CONDITIONAL ENTROPY-CONSTRAINED TREE-STRUCTURED VECTOR QUANTIZATION WITH APPLICATIONS TO SOURCES WITH MEMORY*

Ligang Lu and *William A. Pearlman*†

Abstract

An algorithm is derived for designing tree-structured vector quantizers to encode sources with memory. The algorithm minimizes the average distortion subject to a conditional entropy constraint and the tree structure restriction. This technique called conditional entropy constrained tree-structured vector quantization (CECTSVQ) can more efficiently exploit the source memory. This work is an extension of the recent work by Balakrishnan, Pearlman, and Lu[6] for the variable rate tree-structured vector quantization. The tree is grown by applying a rate-constrained iterative splitting process to achieve the maximum attainable ratio of the decrease in the average distortion over the increase in the conditional entropy of the output of the quantizer. The iterative splitting process and thus the design algorithm are shown to converge. A fast algorithm is also developed to dramatically reduce the computational cost while maintaining virtually the same performance as the ideal

[1]This material is based on work supported by the National Science Foundation under Grant No. NCR-9004758. The government has certain rights in this material.

[2]Ligang Lu is with IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598, lul@us.ibm.com and William A. Pearlman is with the Electrical, Computer and System Engineering Department, Rensselaer Polytechnic Institute, Troy, NY 12180-3590, pearlman@ecse.rpi.edu.

algorithm. Experiments on sampled speech and synthetic sources have shown that, at rates under 1 bit/sample and under the square error distortion measure, more than 2dB gain in the signal-to-quantization noise ratio (SQNR) can be achieved over the full-search entropy-constrained vector quantization (ECVQ)[4] and the entropy-constrained tree-structured vector quantization (ECTSVQ)[6, 17] on the speech source; about 1dB improvement over ECVQ and 1.2 dB over ECTSVQ on the synthetic sources can be achieved. Furthermore, in the entire tested rate range, CECTSVQ can be expected to have performance very close to that of the conditional entropy-constrained full-search vector quantization (CECVQ)[2].

Index Terms — Source coding, vector quantization, rate-distortion theory, tree-structured vector quantization, variable rate coding, entropy coding, conditional entropy coding.

I Introduction

Consider the compression of a discrete-time stationary ergodic random vector source with memory using vector quantization(VQ). The codeword index sequence of the quantizer output will contain inter-codeword correlation since there is memory between the contiguous source vectors. On the other hand, information theory[1] has shown that side information may reduce average uncertainty, or in another words, conditioning can possibly decrease the entropy, that is

$$H(X|Y) \leq H(X) \tag{1}$$

with equality if and only if X and Y are independent, where X and Y are random variables of the output of the source and $H(X)$ and $H(X|Y)$ are the entropy of X and the conditional entropy of X given Y, respectively. Therefore a vector quantizer which appropriately exploits this inter-codeword correlation could be expected to have a performance gain over ones that do not. Recently Chou and Lookabaugh[2] proposed such an algorithm to uti-

lize the inter-codeword memory in designing unstructured, i.e., full-search vector quantizers. Their algorithm, called *conditional entropy constrained vector quantization*(CECVQ), is an extension of the full-search entropy constrained vector quantization(ECVQ) algorithm[4]. In the full-search CECVQ design algorithm, the average distortion is minimized subject to the constraint of the first-order conditional entropy of the codewords instead of their first-order unconditional entropy as in the full-search ECVQ algorithm. Each output of the quantizer is then losslessly encoded by an entropy coder matched to the conditional probability distribution of that output. Recently Garrido and Pearlman developed a pairwise nearest neighbor merging algorithm to provide an alternative for the design of full-search CECVQ[3]. It has been shown[2, 3] that CECVQ can achieve significant rate-distortion performance gain over VQ and ECVQ.

Tree-structured vector quantization(TSVQ), first proposed by Buzo, *et al.*[10], is a form of VQ with significantly lower computational complexity. The codebook of a tree-structured vector quantizer is organized as a tree, usually a binary tree, to facilitate the codebook search. When mapping a source vector onto an output codevector, the operation starts at the root node of the tree and determines which child of it minimizes the adopted distortion measure. The operation repeats from that child node until a leaf node is reached. Therefore the search complexity is reduced to $\mathcal{O}(\log \mathcal{M})$ from $\mathcal{O}(\mathcal{M})$ for an unstructured codebook of size M. Obviously, the performance of TSVQ is in general inferior to that of unstructured VQ of the same rate since a source vector may not be mapped onto the optimum codevector because of the imposed structure.

The tree-structured vector quantizers in [10] were designed one level at a time using the splitting method of the generalized Lloyd algorithm(GLA)[9]. The grown tree is a balanced tree that implements a fixed rate code. More recent efforts in TSVQ have been devoted to designing variable rate tree-structured vector quantizers to improve the coding efficiency. Makhoul, Roucos, and Gish[11] and Lindsay[12] grow the tree one node at a time by splitting the leaf node with the highest distortion. However, this splitting approach guarantees neither the greatest distortion decrease nor the best rate-distortion trade-off.

A better approach of growing variable rate tree-structured vector quantizers has been proposed by Riskin and Gray[13]. This approach differs from the previous one in that, instead of splitting the leaf node having the largest distortion, it grows the tree by splitting the leaf node which has the largest ratio of the decrease in distortion over the increase in rate. But the splitting in their approach is not rate-constrained, consequently the largest attainable ratio is not actually achieved in the splitting. Another approach for designing variable rate tree-structured vector quantizers was proposed by Chou, Lookabaugh, and Gray[7]. They first grow a balanced fixed rate tree-structured vector quantizer as in [10], then optimally prune the tree using the generalized Breiman, Friedman, Olshen, and Stone (BFOS) algorithm[14]. This technique is called the pruned TSVQ (PTSVQ). However, the generalized BFOS tree pruning algorithm can only produce a best subtree, which corresponds to a tree-structured codebook, from the given initial tree. Therefore, it is important to investigate better tree growing algorithms in the design of tree-structured vector quantizers. Recently, Balakrishnan, Pearlman, and Lu developed an algorithm for designing variable rate memoryless tree-structured vector quantizers[6]. The tree is grown by an iterative rate-constrained splitting process to achieve the maximum attainable ratio of the decrease in distortion over the increase in rate in each growing step. The variable rate TSVQ has been shown capable of good performance in many applications[6, 7, 8, 15, 16].

In this paper, we generalize the algorithm in [6] to design tree-structured vector quantizers which have memory. This technique, called *conditional entropy-constrained tree-structured vector quantization* (CECTSVQ), can more efficiently exploit the source memory. We shall first formulate the problem of designing the conditional entropy constrained tree-structured vector quantizers. Then we develop an ideal design algorithm and based on that derive a fast algorithm. We shall also show the convergence of the iterative splitting algorithm. Finally, we compare the performance of CECTSVQ against CECVQ, ECVQ, and the entropy-constrained TSVQ (ECTSVQ)[6, 17] based on results with synthetic and speech sources.

II The Preliminaries

Let the K-dimensional vector source $\{X_0, X_1, X_2, \ldots\}$ be a strictly stationary ergodic discrete-time random process. The random vector X can be viewed as a point in the K-dimensional Euclidean space R^K and is statistically described by a probability density function $f_X(X)$. Consider encoding X using a tree-structured vector quantizer T^J. Without losing generality, we concentrate in this paper on the binary tree case for simplicity. The development can be easily extended to an M-ary tree. Let T^J have L^J leaf nodes and hence $L^J - 1$ interior nodes. Associated with each leaf node there is a reproduction codevector and with each interior node an auxiliary codevector. Denote the leaf node set of T^J as $\tilde{L}^J = \{1, 2, \ldots, L^J\}$. Let $Y_l^J, l = 1, 2, \ldots, L^J$, be the reproduction codevector associated with the leaf node l. The encoding of the source vectors X using T^J is a sequence of refining quantization operations. Starting from the root node, X is repeatedly assigned to a child node of the current node that minimizes the distortion until X reaches a leaf node. Each assignment is a refinement of the previous one. The source vector X is represented by the codevector associated with the leaf node. This encoding process reflects the successive refining property of TSVQ and its difference from the unstructured VQ. To facilitate the illustration of the conditional encoding, we define a state variable

$$S^j(X_k, X_{k-1}, X_{k-2}, \ldots) \tag{2}$$

to represent the node in the jth layer of the tree, to which the source vector X_k is coded. For convenience, we use $S^j(X_k)$ as a short notation to represent the state variable in (2). Using the state variable the prior probability P_l^j that a source vector X_k is mapped onto a node l and represented by Y_l^j can be expressed by

