IEEE Transactions on Circuits and Systems for Video Technology, Vol. 5, pp. 83–95, April 1995

A CLUSTERING ALGORITHM FOR ENTROPY-CONSTRAINED VECTOR QUANTIZER DESIGN WITH APPLICATIONS IN CODING IMAGE PYRAMIDS[1]

Diego P. de Garrido[2], *Student Member, IEEE*, William A. Pearlman[3], *Senior Member, IEEE*, and Weiler A. Finamore[4], *Member, IEEE*

Abstract–**A clustering algorithm for the design of efficient vector quantizers to be followed by entropy coding is proposed. The algorithm, called entropy-constrained pairwise nearest neighbor (ECPNN), designs codebooks by merging the pair of Voronoi regions which gives the least increase in distortion for a given decrease in entropy. The algorithm is an entropy-constrained version of the pairwise nearest neighbor (PNN) clustering algorithm of Equitz and can be used as an alternative to the entropy-constrained vector quantizer design (ECVQ) proposed by Chou, Lookabaugh and Gray. By a natural extension of the ECPNN algorithm we develop another algorithm that designs alphabet- and entropy-constrained vector quantizers and call it alphabet- and entropy-constrained pairwise nearest neighbor (AECPNN) design. The AECPNN algorithm can be used as alternative to the alphabet- and entropy-constrained vector quantizer design (AECVQ) proposed by Rao and Pearlman, that is directly based on the ECVQ design algorithm. Through simulations on synthetic sources, we show that ECPNN and ECVQ have indistinguishable mean-square-error versus rate performance and that the ECPNN and AECPNN algorithms obtain as close performance by the same measure as the ECVQ and AECVQ algorithms. The advantages over**

[1]Portion of this material were presented at the SPIE Conf. on Visual Commun. and Image Processing, Boston, MA, Nov. 1991 and at the IEEE Conf. on Acoustics, Speech and Signal Processing, 1992. The work of D. P. de Garrido was supported by the Conselho Nacional de Densenvolvimento Científico e Tecnológico (CNPq-Brazil) under Grant 202.178-90. This material is based upon work supported by the National Science Foundation under Grant Nos. INT-8802240 and NCR-9004758. The government has certain rights in this material.

[2]D. P. de Garrido is with IBM T. J. Watson Research Center, Yorktown Heights, NY 10598

[3]W. A. Pearlman is with the Electrical, Computer and Systems Engineering Department, Rensselaer Polytechnic Institute, Troy, NY 12180-3590.

[4]W. A. Finamore is with Department of Electrical Engineering, Pontifica Universidade Católica, Caixa Postal 38053, Agência PUC, 22451-Rio de Janeiro-RJ, Brazil.

ECVQ are that the ECPNN approach enables much faster codebook design and uses smaller codebooks. A single pass through the ECPNN (or AECPNN) design algorithm, which progresses from larger to successively smaller rates, allows the storage of any desired number of intermediate codebooks. In the context of multi-rate subband (or transform) coders, this feature is especially desirable, since the ECPNN design algorithm must be run for each subband and storage of codebooks of different rates are required for each subband (or transformed coefficients). Coding image pyramids using ECPNN and AECPNN codebooks at rates from $\frac{1}{3}$ to 1.0 bit/pixel, achieves performance, judged both visually and using peak-to-peak SNR criterion, which is competitive with that of the best methods currently known.

I. Introduction

Vector quantization is a scheme to represent digitally a sequence of discrete-time continuous-amplitude source symbols $\{X_j\}_{j=0}^{\infty}$. A vector quantizer maps points $\mathbf{x}_i = [X_{iL}, ..., X_{iL+L-1}]^T$ belonging to an $L-$dimensional real vector space, $\mathbf{x}_i \in \Re^L$, into a vector in the codebook $\mathcal{C} = \{\mathbf{y}_1, ..., \mathbf{y}_M\}$, $\mathcal{C} \subset \Re^L$. This quantizer is referred to as M-level vector quantizer. At each time index i the VQ encoder performs the mapping $\mathbf{y}_{m_i} = Q(\mathbf{x}_i)$ where m_i is an integer index in the codebook and $Q(.)$ is the function that characterizes the quantizer. The binary representation of m_i is sent to the decoder. The VQ decoder has a codebook identical to the VQ encoder, and decoding can be implemented as a look-up table operation. The error of the mapping of $\mathbf{x}_i$ to $\mathbf{y}_{m_i}$ is measured by a non-negative function $d(\mathbf{x}_i, \mathbf{y}_{m_i})$, which is a general distortion criterion.

