSVQ

The generalized BFOS algorithm is an efficient method of locating the pruned subtrees which determine the lower convex hull of the operational distortion-rate function. However, the generalized BFOS algorithm can be modified to exploit the additional codebooks stored with variable fanout trimmed TSVQ, and thus potentially generate an even *lower* convex hull. The tree operation of trimming provides this extension. We again let T be a variable fanout TSVQ with fanout vector $\mathbf{b}$. Let $t_{i,j}$ denote the j^{th} node at depth-i within T. We define $\beta(t_{i,j})$ as the current fanout of node $t_{i,j}$, so that before trimming, $\beta(t_{i,j}) = 2^{b_i}$ for all interior nodes. In order to trim node $t_{i,j}$, the procedure below is followed:

Trimming a Node

STEP ONE: Prune any branches extending from $t_{i,j}$'s children.

STEP TWO: Reduce $t_{i,j}$'s children by a factor of $1/2^m$ where m is a positive integer such that $2^m \leq \beta(t_{i,j})$ (i.e., $\beta(t_{i,j}) \leftarrow \beta(t_{i,j})/2^m$).

STEP THREE: If $\beta(t_{i,j}) = 1$, prune at node $t_{i,j}$, else replace the codevectors from the remaining children with the vectors stored in $C_{\beta(t_{i,j})}^{(t_{i,j})}$.

The codebook replacement in STEP THREE guarantees that $t_{i,j}$'s new children generate the minimum MSE partition (as determined by the GLA) for its input cell. Without such replacement, the codevectors of the remaining children would no longer satisfy the centroid condition. Figure 2 shows a subtree of a variable fanout TSVQ with $\mathbf{b} = \langle 2, 2, 1 \rangle$ which could be produced by this trimming procedure. The dashed portion of the tree identifies the nodes which are removed in order to trim $t_{1,2}$ using $m = 1$. Note that during trimming, the remaining two children of $t_{1,2}$ will have their codevectors replaced by the 2-point FSVQ which was created

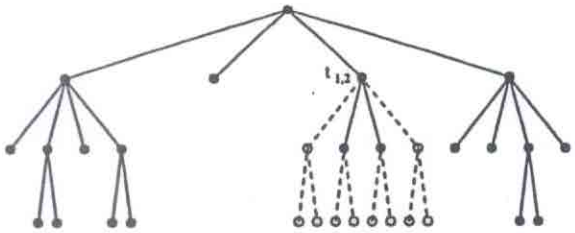


Fig. 2: Example of a subtree for variable fanout trimmed TSVQ . The dashed lines indicate a possible way to trim at node $t_{1,2}$.

for $t_{1,2}$ during the design of the initial variable fanout tree.

Let $\mathcal{P}_T$ denote the set of all subtrees which can be created by pruning some initial tree T. Likewise, let $\mathcal{T}_T$ denote the set of all subtrees which can be created by trimming T. It is important to observe that when the *if* directive in STEP THREE of the trimming procedure is executed, trimming is identical to pruning. This is because STEP ONE and STEP TWO only affect tree nodes which are then removed by the pruning in STEP THREE. Thus, trimming is a generalization of pruning. This implies $\mathcal{P}_T \subseteq \mathcal{T}_T$. In words, *any* pruned subtree of T can be generated by some trimming sequence. However, there exist, in general, trimmed subtrees of T which cannot be generated by any pruning sequence. It is the subtrees in $\mathcal{T}_T \setminus \mathcal{P}_T$ which can enable variable fanout trimmed TSVQ to outperform PTSVQ in a rate-distortion sense. For this to occur, there needs to be at least one $S \in \mathcal{T}_T$ that generates a rate-distortion pair which lies below the lower convex hull of $\mathcal{P}_T$. Even if there is no such S in $\mathcal{T}_T \setminus \mathcal{P}_T$, since $\mathcal{P}_T \subseteq \mathcal{T}_T$, variable fanout trimmed TSVQ will always perform at least as well as PTSVQ.

It can be shown, in a manner similar to [1], that the trimmed subtrees lying on the lower convex hull of the operational distortion-rate function are nested in the sense that each can be obtained from its higher rate neighbor through a sequence of trims. Since the rate-distortion optimal subtrees are nested, they can be generated efficiently for use at both the encoder and decoder from the initial variable fanout tree structure and its additional codebooks. Numeric tags within each node can be used to store the optimal trimming sequence [5].