$$P_l^j = P\{S^j(X_k) = l\}, \quad k = 0, 1, 2, \ldots. \tag{3}$$

Let P_{ml}^j be the joint probability that two contiguous source vectors X_{k-1} and X_k are mapped to the codevectors Y_m^j and Y_l^j in order. P_{ml}^j can be defined as

$$P_{ml}^j = P\{S^j(X_{k-1}) = m, S^j(X_k) = l\} \tag{4}$$

Denote $P^j_{l|m}$ the conditional probability of a source vector X to be represented by Y^j_l, given that its predecessor was mapped to Y^j_m, for $l, m = 1, 2, \ldots, L^j$. Then, from Bayes's Theorem of probability theory,

$$P^j_{l|m} = \frac{P^j_{ml}}{P^j_m}. \tag{5}$$

Let $d(X, Y)$ be the distortion measure resulting from reproducing X as Y, where $d(.)$ is a non-negative function defined on the source vector alphabet and the reproduction alphabet. Starting from the root node, the encoding of X using T^j is to successively map X to one of its child nodes according to the *nearest neighbor rule* until X is mapped onto a leaf node and represent X by the reproduction codevector $Y_l{}^J$. Let i be a non-leaf node of T^J and i_1 and i_2 be the child nodes of i. The mapping is defined by the *nearest neighbor rule*: assign X to i_k if

$$d(X, Y^j_{i_k}) + \lambda_i r^j_{i_k} \leq d(X, Y^j_{i_m}) + \lambda_i r^j_{i_m}; k \neq m, \tag{6}$$

where $k, m = 1, 2$; and r^j_{i1} and r^j_{i2} are the rates to encode X as Y^j_{i1} and Y^j_{i2}, respectively. The parameter λ_i is a Lagrange multiplier associated with i, which controls the relative importance of the rate and distortion. The conditional distortion of encoding all source vectors by $Y_l{}^J$ is the conditional expectation of $d(X, Y_l{}^J)$,

$$D^J_l = E\{d(X, Y_l{}^J) | S^J(X) = l\}. \tag{7}$$

The overall average distortion of encoding the source X by T^J can be then evaluated as

$$D^J = E\{D^J_l\} = \sum_{l \in \tilde{L}^J} P^J_l D^J_l = \sum_{l \in \tilde{L}^J} E\{d(X, Y_l{}^J) | S^J(X) = l\}. \tag{8}$$

If the rate of encoding a source vector by a codevector Y^j_l is defined to be the self-entropy $-\log_2 P^j_l$, the average rate per vector of encoding the source by T^J can be measured by the first-order entropy of the quantizer output

$$R^J = H^J = -\sum_{l \in \tilde{L}^J} P^J_l \log_2 P^J_l. \tag{9}$$

To exploit the inter-codeword correlation we define the rate of encoding a source vector X_k by $Y_l{}^J$, given its predecessor X_{k-1} was encoded by Y^J_m, to be the conditional self-entropy

$-\log_2 P^J_{l|m}$. Then the average rate per vector for encoding those source vectors whose predecessors were encoded by Y^J_m is

$$R^J_m = H^J_m = -\sum_{l \in \tilde{L}^J} P^J_{l|m} \log_2 P^J_{l|m}. \tag{10}$$

The overall average coding rate per vector is the first-order conditional entropy of the quantizer output

$$R^J = H^J = \sum_{m \in \tilde{L}^J} P^J_m H^J_m = -\sum_{m \in \tilde{L}^J} \sum_{l \in \tilde{L}^J} P^J_{ml} \log_2 P^J_{l|m}. \tag{11}$$

In the rest of this paper we also refer R^J as the conditional entropy of the tree T^J.

Vector quantizers designed by minimizing the functional

$$J = D + \lambda R \tag{12}$$

with the self-entropy as the definition of rate in the *nearest neighbor rule* (6) are called entropy-constrained vector quantizers. Likewise, if the rate is defined as the first-order conditional self-entropy, the designed quantizers are called the conditional entropy-constrained vector quantizers. Our objective is then to design a tree-structured vector quantizer which minimizes the overall average distortion subject to the overall average rate constraint and the tree structure restriction. Since the tree-structured quantizers have the property that an optimal quantizer of higher rate embeds optimal quantizers of smaller rates[6], it is reasonable to design a higher rate tree-structured quantizer by growing and extending successively lower rate tree-structured quantizers. Consider obtaining a tree T^{J+1} of higher rate from T^J by splitting a leaf node i of T^J. This splitting should result in a tree T^{J+1} from T^J with the best trade off between the overall average distortion decrease and the average rate, i.e., the conditional entropy increase. We can apply a non-memoryless nearest neighbor rule in the splitting to achieve our objective. That is, when we split a leaf node i of T^J into i_1 and i_2 to obtain a new tree T^{J+1}, we examine each source vector X_k that has been mapped onto i. Suppose that X_k's predecessor X_{k-1} was coded by a reproduction codevector $Y_l{}^J$; if $l \neq i$, l is now also a node in the layer $J+1$ with the codevector Y_l^{J+1} and otherwise, if $l = i$, X_{k-1} would have been re-assigned to either $Y_{i_1}^{J+1}$ or $Y_{i_2}^{J+1}$, so, in this case, l now refers to either i_1 or i_2. We

assign X_k to $Y_{i_1}^{J+1}$ or $Y_{i_2}^{J+1}$ using a non-memoryless biased distortion measure, i.e., assign X_k to $Y_{i_1}^{J+1}$ if

$$d(X_k, Y_{i_1}^{J+1}) - \lambda_t^J \log_2 P_{i_1|l}^{J+1} \leq d(X_k, Y_{i_2}^{J+1}) - \lambda_t^J \log_2 P_{i_2|l}^{J+1}, \tag{13}$$

where $P_{i_m/l}^{J+1}, m = 1, 2$, is the probability that X_k is mapped onto i_m given that X_{k-1} was mapped onto l and t in λ_t^J is an iteration index. Since the trees grown by splitting the leaf nodes will have discrete distortions and rates, the slope of the line joining points (D^J, R^J) and (D^{J+1}, R^{J+1}) in the distortion-rate plane is

$$\lambda_{t+1}^J = \frac{D^J - D^{J+1}}{R^{J+1} - R^J} = \frac{\Delta D^J}{\Delta R^J}, \tag{14}$$

where ΔD^J and ΔR^J are the decrease in the distortion and the increase in the rate resulting from the Jth splitting, respectively:

$$\begin{aligned} \Delta D^J &= D^J - D^{J+1} = E\{d(X, Y_i^J)|S^J(X) = i\} - E\{d(X, Y_{i_1}^{J+1})|S^{J+1}(X) = i_1\} \\ &\quad -E\{d(X, Y_{i_2}^{J+1})|S^{J+1}(X) = i_2\}, \end{aligned} \tag{15}$$

and

$$\Delta R^J = R^{J+1} - R^J = -\sum_{m=1}^{L^{J+1}} \sum_{l=1}^{L^{J+1}} P_{ml}^{J+1} \log_2 P_{l|m}^{J+1} + \sum_{m=1}^{L^J} \sum_{l=1}^{L^J} P_{ml}^J \log_2 P_{l|m}^J. \tag{16}$$

Obviously, the best tree T^{J+1} resulting from the splitting of a leaf node is the one that maximizes λ_{t+1}^J. Hence our objective can be fulfilled by finding a leaf node in T^J which maximizes λ_{t+1}^J and split that node according to (13). To do this, one needs to first find the maximum $\lambda_{l,max}^J$ attainable from the splitting a leaf node l, for $\forall l \in \tilde{L}^J$, then choose the leaf node i to split that achieves $\lambda_{i,max}^J \geq \lambda_{l,max}^J$, for $\forall l \in \tilde{L}^J$. However, the maximum attainable $\lambda_{l,max}^J$ for each leaf node is , in general, not known. To find $\lambda_{l,max}^J$, we can start with $\lambda_{l,max}^J = 0$ and iteratively split l and update $\lambda_{l,max}^J$ until it stops increasing. In the following, we develop such an algorithm for design of conditional entropy-constrained tree-structured vector quantizers.

III The CECTSVQ Design Algorithm

In this section, we present the algorithm for designing the conditional entropy constrained tree-structured vector quantizers. The algorithm grows a tree-structure vector quantizer by splitting a leaf node at a time. The resulting tree-structured vector quantizer minimizes the overall average distortion constrained by the conditional entropy and the tree structure restriction. Before we start to describe the algorithm, let us formally define our conditional entropy-constrained tree-structured vector quantizer.