The optimum minimum-distortion quantizer minimizes the average distortion of this mapping without additional restrictions. A well known algorithm to find a local minimum of the distortion of the $M-$level quantizer has been proposed by Linde, Buzo and Gray [1] and is called generalized Lloyd algorithm (GLA), since it is a vector generalization of a clustering algorithm due to Lloyd [2]. With the GLA, an initial codebook $\mathcal{C}$ and a given set $\mathcal{T} = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$ of $L-$dimensional vectors, called the training set (TS), a locally $M-$level minimum distortion quantizer can be approached iteratively.

The design of minimum distortion vector quantizers (MDVQ's) via the GLA is rather time-consuming. With the purpose of alleviating this problem, a different procedure has been proposed by Equitz [3]. Equitz's algorithm is a recursive procedure which is called pairwise nearest neighbor (PNN). Substituting the TS as the initial codebook, the PNN algorithm recursively merges the pair of reproduction vectors giving the least increase of distortion until the desired codebook size is reached.

The minimum achievable rate of a quantizer is its output entropy, also called its codeword or index entropy. Let the L-dimensional codevectors $\{\mathbf{y}_1, ..., \mathbf{y}_M\}$ have respective probabilities $\{P_1, ..., P_M\}$. Then the quantizer's output entropy in bits per sample is

$$H_Q = -\frac{1}{L}\sum_{m=1}^{M} P_m \log_2 P_m. \tag{1}$$

An optimal vector quantizer minimizes average distortion D under the condition that the entropy not exceed a given value H_0. Such quantizers are called entropy-constrained vector quantizers (ECVQ's). An MDVQ designed by the GLA or PNN algorithm attempts to minimize D, fixing the size M of the codebook and hence allows the entropy to reach its maximum of $(1/L)\log_2 M$. The entropy reaches this value only when the codevectors are equally probable, which is generally not the case. For a given output entropy H_0, the minimum D for such quantizer can be no less than that of a quantizer having no restriction on the codebook size, as with ECVQ. The design of an entropy-constrained vector quantizer can accomplished by a modification of the GLA. This design procedure is named ECVQ algorithm and was originally introduced by Chou, Lookabaugh and Gray [4]. It has been shown in [4] and [5] among other references that the entropy-constrained vector quantizers are in fact substantially better in the distortion-rate sense than the minimum distortion vector quantizers (*even when entropy coding of codevector indices is performed*).

Another class of quantizers is the so called alphabet- and entropy-constrained vector quantizers, which were introduced recently by Rao and Pearlman [6]. These quantizers have the capability to traverse the operational distortion-rate curve using the same set of codevectors (alphabet) but with a different set of (binary) codewords for each rate. The design procedure of these quantizers is a modification of the ECVQ design that forces operation with a fixed codebook for all rates. The main advantage of this double-constrained quantizer is to decrease the storage of the encoding-decoding process with a negligible increase of distortion when compared with an entropy-constrained quantizer.

In this paper we present an algorithm based on a procedure which constrains the entropy in the PNN quantizer design. We name the procedure ECPNN and the resulting quantizer combined with subsequent entropy coding the ECPNN-quantizer. An earlier version of this approach can be found in [7] with further developments in [8]. The alphabet constrained version of the ECPNN, called AECPNN, is developed by a modification of the design strategy. The AECPNN approach

[9], substantially simplifies operation to that of a multi-rate coder, since the same codebook is used for the desired operational rates.

In the next section we introduce some concepts that will be useful for the development of the ECPNN and AECPNN algorithms that are presented in the Section III. In the Section IV we give some implementation aspects and computational complexity. Results of quantizing synthetic sources are presented in Section V and are compared with those obtained with the ECVQ and AECVQ algorithms. A multi-rate subband coder for images, based on ECPNN and AECPNN codebooks, is presented in Section VI.