C. Choosing the Fanout Vector to Minimize $\bar{D}$

So far, we have assumed that the fanout vector b was fixed. However, it can be shown that for a given maximum rate constraint B, there exist 2^{B-1} possible fanout vectors. This introduces an extra degree of freedom which is not present in PTSVQ systems. Since each fanout vector b yields a different initial tree structure, there is generally a unique lower convex hull associated with each b. If the allowable transmission rates for a given channel are known, the fanout vector yielding the trimmed subtrees with the lowest MSEs at or near these rates should be chosen. Likewise, if a probability distribution for available transmission rates, f_C, is either known or can be estimated for the channel, the fanout vector which minimizes $\bar{D}$ (see eq. (1)) should be chosen.

Since the variable fanout trimming process generates rate-distortion pairs which determine the lower convex hull, a numeric integration subroutine can be embedded within the algorithm in order to approximate $\bar{D}$ for a given rate density f_C. Unfortunately, as the maximum rate B grows, it may become computationally infeasible to design all 2^{B-1} variable fanout TSVQs in order to minimize $\bar{D}$. The following heuristic approach can be shown experimentally to be satisfactory.

Given a transmission rate density f_C with peaks at $r_1, r_2, \ldots, r_n$, the initial tree structures which minimize $\bar{D}$ will be those which satisfy

$$\left(r_j - \frac{1}{2}\right) \leq \sum_{i=0}^{m_j} b_i \leq \left(r_j + \frac{1}{2}\right) \qquad (2)$$

for each $j = 1, 2, \ldots, n$ and $m_j \in \{0, 1, 2, \ldots, L-1\}$. For example, if f_C is bimodal with peaks at 5 bits/vector and 8 bits/vector, fanout vectors such as $\langle b_0 = 5, b_1 = 3, \ldots \rangle$ or $\langle b_0 = 1, b_1 = 4, b_2 = 3, \ldots \rangle$ would likely perform well. Thus, if computational resources are limited, these tree structures alone can be generated and tested.

III. Experimental Results

We now consider several examples which demonstrate that the heuristic method introduced in Section II.C for choosing b given f_C yields tree structures which minimize $\bar{D}$. To accomplish this, we list the 10 minimum-$\bar{D}$ fanout vectors for various channel rate distributions. The variable fanout tree structures were designed for a maximum rate of $B = 10$ bits/vector using two 512×512 images ("Man" from the USC database and "Goldhill" from the RPI database) and vector dimension $k = 4$. Let $f_{C,0}$ be uniform on $R \in [0, 10]$. Let $f_{C,1}$ and $f_{C,2}$ both be Gaussian with $\sigma^2 = 0.01$, $\mu_1 = 5.5$ bits/vector, and $\mu_2 = 8.0$ bits/vector. Let $f_{C,3} = \frac{1}{2}(f_{C,1} + f_{C,2})$, a bimodal distribution. We evaluated $\bar{D}$ for 444 of the 512 possible variable fanout trees. The results are presented in Table 1.

For $f_{C,0}$, the uniform distribution, we find that most of the minimum-$\bar{D}$ fanout vectors begin with 2 or 3 unitary allocations. This assures that there are trimmed subtrees at low bit rates. This is important because the MSE is greatest for subtrees with small R. For $f_{C,1}$, the Gaussian distribution with $\mu_1 = 5.5$ bits/vector, we find that the minimum-$\bar{D}$ fanout vectors all have $\mathrm{b} = \langle 1, 4, \ldots \rangle$. It is somewhat surprising that trees with $b_0 = 5$ are not ranked highest for

Tab. 1: Fanout vectors which minimize $\bar{D}$ on the training images "Man" and "Goldhill" for various available rate distributions ($k = 4$ and $B = 10$ bits/vector).