Definition: A K-dimensional first-order conditional entropy-constrained tree-structured vector quantizer T^J, with a leaf node set $\tilde{L}^J = \{1, 2, \ldots, L^J\}$ and associated reproduction codevectors $Y_1^J, Y_2^J, \ldots, Y_{L^J}^J$, is a mapping of an input vector $X \in R^K$ to a reproduction codevector $Y_l^J, l = 1, 2, \ldots, L^J$ determined by the sequential refining quantization operation described earlier using the *nearest neighbor rule* defined in (6) with the rate defined as the conditional self-entropy $r_{i_k}^j = -\log_2 P_{i_k/l}^j$, where r_k and l are the nodes at the same level j and l is the node through which the preceding vector X_{k-1} was encoded. The rate $r_{i_k}^j$ is the code length for encoding X_k from the root to the node i_k at the jth level given that $S^j(X_{k-1}) = l$. We shall assume throughout the rest of the paper that the distortion measure is mean-squared error, i.e.,

$$d(X, Y) = \rho(\|X - Y\|) = \|X - Y\|^2. \tag{17}$$

III.1 An Ideal Design Algorithm

Given a training vector set and a target rate, we grow the tree by splitting a leaf node at a time. The initial tree has only one node, i.e., the root node t. One way to initialize the tree is to choose the centroid of the training source as the root node. In the Jth growing step, to split a leaf node l into two child nodes, the codevector of l is first perturbed by a small number into two points as the initial codevectors of the two child nodes. Then an iterative rate-constrained splitting technique is used to obtain the maximum attainable $\lambda_{l,max}^J$ resulting

from the splitting of l. The iterative splitting process starts with $\lambda^J_{l,max} = 0$ and minimizes (12) using the CECVQ algorithm in splitting. After the CECVQ algorithm converges, the decrease in the distortion and the increase in the rate due to the splitting with $\lambda^J_{l,max} = 0$ are computed using (15) and (16), respectively. $\lambda^J_{l,max}$ is then updated to their ratio as in (14). The iterative splitting process uses CECVQ algorithm to resplit the leaf node with the new $\lambda^J_{l,max}$ and the two codevectors obtained by CECVQ algorithm in the last iteration as the initial codevectors to minimize (12). The iteration process of $\lambda^J_{l,max}$ is repeated until it stops increasing. The value of $\lambda^J_{l,max}$ is the maximum achievable $\lambda^J_{l,max}$ from splitting the leaf node l currently being examined. After each leaf node has been examined, the design algorithm chooses to split the leaf node i which achieves $\lambda^J_{i,max} \geq \lambda^J_{l,max}, \forall l \in \tilde{L}^J$. Thus each growing step guarantees to achieve the best trade-off between the decrease in distortion and the increase in rate. We summarize the conditional entropy-constrained tree-structured vector quantizer design algorithm as follows.

Algorithm

- **Step 1** (initialization)

  Given a target rate R_{target} and a training source X, obtain T^0 which has only the root node with the centroid of X as its associated codevector. Set J=0; $R^0 = 0$; $D^0 = \sigma^2$, where σ^2 is the variance of X.

- **Step 2** (Find the maximum achievable λ for each leaf node)

  For each leaf node l of T^J, $l \in \tilde{L}^J$, use the iterative rate-constrained splitting process to find the maximum attainable $\lambda^J_{l,max}$ from splitting l.

  - **Step 2.1**

    Set initial $\lambda^J_{l,max} = 0$; perturb the codevector of j into two codevectors as the initial codevectors of j's child nodes.

  - **Step 2.2**

    Use CECVQ algorithm to minimize (12) with the nearest neighbor rule (13).

- **Step 2.3**

  Calculate ΔD^J and ΔR^J resulting from the current splitting using (15) and (16). If $\frac{\Delta D^J}{\Delta R^J} > \lambda^J_{l,max}$, set $\lambda^J_{l,max} = \frac{\Delta D^J}{\Delta R^J}$ and go to **Step 2.2**. Otherwise the maximum achievable $\lambda^J_{l,max}$ for splitting l is found.

- **Step 3** (Growing T^J to obtain T^{J+1})

  Find the leaf node i which can achieve the best rate-distortion trade-off by the following

$$i = \arg\max_{j \in \check{L}^J} \lambda_j.$$

  Split i to obtain the new tree T^{J+1}.

- **Step 4** (Updating)

  Calculate ΔR^J and ΔD^J using (15) and (16), respectively.

$$\begin{aligned} R^{J+1} &= R^J + \Delta R^J; \\ D^{J+1} &= D^J - \Delta D^J. \end{aligned}$$

  Re-estimate the new conditional probabilities and update the conditional probability table.

- **Step 5** (Stopping criterion)

  If $R^{J+1} < R_{target}$, set $J = J + 1$ and go to Step 2. Otherwise, store the conditional probability table, codevectors for all nodes, and λ's of all interior nodes, then stop.

III.2 A Fast Design Algorithm

The major computational cost of the above ideal design algorithm is in Step 2, especially when the size of the tree is getting large. After each growing step, the iterative splitting process has to be applied to each leaf node to re-evaluate the maximum achievable λ of splitting that leaf. This will slow down the design process of the codebooks of CECTSVQ. For example, to design a tree having L leaf nodes, it requires $\frac{1}{2}L(L-1)$ iterative splittings for calculating the

maximum attainable λ. This may cause difficulty for designing large codebooks of CECTSVQ with limited computing capability. However, the design process of large codebooks can be made much faster by making a minor approximation in the above ideal algorithm.

Let $\lambda^J_{l,max}$ denote the maximum achievable λ from splitting l. After splitting a leaf node i into two children i_1 and i_2, the maximum achievable $\lambda^J_{l,max}$ from splitting another leaf node l into l_1 and l_2 may or may not be changed. Clearly, if there is no training vector mapped on l whose predecessor is mapped on i, i.e, the conditional probability $P_{l|i} = 0$, the splitting of i into i_1 and i_2 will have no effect to the value of $\lambda^J_{l,max}$, since the iterative splitting process assigns the training vectors mapped on l to l_1 or l_2 according to the nearest neighbor rule (13) and there will be also no training vector mapped on l with its predecessor mapped on i_1 or i_2. If $P_{l|i} > 0$, then the splitting of i may change the value of $\lambda^J_{l,max}$. Before the splitting of i, $\lambda^J_{l,max}$ is obtained by iteratively assigning the training vectors mapped on l to l_1 or l_2 using the nearest neighbor rule (13). Particularly, those training vectors X_k with $S(X_k) = l$ whose predecessors X_{k-1} have the state $S(X_{k-1}) = i$ are assigned to l_1 if

$$d(X_k, Y_{l_1}) - \lambda_l \log_2 P_{l_1|i} \leq d(X_k, Y_{l_2}) - \lambda_l \log_2 P_{l_2|i}; \tag{18}$$

or to l_2 by the similar criterion. Note that in splitting l, l_1, l_2, and i are at the leaf node level. If i had been split into i_1 and i_2, the predecessors' state would have become either $S(X_{k-1}) = i_1$ or $S(X_{k-1}) = i_2$ and the splitting of l would have been according to

$$d(X_k, Y_{l_1}) - \lambda_l \log_2 P_{l_1|i_m} \leq d(X_k, Y_{l_2}) - \lambda_l \log_2 P_{l_2|i_m}; \tag{19}$$

where $m = 1, 2$. Note that in this splitting, l_1, l_2, i_1, and i_2 are at the leaf node level. However, the conditional probabilities have the following relations:

$$P_{l_1|i} = \frac{P_{i_1}}{P_i} P_{l_1|i_1} + \frac{P_{i_2}}{P_i} P_{l_1|i_2} = \frac{P_{i_1 l_1}}{P_i} + \frac{P_{i_2 l_1}}{P_i}; \tag{20}$$

$$P_{l_2|i} = \frac{P_{i_1}}{P_i} P_{l_2|i_1} + \frac{P_{i_2}}{P_i} P_{l_2|i_2} = \frac{P_{i_1 l_2}}{P_i} + \frac{P_{i_2 l_2}}{P_i}. \tag{21}$$

When the number of leaf nodes is large, the codevector co-occurrence matrix is sparse and most of the co-occurrence frequencies are small. Therefore we can make the following approx-

imations

$$\log_2 P_{l_1|i} \approx \log_2 P_{l_1|i_1} + \log_2 \frac{P_{i_1}}{P_i}; \tag{22}$$

$$\log_2 P_{l_2|i} \approx \log_2 P_{l_2|i_1} + \log_2 \frac{P_{i_1}}{P_i}; \tag{23}$$

or

$$\log_2 P_{l_1|i} \approx \log_2 P_{l_1|i_2} + \log_2 \frac{P_{i_2}}{P_i}; \tag{24}$$

$$\log_2 P_{l_2|i} \approx \log_2 P_{l_2|i_2} + \log_2 \frac{P_{i_2}}{P_i}. \tag{25}$$

Since adding a limited term to both sides in (19) will not affect the assignment, the approximations mean if $P_{i_m l_1}$ and $P_{i_m l_2}, m = 1, 2$, are small, $\lambda^J_{l,max}$ will only have very little change after the splitting of i. From the above analysis we can see that the maximum achievable λ's of splitting other leaf nodes will, in most cases, not or approximately not change after i has been split. Therefore we can assume that after each growing step the maximum achievable λ's of splitting the other leaf nodes will approximately remain unchanged and modify the Step 2 of the ideal algorithm accordingly. Instead of evaluating the maximum achievable λ for each leaf node in Step 2, we only need to find the maximum achievable λ's for the two new leaf nodes after each growing step. This modification of the ideal algorithm can dramatically reduce the time of designing large CECTSVQ codebooks. The required number of iterative splittings for finding the maximum achievable λ's in designing a codebook having L leaf nodes is now reduced to $2(L-1)$. Table 1 gives some comparisons of the ideal algorithm and the fast algorithm. We also can anticipate that the performance of the fast algorithm will be close to the ideal algorithm. This anticipation is indeed true as confirmed by the experimental results.