II. Entropy-constrained Vector Quantization

Let us consider L-dimensional random vectors generated by the information source, $\{X\}$, described by a joint probability density function $f_{\mathbf{X}}(\mathbf{x})$. Without loss of generality, we focus our interest on the squared error distortion measure between two vectors $\mathbf{x}$ and $\mathbf{y}$ given by:

$$d(\mathbf{x},\mathbf{y}) = \| \mathbf{x}-\mathbf{y} \|^2 = \sum_{k=1}^{L}(x_k - y_k)^2, \tag{2}$$

where the notation $\| \,.\, \|$ is used for the Euclidean norm of vector. All the developments presented here can be extended to more general distortion measures.

Given a source vector $\mathbf{x}$, the entropy-constrained vector quantizer [4] maps $\mathbf{x}$ to a codevector $\mathbf{y}_m = Q(\mathbf{x}) \in \mathcal{C}$ such that

$$d(\mathbf{x},\mathbf{y}_m) + \lambda \ell_m \leq d(\mathbf{x},\mathbf{y}_k) + \lambda \ell_k \;, \forall k = 1, ..., M, \tag{3}$$

where ℓ_m is the length of the binary codeword associated with $\mathbf{y}_m$ and is set to be equal to its self-entropy, i.e.,

$$\ell_m = -\log_2 P_m. \tag{4}$$

The average binary codeword length is then equal to its first-order entropy and can be achieved arbitrarily closely through entropy coding. The parameter $\lambda \geq 0$ is a Lagrange multiplier whose value fixes the output entropy or rate. Setting $\lambda = 0$ produces the minimum distortion quantization rule of an MDVQ.

The quantization rule induces a partitioning of L-dimensional Euclidean space $\Re^L$ called the encoder partition $\mathcal{E}^{\mathcal{P}} = \{\mathcal{P}_1, ..., \mathcal{P}_M\}$, where $\mathcal{P}_m = \{\mathbf{x} : Q(\mathbf{x}) = \mathbf{y}_m\}$, $m = 1, ..., M$ are called partition cells or Voronoi regions. For the squared error distortion measure, the codevectors are centroids of their associated partition, i.e.,

$$\mathbf{y}_m = \int_{\mathcal{P}_m} \mathbf{x} f_{\mathbf{X}}(\mathbf{x}) d\mathbf{x} / P_m, \;\; m = 1, ..., M, \tag{5}$$

where $P_m = \int_{\mathcal{P}_m} f_{\mathbf{X}}(\mathbf{x}) d\mathbf{x} = \Pr[Q(\mathbf{x}) = \mathbf{y}_m]$. The conditions for the set of partition cells $\mathcal{E}^{\mathcal{P}} = \{\mathcal{P}_1, ..., \mathcal{P}_M\}$ and the set of codevectors $\mathcal{C} = \{\mathbf{y}_1, ..., \mathbf{y}_M\}$ in (3) and (5) are necessary to achieve a minimum of the average distortion per sample

$$D = \frac{1}{L} \sum_{m=1}^{M} \int_{\mathcal{P}_m} d(\mathbf{x}, \mathbf{y}_m) f_{\mathbf{X}}(\mathbf{x}) d\mathbf{x}. \tag{6}$$

The entropy-constrained vector quantizer design of Chou, Lookabaugh and Gray [4] starts with a codebook designed by the GLA with a large M and iterates between (3) and (5) until convergence is reached for a particular value of λ. This procedure achieves a codebook $\mathcal{C}$ and an encoder partition $\mathcal{E}^{\mathcal{P}}$ which are optimal for a single rate associated with the value of λ. For another rate, a different value of λ is set and the algorithm produces a different codebook $\mathcal{C}$ and encoder partition $\mathcal{E}^{\mathcal{P}}$.

III. Constrained Pairwise Nearest Neighbor Quantizer Design

Let us consider now the process of design of a succession of quantizers when a training set replaces the information source. At the j-th stage let Q^j denote the quantizer and $\mathcal{E}^{\mathcal{P}^j} = \{\mathcal{P}_1^j, ..., \mathcal{P}_{A_j}^j\}$ denote the encoder partition with number of cells A_j. Let $\mathcal{C}^j = \{\mathbf{y}_1^j, ..., \mathbf{y}_{A_j}^j\}$ be the codebook where $\mathbf{y}_i^j$'s are the centroids derived from the training set $\mathcal{T} = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$ which are sample vectors of the information source $\{X\}$. The centroids are then defined by

$$\mathbf{y}_m^j = \frac{1}{n_m^j} \sum_{\mathbf{x}_k \in \mathcal{P}_m^j} \mathbf{x}_k; \;\; m \in \mathcal{A}^j = \{1, ..., A_j\}, \tag{7}$$

where n_m^j is the cardinality of the partition $\mathcal{P}_m^j$ (or the unscaled probability mass) and $\mathcal{A}^j$ indicates the set of integer indices.