	Fanout Vector b			
Rank	$f_{C,0}$	$f_{C,1}$	$f_{C,2}$	$f_{C,3}$
1	$\langle 1,1,1,3,4\rangle$	$\langle 1,4,2,1,2\rangle$	$\langle 7,3\rangle$	$\langle 1,4,3,2\rangle$
2	$\langle 1,1,4,4\rangle$	$\langle 1,4,2,3\rangle$	$\langle 1,6,3\rangle$	$\langle 1,4,2,3\rangle$
3	$\langle 1,1,1,3,2,2\rangle$	$\langle 1,4,2,2,1\rangle$	$\langle 7,1,2\rangle$	$\langle 1,4,2,1,2\rangle$
4	$\langle 1,4,2,3\rangle$	$\langle 1,4,2,1,1,1\rangle$	$\langle 1,6,1,2\rangle$	$\langle 1,4,3,1,1\rangle$
5	$\langle 1,1,1,3,1,3\rangle$	$\langle 1,4,3,2\rangle$	$\langle 7,2,1\rangle$	$\langle 1,4,2,2,1\rangle$
6	$\langle 1,1,4,1,3\rangle$	$\langle 1,4,3,1,1\rangle$	$\langle 1,6,2,1\rangle$	$\langle 1,4,2,1,1,1\rangle$
7	$\langle 1,1,1,3,3,1\rangle$	$\langle 1,4,1,2,2\rangle$	$\langle 1,1,5,3\rangle$	$\langle 5,3,2\rangle$
8	$\langle 1,1,3,2,3\rangle$	$\langle 1,4,1,1,3\rangle$	$\langle 1,2,4,3\rangle$	$\langle 5,2,3\rangle$
9	$\langle 1,4,2,1,2\rangle$	$\langle 1,4,1,2,1,1\rangle$	$\langle 7,1,1,1\rangle$	$\langle 1,4,5\rangle$
10	$\langle 1,1,4,2,2\rangle$	$\langle 1,4,1,3,1\rangle$	$\langle 1,5,4\rangle$	$\langle 5,2,1,2\rangle$

Tab. 2: PSNR gain of variable fanout trimmed TSVQ compared to unbalanced binary PTSVQ on individual test images for two different channel densities.

	$f_{C,1}$		$f_{C,2}$	
Test Image	Bits/Level	Gain (dB)	Bits/Level	Gain (dB)
"Plane"	111322	1.433	1243	-0.652
"Woman"	1144	0.810	1243	0.917
"Peppers"	1423	2.364	1612	0.612

$f_{C,1}$, since their trimmed subtrees are approximately size-32 FSVQs at $R \approx 5$ bits/vector. However, such trees only have trimmed subtrees at integer rates below $R = 5$ bits/vector (i.e., $R = 4, 3, 2, 1, 0$). Furthermore, we note that for $f_{C,1}$, the fanout vectors whose $\bar{D}$ values ranked 18–32 smallest (not included in Table 1) all have $\mathbf{b} = \langle 5, \ldots \rangle$ while those from 33–44 all have $\mathbf{b} = \langle 2, 3, \ldots \rangle$. Thus, a tree's performance is closely related to whether nodes are located at the channel's peak rates. The results in Table 1 for $f_{C,2}$ can be explained in a manner similar to $f_{C,1}$ (note that none of the top 10 fanout vectors have $b_0 = 8$ for $f_{C,2}$). Finally, $f_{C,3}$ illustrates the importance of performing well at those rates which contribute most to $\bar{D}$ since many of the best tree structures for $f_{C,3}$ are also optimal for $f_{C,1}$. If $f_{C,1}$ and $f_{C,2}$ had been weighted differently in creating $f_{C,3}$, this would not necessarily be the case.

We now quantify how well variable fanout trimmed TSVQ can perform on a single image using the peak signal-to-noise ratio ($\text{PSNR} = 10 \log_{10}(255^2/\bar{D})$). In Table 2, we list the maximum PSNRs achieved on each of three test images ("Plane," "Woman," and "Peppers" from the USC database) using the fanout vectors from the first three columns and the top eight rows of Table 1. All PSNRs in Table 2 are relative to the PSNR achieved by unbalanced binary PTSVQ on the corresponding test image (unbalanced binary PTSVQ is chosen since it typically outperforms both balanced binary PTSVQ and variable fanout *pruned* TSVQ). We find that in many cases, variable fanout trimmed TSVQ can outperform unbalanced binary PTSVQ by a substantial margin. For example, coding "Peppers" with $\mathbf{b} = \langle 1, 4, 2, 3 \rangle$ provides a PSNR gain of over 2 dB. However, "Plane" transmitted over the channel with density $f_{C,2}$ is an exception, a result which may be due to unbalanced binary PTSVQ's substantially larger codebook at higher rates.

References

[1] P. A. Chou, T. Lookabaugh, and R. M. Gray, "Optimal pruning with applications to tree-structured source coding and modeling," *IEEE Trans. Inform. Theory*, vol. 35, no. 2, pp. 299–315, 1989.

[2] D. Y. Wong, B.-H. Juang, and A. H. Gray, Jr., "An 800 bit/s vector quantization LPC vocoder," *IEEE Trans. Acoust. Speech Signal Process.*, vol. 30, no. 5, pp. 770–779, 1982.

[3] P. A. Chou, "Application of information theory to pattern recognition and the design of decision trees and trellises," Ph.D. dissertation, University of California, Stanford, CA, 1988.