III.3 Encoding and Decoding

To encode an input source vector X_k using the designed conditional entropy-constrained tree-structured vector quantizer T^J, we need the path map of X_{k-1} which is a sequence of indexes

leaf numbers	number of iterative splits	
	ideal algorithm	fast algorithm
128	8128	254
256	32640	510
512	130816	1022
1024	523776	2046

Table 1: Comparison of number of iterative splits for ideal and fast algorithms.

of nodes to which X_{k-1} is mapped. As we have described in the previous section, the encoding of X_k using T^J is a sequential refining quantization operation. We start at the root node and look up the node in the first level from the path map of X_{k-1}. Then we find the two conditional probabilities in the conditional probability table and use the λ stored at the root node to decide the child node to which X_k should be mapped according to (13). This quantization operation is successively repeated at the next level until X_k is mapped on a leaf node. The path map is entropy coded by an entropy encoder matched to the conditional probability distribution of that leaf node and transmitted to the decoder. Since the quantization operation has memory, the encoding depends on the initial state. The first input source vector can be coded by the biased nearest neighbor rule with a memoryless entropy code.

The decoding process is the same as that of the conventional TSVQ. The received codeword is first decoded back to the path map by the entropy decoder. The path map is then used to find the leaf node that X_k is mapped on and the reproduction codevector at that leaf node reconstructs X_k.

Obviously, this design algorithm works only if the iterative splitting process converges. The next section is devoted to that question.

IV On The Convergence Of The Algorithm

In this section we study the convergence problem of CECTSVQ algorithm. We shall analyze the sufficient conditions for the CECTSVQ design algorithm to converge. Since a splitting which does not result in an increase in rate has no meaning to our purpose, we consider only the splittings which result in an increase in rate. To proceed further, we first introduce the following lemma.

Lemma: Let S be a bounded and closed set on the real line and $D(R)$ a real-valued function with $R \in S$. If R_1 is a solution to

$$\min_{R \in S}\{D(R) + \lambda_1 R\} \tag{26}$$

and R_2 is a solution to

$$\min_{R \in S}\{D(R) + \lambda_2 R\}, \tag{27}$$

then, for any function $D(R)$, we have

$$(\lambda_2 - \lambda_1)(R_1 - R_2) \geq 0. \tag{28}$$

Proof: Since R_1 and R_2 are the solution to Eqn.'s (26) and (27), respectively, it follows that

$$\begin{aligned} D(R_1) &+ \lambda_1 R_1 \leq D(R_2) + \lambda_1 R_2, \\ D(R_2) &+ \lambda_2 R_2 \leq D(R_1) + \lambda_2 R_1. \end{aligned}$$

Rewrite the above inequalities as

$$\begin{aligned} D(R_1) &- D(R_2) \leq \lambda_1(R_2 - R_1), \\ D(R_2) &- D(R_1) \leq \lambda_2(R_1 - R_2). \end{aligned}$$

Thus the lemma is proved if we add both sides of the above two inequalities to get

$$0 \leq (R_1 - R_2)(\lambda_2 - \lambda_1).$$

A corollary of this lemma immediately follows.

Corollary: The solution R is monotonically nonincreasing with λ.

Proof: Suppose that $\lambda_2 > \lambda_1 > 0$ in (28), we get

$$R_1 - R_2 \geq 0$$

which proves the corollary.

Suppose that we have grown a tree-structured quantizer T^J which has the average distortion D^J and average rate R^J, corresponding to a point (D^J, R^J) on the curve of the operational distortion-rate function $\hat{D}_n(R)$ in the distortion-rate plane as shown in Figure 1. We grow the tree to obtain a new tree of higher rate by splitting a leaf node which achieves the best trade-off between the increase in rate and the decrease in distortion. To find that leaf node, the CECTSVQ design algorithm tests each leaf node $l \in \tilde{L}^J$ by first splitting it with $\lambda = 0$ and minimizing the Lagrangian in (12). Then the rate of the tree will increase and the distortion will decrease, respectively. Denote the resulting tree as $T^{'}$ which has the average distortion $D^{'}$ and the average rate $R^{'}$. corresponding to the point $(D^{'}, R^{'})$ in Figure 1. Then we update λ using

$$\lambda = \frac{D^J - D^{'}}{R^{'} - R^J} = \tan\alpha > 0 \tag{29}$$

and split the leaf node and minimize (12) again with the new λ. Now the resulting tree $T^{''}$ must correspond to a point on the distortion-rate plane below or on the line joining (D^J, R^J) and $(D^{'}, R^{'})$. To see this, assume that $T^{''}$ has the average distortion $D^{''}$ and $R^{''}$. Since $D^{''}$ and $R^{''}$ minimize (12) after λ is updated to $\tan\alpha$, we must have

$$D^{''} + \tan\alpha R^{''} \leq D^{'} + \tan\alpha R^{'}; \tag{30}$$

and since $\tan\alpha > 0$, by the lemma and the corollary, $R^{'} \geq R^{''}$. When $R^{'} > R^{''}$, we have

$$\tan\alpha \geq \frac{D^{''} - D^{'}}{R^{'} - R^{''}} = \tan\beta. \tag{31}$$

When $R^{'} = R^{''}$, then $D^{''} \leq D^{'}$ and $\tan\beta \leq 0$. Therefore the point $(D^{''}, R^{''})$ must lie below or on the line joining (D^J, R^J) and $(D^{'}, R^{'})$; this means that

$$\tan\gamma = \frac{D^J - D^{''}}{R^{''} - R^J} \geq \tan\alpha. \tag{32}$$

So if we update λ with the new ratio in each iteration, it will be monotonically nondecreasing in the iterative splitting process for every leaf node. For finite training set case, since the number of the training vectors mapped onto a leaf node l is finite, there exist only finite ways to split l. Because we consider only those meaningful splittings which result in an increase in the rate (see Appendix), the smallest increase in rate results from a single vector split and we can show that λ is also upper bounded in the iterative splitting process for every leaf node. Let B_l be the finite set of all splittings which increase the rate and $\Delta R^J_{l,min}$ be the smallest increase in rate from the splitting in B_l, i.e.,

$$\Delta R^J_{l,min} = \min_{B_l} \Delta R^J_l. \tag{33}$$

Clearly, the largest decrease in distortion resulting from splitting l is upper bounded by D^J_l,

$$\Delta D^J_{l,max} = \max_{B_l} \Delta D^J_l \leq D^J_l. \tag{34}$$

Consequently, it follows that $\lambda^J_{l,max}$ is also upper bounded in the iterative splitting process for the leaf node l,

$$\lambda^J_{l,max} \leq \frac{\Delta D^J_{l,max}}{\Delta R^J_{l,min}} \leq \frac{D^J_l}{\Delta R^J_{l,min}}. \tag{35}$$

For the infinite training set case, we show in the Appendix that the maximum λ in the iterative splitting process must be achieved for a ΔR greater than zero. Since ΔD is upper bounded by D^J_l, $\lambda^J_{l,max}$ is also upper bounded. Since for each leaf node l, $\lambda^J_{l,max}$ is monotonically nondecreasing and bounded, the iterative splitting process at l will converge when $\lambda^J_{l,max}$ stops increasing and hence the algorithm also converges. For the purpose to grow a tree with increasing rate, we exclude the splittings which do not increase rate. In the Appendix we provide a proof that splitting off a single vector, in the case of either a finite or infinite training set, results in a finite λ, so can not always occur.