In order to describe our results in a more convenient way, we will introduce next a notation for an operator named the "merge operator". Let us first introduce the notation $\mathcal{P}^{j+1}(a, b)$ to represent the partition cell which is obtained when we replace, in the set $\mathcal{E}^{\mathcal{P}^j}$, the two partition cells $\mathcal{P}_a^j$ and $\mathcal{P}_b^j$ $a, b \in \mathcal{A}^j$, $a \neq b$, by their union $\mathcal{P}_a^j \bigcup \mathcal{P}_b^j$. When the merge operation is applied to the A_j-level quantizer Q^j, indicated by $\mathcal{M}_{ab}[Q^j]$, it is understood that the new quantizer $Q^{j+1} = \mathcal{M}_{ab}[Q^j]$ with $A_j - 1$ levels, associated with the partition cell $\mathcal{P}^{j+1}(a, b)$, is obtained. The quantizer $Q^{j+1}(a, b)$ is said to be the merged (a, b) version of Q^j under operation $\mathcal{M}_{ab}$.

The PNN algorithm generates recursively a sequence of quantizers where the initial element is the quantizer Q^0 associated with $\mathcal{E}^{\mathcal{P}^0}$. For purposes of initialization, a larger GLA codebook or the full training sequence can be used. The latter was done by Equitz [3], whereby each partition started with just one element and $\mathbf{y}_m^0 = \mathbf{x}_m^0$. The quantizers in the sequence have a decreasing number of levels or codevectors and a non-decreasing distortion. The number of levels of the $j-$th quantizer in the sequence is $(A_0 - j)$ and the (empirical) distortion per sample that results when all the vectors belonging to the training set (or TS), are quantized with Q^j, is given by

$$D^j = \frac{1}{LN} \sum_{m=1}^{A_j} d_m^j, \tag{8}$$

where

$$d_m^j = \sum_{\mathbf{x}_k \in \mathcal{P}_m^j} d(\mathbf{x}_k, \mathbf{y}_m^j). \tag{9}$$

Consider the quantizer Q^j which achieves an average distortion D^j when used to quantize the training set. Consider now the merged (a,b) version of Q^j, i.e., $Q^{j+1} = \mathcal{M}_{ab}[Q^j]$. When this quantizer is used to quantize the vectors associated with the TS, the average distortion denoted by $D^{j+1}(a,b)$ is achieved. The distortion increment observed when the merge operator $\mathcal{M}_{ab}$ is applied to Q^j is

$$\Delta D_{ab}^{j+1} = D^{j+1}(a,b) - D^j. \tag{10}$$

Notice that the cells of the partition associated with Q^{j+1} are the cells associated with Q^j with $\mathcal{P}_a^j$ and $\mathcal{P}_b^j$ removed and replaced by their union $\mathcal{P}_{ab}^j$. As was verified by Wishart [10] and Equitz [3] for the mean squared error case, the centroid of the union partition cell, denoted by $\mathbf{y}^{j+1}(a,b)$, is given by

$$\mathbf{y}^{j+1}(a,b) = \frac{1}{n_a^j + n_b^j}(n_a^j \mathbf{y}_a^j + n_b^j \mathbf{y}_b^j). \tag{11}$$

Replacing the distortions in (10) by their respective expressions, considering (2), (8), and (9), the distortion increment in (10) becomes

$$\Delta D_{ab}^{j+1} = \frac{1}{LN}\{\sum_{\mathbf{x}_k \in \mathcal{P}_{ab}^j} \| \mathbf{x}_k - \mathbf{y}^{j+1}(a,b) \|^2 - d_a^j - d_b^j\}. \tag{12}$$

The average distortion per sample $D^{j+1}(a,b)$ obtained when the vectors associated with the TS are quantized with $Q^{j+1}(a,b)$ is then

$$D^{j+1}(a,b) = D^j + \Delta D_{ab}^{j+1}. \tag{13}$$

The PNN procedure is a search for the smallest distortion increment in the generation of the Q^{j+1} obtained by selecting the partition cells α and β defined by Q^j such that

$$\Delta D_{\alpha\beta}^{j+1} = \min\ [\Delta D_{ab}^{j+1}]\ \forall\ a, b \in \mathcal{A}^j,\ a \neq b. \tag{14}$$

One important point should be emphasized: although each merge produces a minimum distortion, the overall procedure need not produce the optimal (minimum distortion) codebook for the given number of levels [3], since it does not ensure satisfying the necessary conditions for optimality corresponding to (3) and (5) .