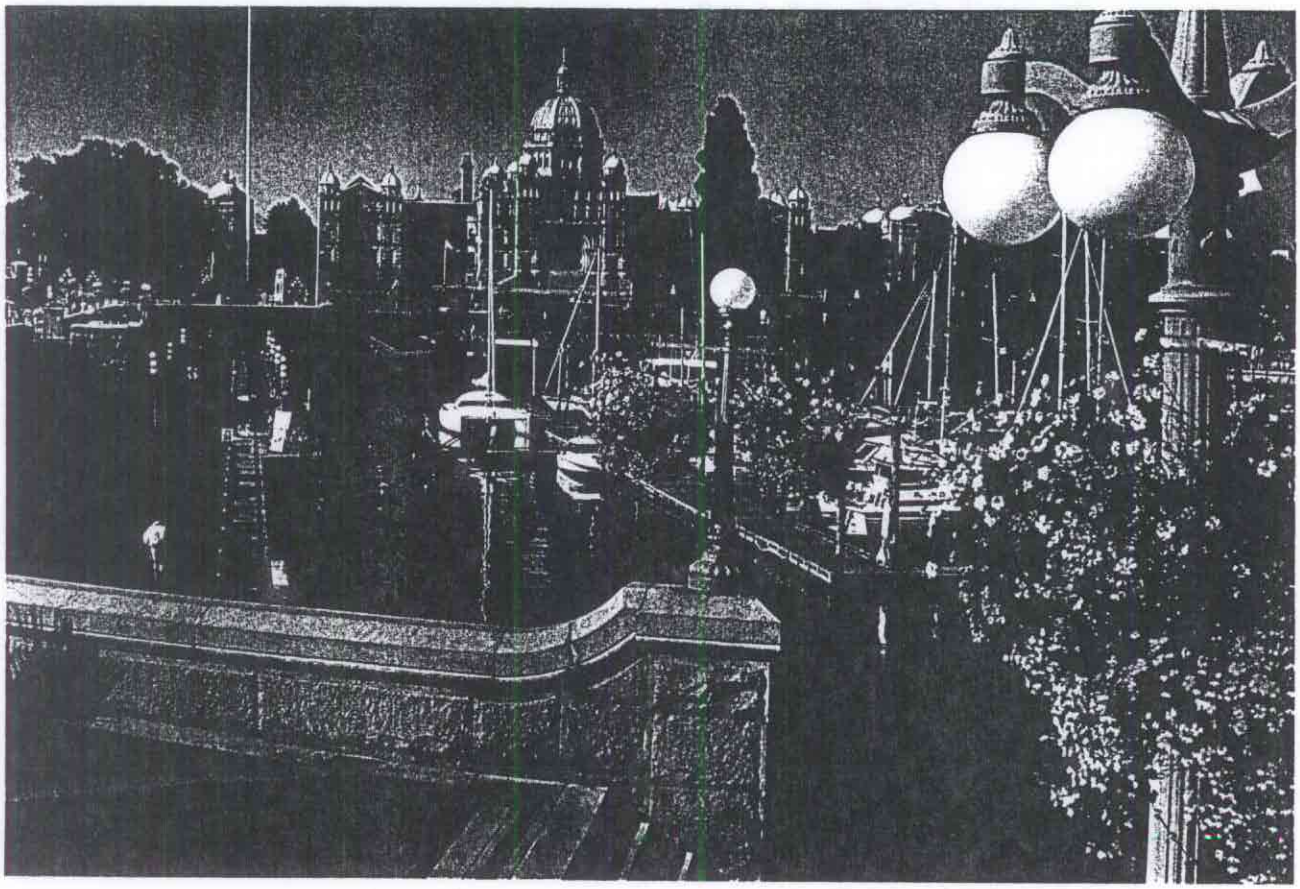
[4] E. A. Riskin and R. M. Gray, "A greedy tree growing algorithm for the design of variable rate vector quantizers," *IEEE Trans. Signal Process.*, vol. 39, no. 11, pp. 2500–2507, 1991.

[5] S.-Z. Kiang, R. L. Baker, G. J. Sullivan, and C.-Y. Chiu, "Recursive optimal pruning with applications to tree structured vector quantizers," *IEEE Trans. Image Process.*, vol. 1, no. 2, pp. 162–169, 1992.

1996 IEEE International Symposium on Information Theory and Its Applications

Victoria, B.C., Canada
September 17–20, 1996

PROCEEDINGS - Volume I



SITA

SPONSORED BY: The Society for Information Theory and Its Applications

IEEE

EIC

In cooperation with IEEE IT Society, IEICE, and the Canadian Society for Information Theory