V Experimental Results

We have studied the rate-distortion performances of CECTSVQ on synthetic and real sources. The distortion measure used in the tests is the mean-squared error and the vector dimension

is 4. The first synthetic source tested is a first-order Gauss-Markov source AR(1) generated by

$$X(n) = aX(n-1) + W(n), \tag{36}$$

where a is the correlation coefficient, and $W(n)$ is an i.i.d. Gaussian random sequence with zero mean and variance σ^2. For the source to be stationary, a should satisfy $|a| < 1$. The correlation coefficient a is typically chosen as 0.9 to model highly correlated sources such as image and speech data. The autocorrelation sequence of this source is [19]

$$R_{XX}(n) = \frac{\sigma^2}{1-a^2} a^{|n|}. \tag{37}$$

We used the samples of a zero mean and unit variance AR(1) source with $a = 0.9$ to generate 200,000 vectors as the training source and another 60,000 vectors as the test source. CECTSVQ codebooks were designed using the training source and tested using the test source. For comparison, CECVQ, ECVQ, and ECTSVQ codebooks were also designed and tested using the same training and test sources. The codebook size for CECVQ and ECVQ was 256 codevectors. The marginal and conditional probability distributions were estimated from the codevector occurrence and co-occurrence frequencies. The average squared error distortion is presented in terms of the signal-to-quantization-ratio(SQNR) measured in dB, defined as $SQNR = 10\log_{10}(\frac{\sigma_X^2}{D})$. As a theoretical bound on the achievable rate-distortion performance, the rate-distortion function of AR(1) Gaussian source can be computed by[18]

$$R_\theta = \frac{1}{2\pi}\int_{-\pi}^{\pi} \max[0, \frac{1}{2} log_2 \frac{\Phi_{XX}(\omega)}{\theta}] d\omega, \tag{38}$$

$$D_\theta = \frac{1}{2\pi}\int_{-\pi}^{\pi} \min[\theta, \Phi_{XX}(\omega)] d\omega. \tag{39}$$

Where $\Phi_{XX}(\omega)$ is the discrete-time source power spectral density (psd)

$$\Phi_{XX}(\omega) = \sum_{n=-\infty}^{\infty} R_{XX}(n) e^{-jn\omega} = \frac{1-a^2}{1-2a\cos\omega + a^2}. \tag{40}$$

The nonzero portion of the $R(D)$ (or $D(R)$) curve is generated as the parameter θ traverses the interval $0 \le \theta \le$ *ess sup* $\Phi_{XX}(\omega)$ with *ess sup* denotes the essential supremum of a function. The results of the testing for all four systems at various rates are plotted in Figure 2. Also

shown in Figure 2 is the SQNR curve derived from (38) and (39). It can be concluded that on this AR(1) source and in the tested rate range , CECTSVQ performed consistently very close to CECVQ and significantly outperformed ECVQ and ECTSVQ. For example, at 0.6 bits/sample, CECTSVQ achieved 1dB gain over ECVQ and about 1.3dB over ECTSVQ.

The second source we tested is another widely used standard synthetic source, an AR(2) Gaussian source, given by

$$X(n) = a_1 X(n-1) + a_2 X(n-2) + W(n), \tag{41}$$

where a's are the regression coefficients and $W(n)$ again is an i.i.d Gaussian random sequence with zero mean and variance σ^2, and $a_1 = 1.515$ and $a_2 = -0.752$. This source is commonly used to model long-term statistical behavior of speech sources. The autocorrelation sequence of AR(2) is given by[19, pp. 123-132]

$$R_{xx}(n) = \sigma_X^2 \frac{(1-\mu_2^2)\mu_1^{|n|+1} - (1-\mu_1^2)\mu_2^{|n|+1}}{(\mu_1 - \mu 2)(1+\mu_1\mu_2)}, \tag{42}$$

where μ_1 and μ_2 are the roots of the quadratic $f(z) = z^2 - a_1 z - a_2$ and the source variance

$$\sigma_X^2 = \frac{(1-a_2)\sigma^2}{(1+a_2)(1+a_1-a_2)(1-a_1-a_2)}. \tag{43}$$

For stationarity it is required that $|\mu_1| < 1$ and $|\mu_2| < 1$. Since it is a Gaussian source, the rate-distortion function is also expressed by Eqns (38) and (39)[18] with the discrete-time source power spectral density (psd) given by

$$\begin{aligned} \Phi_{XX}(\omega) &= \sum_{k=-\infty}^{\infty} R_{XX}(k) e^{-jk\omega} \\ &= \frac{\sigma_X^2(1-\mu_1^2)(1-\mu_2^2)}{(\mu_1-\mu_2)(1+\mu_1\mu_2)} \{ \frac{\mu_1}{1-2\mu_1\cos\omega+\mu_1^2} - \frac{\mu_2}{1-2\mu_2\cos\omega+\mu_2^2} \}. \end{aligned} \tag{44}$$

Again output samples were drawn from an AR(2) Gaussian source with zero mean and unit variance to obtain 200,000 training vectors and 60,000 test vectors as the training source and test source. Similarly various codebooks of CECTSVQ, CECVQ, ECVQ, and ECTSVQ were designed using the training source and their performances were evaluated using the test source. The results are plotted in Figure 3. The rate-distortion curve is also computed as a theoretical

reference of the achievable performance. On this source, CECTSVQ also performed closely, within 0.8dB, to CECVQ, and significantly, more than 1dB, better than ECVQ and ECTSVQ.

The last source we used to test CECTSVQ is a speech source recorded from several male and female speakers. The speech was lowpass filtered and digitized into 16-bit linear PCM at 8 KHz. We divided the speech source into a training source of 200,000 vectors and a test source of 60,000 vectors. Both the training source and test source were then normalized to zero mean and unit variance. Once again CECTSVQ, CECVQ, ECVQ, and ECTSVQ codebooks were designed from the training source and their performances were evaluated using the test source. The trees designed by ECTSVQ were pruned using the generalized BFOS pruning algorithm to obtain the optimal codebooks. Because speech sources are usually nonstationary and the codevector co-occurrence matrix is very sparse when the codebook size is large, the estimation of the conditional probability distribution from the co-occurrence matrix is usually not good. A "leave-one-out" technique was used in [2] to estimate the conditional probability distribution when the co-occurrence matrix is sparse. In our test, we used a simplified method of [2] to estimate the conditional probability distribution by the mixture

$$\hat{P}_{l|m} = \beta P_{l|m} + (1-\beta)P_l; \tag{45}$$

or

$$\hat{P}_{ml} = \beta P_{ml} + (1-\beta)P_m P_l; \tag{46}$$

where P_l, $P_{l|m}$, and P_{ml} are the estimates of marginal, conditional, and joint probabilities from the codevector occurrence and co-occurrence, respectively; and β is chosen as 0.8. The performances are shown in Figure 4. The codebooks of CECTSVQ were generated using the approximate but fast design algorithm. Once again, the results have shown that CECTSVQ performed very close to CECVQ and could gain more than 2 dB over ECVQ and ECTSVQ.

In order to see if CECTSVQ can perform better than ordinary ECTSVQ followed by a matching conditional entropy coding, we compare the performances of the two systems in Figure 5, Figure 6, and Figure 7. The results, as expected, show that the gain of CECTSVQ over ordinary ECTSVQ followed by matching conditional entropy coding is relatively small

for the synthetic sources, but significant for the speech source, where it reaches as much as 3.0 dB at some rates.

Finally to compare the performances of the two CECTSVQ design algorithms, we also compared the codebooks designed by the approximate algorithm and the ideal algorithm. The results are plotted in Figure 8. The results show that the two algorithm performed virtually the same. However the approximate design algorithm is much faster than the ideal design algorithm.

VI Conclusions

We have introduced an algorithm to design the tree-structured vector quantizers subject to the first-order conditional entropy constraint. The algorithm grows the tree by splitting a leaf node at a time. The leaf node selected to be split has the largest attainable Lagrange multiplier determined by an iterative rate-constrained splitting process. Therefore each split results in the best trade-off between the decrease in the distortion and increase in the rate. We have also derived a fast algorithm based on the ideal algorithm to achieve tremendous savings in computation in the design process with negligible performance loss. We have shown that the performance of the quantizers designed by the CECTSVQ algorithm on synthetic and speech sources is significantly better than that of ECVQ and ECTSVQ, and is very close to that of CECVQ. We have also shown that CECTSVQ can offer significant gain over ordinary ECTSVQ followed by matching conditional entropy coding in real data compression.