An important aspect noted in the works by Equitz [3] and Wishart [10] is that the summation over $\mathbf{x}_k$ in equation (12), to find ΔD_{ab}^{j+1}, need not be evaluated for each pair of indices in (14). Considering that the sets $\mathcal{P}_a^j$ and $\mathcal{P}_b^j$ are disjoint, we have from substitution of equations (9) and (2) into (12)

$$\Delta D_{ab}^{j+1} = \frac{1}{LN}[\sum_{\mathbf{x}_k \in \mathcal{P}_a^j} (\parallel \mathbf{x}_k - \mathbf{y}^{j+1}(a,b) \parallel^2 - \parallel \mathbf{x}_k - \mathbf{y}_a^j \parallel^2) + \sum_{\mathbf{x}_k \in \mathcal{P}_b^j} (\parallel \mathbf{x}_k - \mathbf{y}^{j+1}(a,b) \parallel^2 - \parallel \mathbf{x}_k - \mathbf{y}_b^j \parallel^2)]$$

$$= \frac{1}{LN}[(n_a^j \parallel \mathbf{y}_a^j - \mathbf{y}^{j+1}(a,b) \parallel^2) + (n_b^j \parallel \mathbf{y}_b^j - \mathbf{y}^{j+1}(a,b) \parallel^2)]. \tag{15}$$

But

$$\mathbf{y}_a^j - \mathbf{y}^{j+1}(a,b) = \frac{n_b^j}{n_a^j + n_b^j}(\mathbf{y}_a^j - \mathbf{y}_b^j) \tag{16}$$

and similarly,

$$\mathbf{y}_b^j - \mathbf{y}^{j+1}(a,b) = \frac{n_a^j}{n_a^j + n_b^j}(\mathbf{y}_b^j - \mathbf{y}_a^j). \tag{17}$$

Substituting (16) and (17) into (15) yields the following result:

$$\Delta D_{ab}^{j+1} = \frac{1}{LN} \frac{n_a^j n_b^j}{n_a^j + n_b^j} \parallel \mathbf{y}_a^j - \mathbf{y}_b^j \parallel^2 . \tag{18}$$

A quantizer Q^j having a codebook $\mathcal{C}^j$ with size A_j has a rate of $(1/L)\log_2 A_j$ bits per source sample since this is the required number of bits to encode each source sample. If some kind of entropy-coding such as a Huffman [11] or arithmetic code [12] is to be used, then the rate of the quantizer is $R^j = H_{Q^j}$, equal to the entropy of quantizer Q^j . But, if entropy coding is to be performed, a sensible design procedure should take it into account explicitly. The steps of the PNN procedure, as described, make no attempt to control the rate of the quantizer. This procedure can be modified to control rate by modifying the search strategy. The strategy must call for a search for a sequence of quantizers which optimizes jointly distortion and rate.

We next define a rate decrement, denoted by ΔR_{ab}^{j+1} as the decrease in entropy of quantizer $Q^{j+1}(a,b)$ relative to the entropy of quantizer Q^j by

$$\Delta R_{ab}^{j+1} = H_{Q^j} - H_{Q^{j+1}} \tag{19}$$

or,

$$\Delta R_{ab}^{j+1} = \frac{1}{LN}[(n_a^j + n_b^j)\log_2(n_a^j + n_b^j) - n_a^j \log_2(n_a^j) - n_b^j \log_2(n_b^j)]. \tag{20}$$