The performance of CECTSVQ is achieved at the cost of the additional storage requirement. To utilize a codebook designed by CECTSVQ, one needs also to store the logarithm table of the conditional probabilities and λ's for all nodes. The amount of storage needed for the conditional entropy codes of a tree with N nodes is

$$2^2 + 3^2 + \ldots + N^2 = \frac{1}{6}N(N+1)(2N+1) - 1. \tag{47}$$

When the size of the tree is large, this storage requirement may become a practical limita-

tion. This limitation will prevent using CECTSVQ in applications of high rates and large vector sizes. Also CECTSVQ can not be applied for progressive transmission applications since the conditional entropy codes for successive levels are not embedded in general. In our experiments, the trees grown have leaf nodes between 512 to 1024. The greatest advantage of CECTSVQ is that it has much lower encoding computational complexity than the full search CECVQ, thus the quantization operation is much faster. With the cost of memory rapidly dropping, CECTSVQ will become more and more attractive. However the trees designed by the CECTSVQ algorithm are not guaranteed to be optimal, since it is a one step look-ahead greedy algorithm. The generalized BFOS algorithm could be applied to prune a tree grown by CECTSVQ to obtain the best subtree for a given rate or distortion.

VII Acknowledgement

The authors wish to thank an anonymous reviewer whose very insightful comments and constructive suggestions were invaluable toward significantly improving the paper.

Appendix

Upper Bound on λ

We show that the CECTSVQ design algorithm will not always split off a single vector and that λ is upper bounded and is maximized for $\Delta R > 0$ for both finite and infinite training set cases. To do so we show that λ, the ratio of decrease in distortion to increase in rate, does not go to infinity in single vector split cases. For the sake of simplicity but without losing generality, let us consider the example of splitting the root node A into two nodes B and C.

Let us assume first that there is a finite training sequence $\{X_0, X_1, \ldots, X_N\}$ with $1+N$ vectors, and that the iterative splitting process ends up with a single vector split, say the kth vector X_k is mapped to C and the rest vectors to B. Then we have the following cases.

1) If X_k is in the middle of the sequence, i.e., $k > 0$ and $k < N$

Because the first vector is only used to provide an initial condition, it is excluded in the probability estimation.

$$\begin{aligned}
P_{B/B} &= \frac{N-2}{N-1}, \quad P_{C/B} = \frac{1}{N-1}, \quad P_{B/C} = 1, \ \textit{and} \ P_{C/C} = 0; \\
P_B &= \frac{N-1}{N}, \quad P_C = \frac{1}{N}; \\
P_{BB} &= P_B P_{B/B} = \frac{N-2}{N}, \quad P_{BC} = P_B P_{C/B} = \frac{1}{N}, \\
P_{CB} &= P_C P_{B/C} = \frac{1}{N}, \quad P_{CC} = 0.
\end{aligned}$$

The conditional entropy H^0 of the root node A, i.e., the initial tree, is 0. The increase in the average rate, i.e., in the conditional entropy resulting from the single vector split is

$$\begin{aligned}
\Delta R &= H^1 - H^0 = H^1 \\
&= -P_{BB}\log_2 P_{B/B} - P_{BC}\log_2 P_{C/B} - P_{CC}\log_2 P_{C/C} \\
&= \frac{N-2}{N}\{\log_2(N-1) - \log_2(N-2)\} + \frac{1}{N}\log_2(N-2) > 0. \qquad (48)
\end{aligned}$$

Where we have taken $P_{CC}\log_2 P_{C/C} = P_C P_{C/C}\log_2 P_{C/C} = 0$ in accordance with *L'Hospital*'s

Rule $\lim_{x\to 0} x \log x = 0$ and the treatment in the implementation.

2) X_k is at the end of the sequence, i.e., $k = N$

In this case,

$$\begin{aligned} P_{B/B} &= \frac{N-1}{N}, \quad P_{B/C} = 0, \quad P_{C/B} = \frac{1}{N}, \ \ and \ \ P_{C/C} = 0; \\ P_B &= \frac{N-1}{N}, \quad P_C = \frac{1}{N}; \\ P_{BB} &= \frac{(N-1)^2}{N^2}, \quad P_{BC} = \frac{N-1}{N^2}, \quad P_{CB} = P_{CC} = 0. \end{aligned}$$

Then,

$$\begin{aligned} \Delta R &= H^1 = -P_{BB} \log_2 P_{B/B} - P_{BC} \log_2 P_{C/B} \\ &= \frac{(N-1)^2}{N^2}\{\log_2 N - \log_2(N-1)\} + \frac{N-1}{N^2} \log_2 N > 0. \end{aligned} \tag{49}$$

3) If X_k is the first vector, $k = 0$, $\Delta R = 0$. But in this case, as mentioned earlier in the paper, we treat such a split as invalid because it does not increase the rate in the conditional case.

The decrease in average distortion for the applicable cases above is $\Delta D = |X_k - X_{centroid}|^2/N$, for mean square error distortion and $X_{centroid}$ the centroid of the training set. So for the finite training set of size $N+1$, the ratio $\lambda = \Delta D/\Delta R$ is finite in the case of a single vector split, since the increase in the average rate and the decrease in the distortion are both finite and non-zero. In fact, for large N, ΔD decreases at the rate of $O(1/N)$ while ΔR decreases at the rate of $O(\log_2 N/N)$, hence $\lambda = O(1/\log_2 N)$. Formally, in the limit of an infinite training set, where $X_{centroid}$ is not changed by a single vector split, we obtain through application of *L'Hospital*'s Rule,

$$\lim_{N\to\infty} \lambda = \lim_{N\to\infty} \frac{\Delta D}{\Delta R} = 0. \tag{50}$$

Therefore, the non-negative function $\Delta D(\Delta R)$ can not be concave at the origin and the maximum λ must be achieved for a ΔR greater than zero.

References

[1] R. E. Blahut, *Principles and Practice of Information Theory*, Addison-Wesley Publishing Company, 1987.

[2] P. A. Chou and T. Lookabaugh, "Conditional Entropy Constrained Vector Quantization," *Proc. ICASSP*, pp. 197-200, April, 1991.

[3] Diego P. de Garrido and W. A. Pearlman,"Conditional Entropy Constrained Vector Quantization: High Rate Theory and Design Algorithms," *IEEE Trans. on Information Theory*, Vol. 41, No. 4, pp. 901-916, July 1995.

[4] P. A. Chou, T. Lookabaugh, and R. M. Gray, "Entropy-Constrained Vector Quantization," *IEEE Trans. on ASSP,* Vol. 37, No.1, pp.31-42, Jan. 1989.

[5] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*, klumar Academic Press, MA, 1991.

[6] M. Balakrishnan, W. A. Pearlman, and L. Lu, "Variable Rate Tree Structured Vector Quantizers," *IEEE Trans. on Information Theory*, Vol. 41, No. 4, pp. 917-930, July 1995.

[7] P. A. Chou, T. Lookabaugh, and R. M. Gray, "Optimal pruning with application to tree-structured source coding and modeling," *IEEE Trans. on Information Theory,* Vol. 35, pp. 299-315, 1989.

[8] B. Mahesh and W. A. Pearlman, "Multi-Rate structured vector quantization of image pyramids", *Journal of Visual Commu. and Image Representation,* Vol. 2, pp. 103-113, June 1991.

[9] Y. Linde, A. Buzo, and R. M. Gray, "An algorithm for vector quantizer design", *IEEE Trans. on Communications*, Vol. COM-28, pp. 84-95, Jan. 1980.

[10] A. Buzo, A. H. Gray, Jr., R. M. Gray, and J. D. Markel, "Speech coding based upon vector quantization," *IEEE Trans. on Information Theory,* Vol. 28, No. 5, pp.562-574, Oct. 1982.

[11] J. Makhoul, S. Roucos, and H. Gish, "Vector quantization in speech coding, " *Proc. IEEE*, Vol. 73, pp. 1551-1588, 1985.

[12] R. Lindsay, unpublished work at Unisys, presented at the NASA Data Compression Workshop, Snowbird, UT, 1988.

[13] E. A. Riskin and R. M. Gray, "A greedy tree growing algorithm for the design of variable rate quantizers," *IEEE Trans. on Signal Processing,* Vol. 73, pp. 1551-1558, 1991.

[14] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, "Classification and Regression trees. The Wadsworth Statistics/Probability Series", Wadsworth, Belmont, CA, 1984.

[15] L. Lu and W. A. Pearlman, "Adaptive Joint Rate Allocation and Quantization in Subband Signal Coding," Proc. IEEE Int. Conf. on Image Processing, Vol. III, pp. 412-415, Washington D.C., Oct. 22-25, 1995.

[16] L. Lu and W. A. Pearlman, "Multi-rate video coding using pruned tree-structured vector quantization," *Proc. ICASSP-93*, Vol. V, pp. 253-256, April, 1993.

[17] L. Lu, *Advances in Tree-Structured Vector Quantizations and Adaptive Video Coding,* PhD Thesis, Rensselaer Polytechnic Institute, August 1995.

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

[19] M. R. Priestley, *Spectral Analysis and Time Series,* CA: Academic Press, San Diego, 1989.

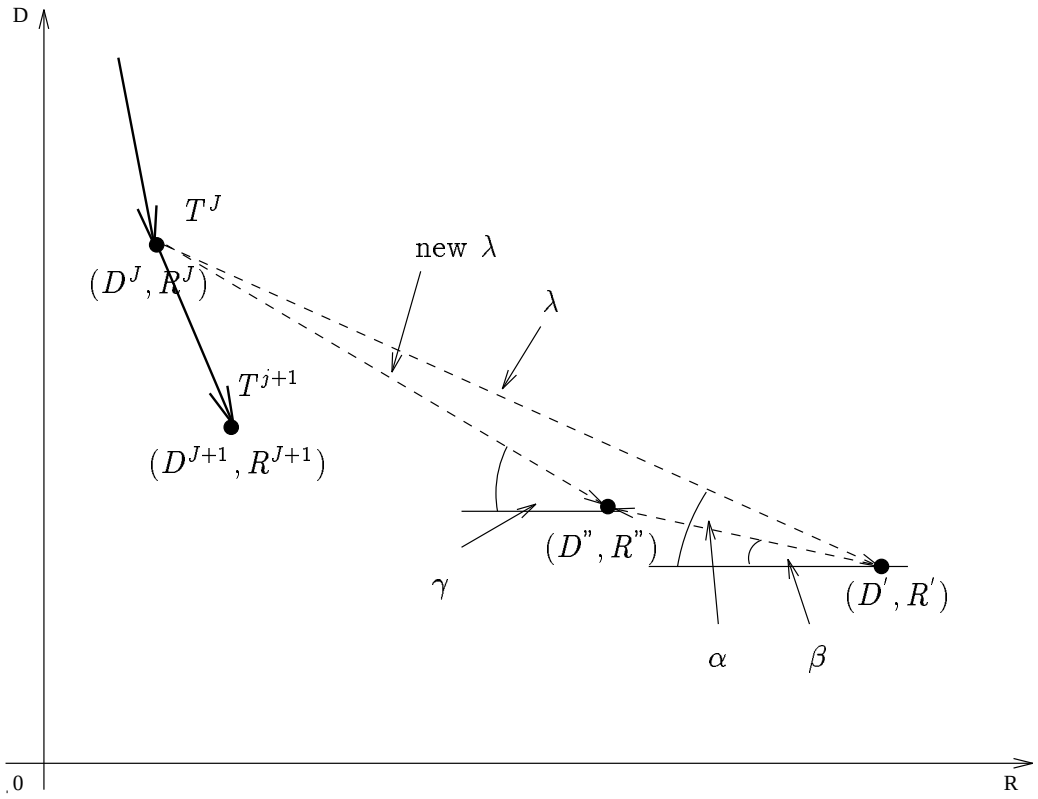

Figure 1: Figure for the proof of the convergence of CECTSVQ.

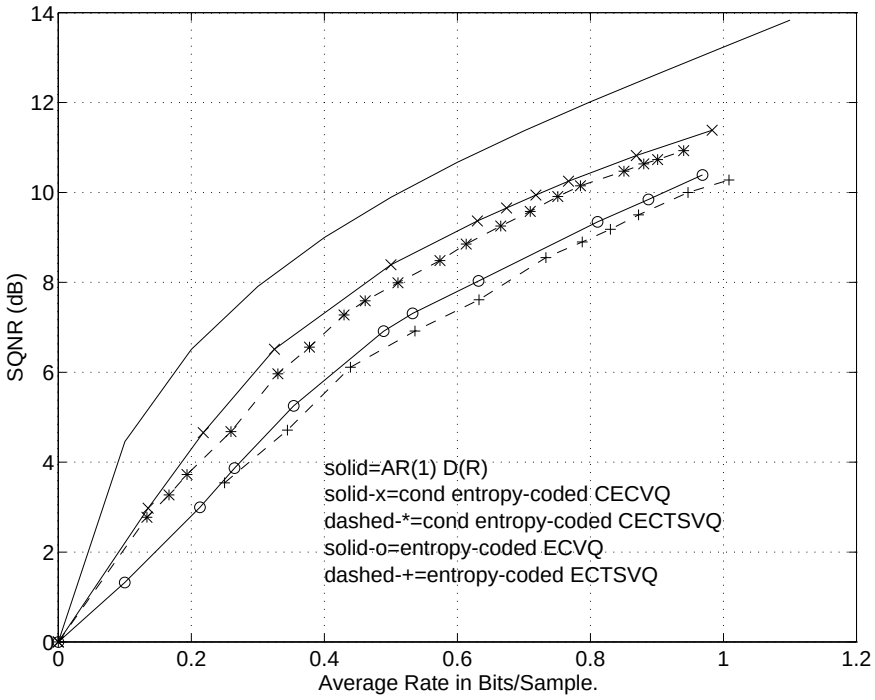

Figure 2: Performance of CECTSVQ on AR(1) source.

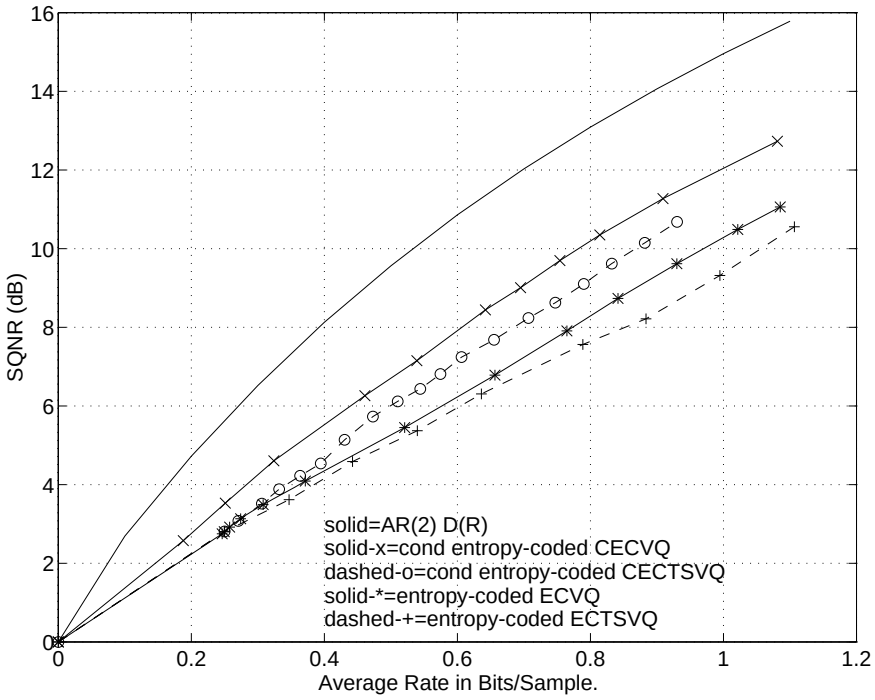

Figure 3: Performance of CECTSVQ on AR(2) source.

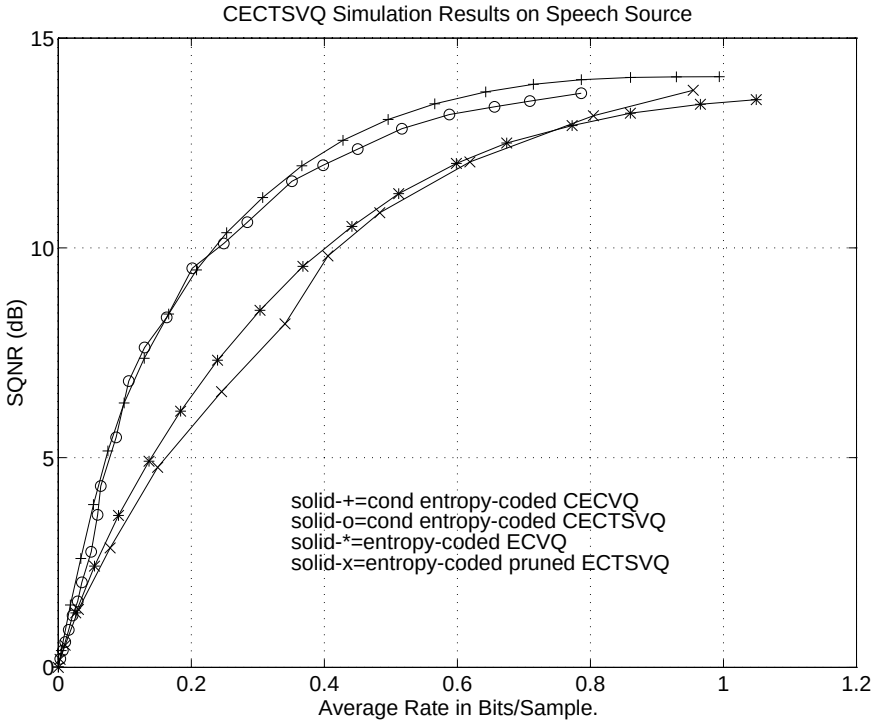

Figure 4: Performance of CECTSVQ on speech source.

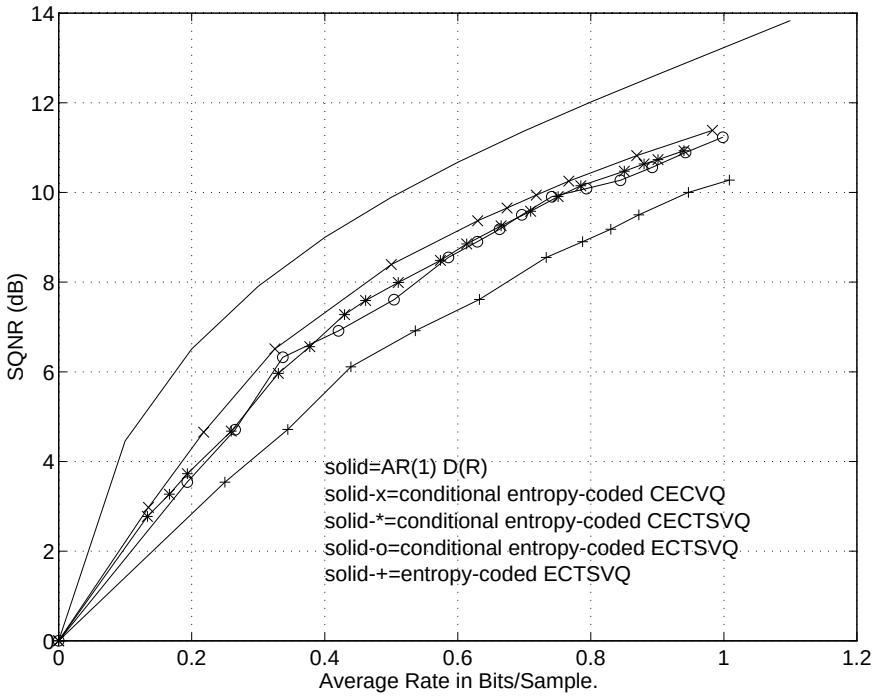

Figure 5: Comparison of CECTSVQ and ECTSVQ followed by conditional entropy coding on AR(1) source.

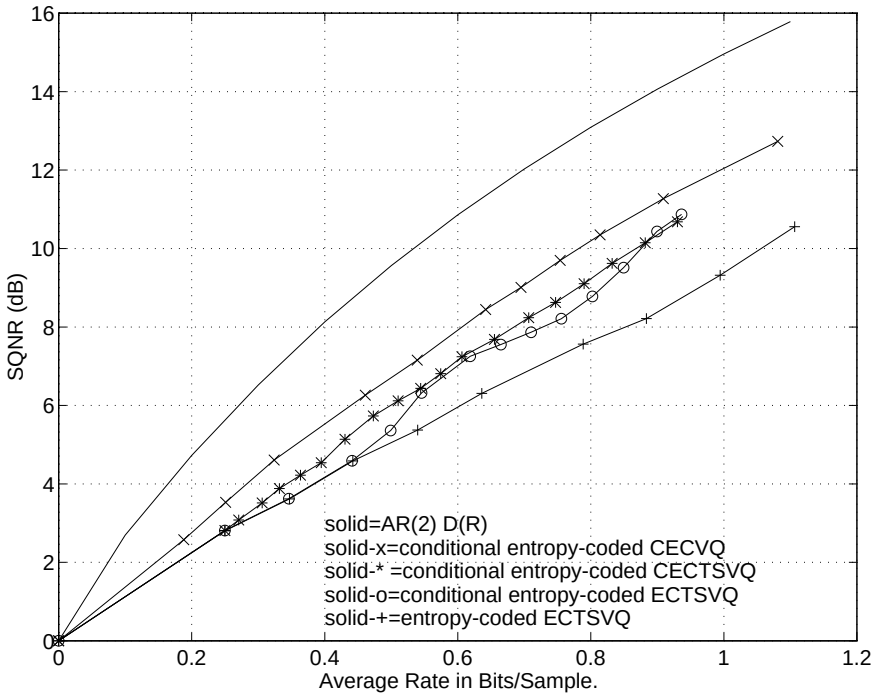

Figure 6: Comparison of CECTSVQ and ECTSVQ followed by conditional entropy coding on AR(2) source.

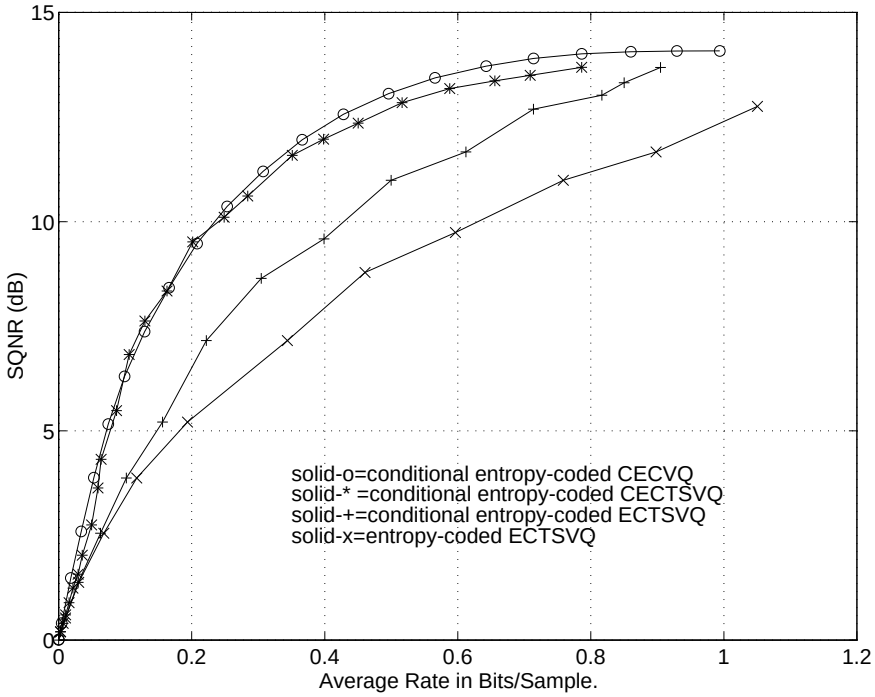

Figure 7: Comparison of CECTSVQ and ECTSVQ followed by conditional entropy coding on speech source.

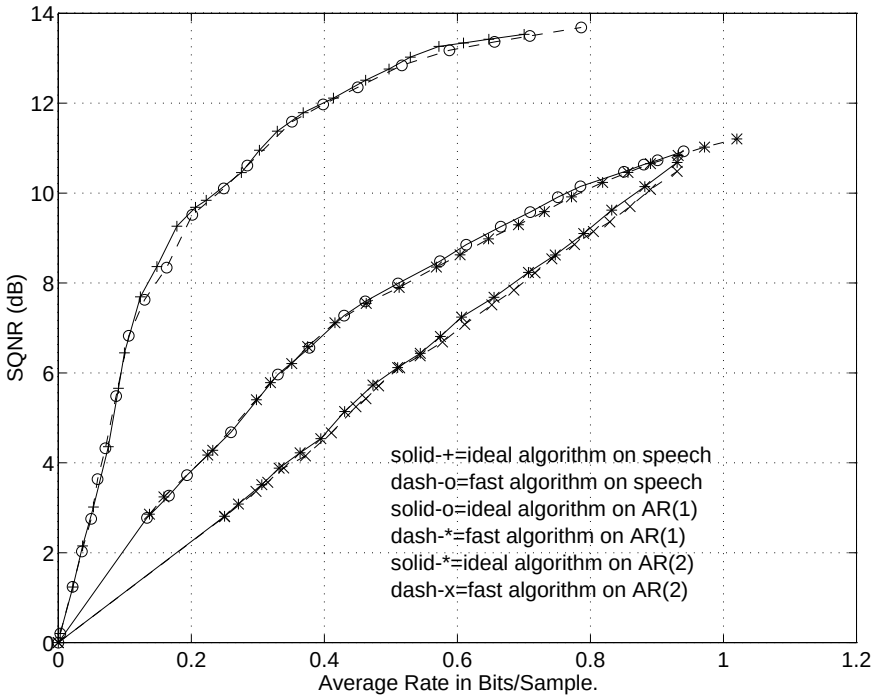

Figure 8: Performance of the CECTSVQ fast design algorithm.