The average rate per sample $R^{j+1}(a,b)$ obtained when the vectors associated with the TS are quantized with $Q^{j+1}(a,b)$ is then,

$$R^{j+1}(a,b) = R^j - \Delta R_{ab}^{j+1}. \tag{21}$$

To discuss optimality in the above sense, let us consider the quantizer Q^j with distortion D^j and rate R^j. The next quantizer in the sequence is a merged (a,b) version $Q^{j+1}(a,b) = \mathcal{M}_{ab}[Q^j]$, for some value of (a,b) with a rate $R^{j+1}(a,b)$. Let us examine, in the distortion versus rate plane, two quantizers $\mathcal{M}_{ab}[Q^j]$ and $\mathcal{M}_{a'b'}[Q^j]$ with respect to Q^j. Figure 1 displays the points (R^j, D^j), $(R^{j+1}(a,b), D^{j+1}(a,b))$ and $(R^{j+1}(a',b'), D^{j+1}(a',b'))$. We can conclude upon examining this figure that, in the distortion-rate sense, the best sequence of quantizers generated with the merging operation is obtained by selecting the next member of the sequence in such a way that the line joining the corresponding (R,D) points has minimum inclination. This choice corresponds to the quantizer giving the least increase in distortion for the given decrease of rate. The optimality of this least ascendent rule guiding the choice of the next quantizer in the sequence can be guaranteed only when the distortion-rate curves are convex [13].

We will denote the ratio of distortion increment to rate decrement obtained in connection with quantizer $Q^{j+1}(a,b)$ by

$$s_{ab}^{j+1} = \frac{\Delta D_{ab}^{j+1}}{\Delta R_{ab}^{j+1}}. \tag{22}$$

The entropy-constrained PNN or ECPNN strategy to choose the new quantizer $Q^{j+1}(a,b) = \mathcal{M}_{ab}[Q^j]$ is to search all possible pairs (a,b) to find the minimum,

$$s_{\alpha\beta}^{j+1} = \min\ [s_{ab}^{j+1}]\ \forall\ a,b \in \mathcal{A}^j,\ a \neq b, \tag{23}$$

where α and β are the pair of cells selected for merging.

The entropy-constrained vector quantizer Q^j has to perform a mapping of the input source vector into a codevector according to (3). Let us say that the point (R^j, D^j) characterizes the values of rate and distortion achieved by the quantizer Q^j as obtained with the ECPNN procedure. The existing continuous D versus R curve passing through this point has a line with slope λ^j as tangent at this point. This slope can be approximated by the slope of the line joining points (R^{j-1}, D^{j-1}) and (R^{j+1}, D^{j+1}) also obtained by the ECPNN procedure. This slope is needed in order to define an entropy-constrained vector quantizer and is evaluated by

$$\lambda^j = -\frac{D^{j+1} - D^{j-1}}{R^{j+1} - R^{j-1}} \text{ for } j = 1, 2, \ldots, \mathcal{J}, \tag{24}$$

where $\mathcal{J} + 1$ is the stopping number of the recursion counter in the procedure. Setting $\lambda = \lambda^j$ in the quantization step in (3) defines the rule for quantizing a source where the output rate is constrained to R^j.

Next we develop the necessary modification for adding an alphabet constraint to our entropy-constrained vector quantizers. Remembering that by (15) the increase of distortion ΔD_{ab}^{j+1} is minimized if $\mathbf{y}^{j+1}(a,b)$ is assigned to be the centroid of the union partition cell. This assignment is made in the ECPNN procedure and creates a new codevector, not belonging to the initial set. Our objective is to develop an alphabet constrained version of ECPNN (AECPNN), where the final set of codevectors is a subset of the initial one. Therefore, $\mathbf{y}^{j+1}(a,b)$ must be assigned to be

$$\mathbf{y}^{j+1}(a,b) = \begin{cases} \mathbf{y}_a^j & \text{if } n_a^j \geq n_b^j \\ \mathbf{y}_b^j & \text{if } n_a^j < n_b^j, \end{cases} \tag{25}$$

because the more probable vector of the two gives less increase in distortion ΔD_{ab}^{j+1} . If we substitute the pseudo-centroid $\mathbf{y}^{j+1}$ into equation (15) the increase of distortion will be given by:

$$\Delta D_{ab}^{j+1} = \frac{\min(n_a^j, n_b^j)}{LN} \| \mathbf{y}_a^j - \mathbf{y}_b^j \|^2. \tag{26}$$

The decrease of rate is the same, since it is independent of the particular centroid and dependent only on the size (or probability) of the merged partition. We note that with this modification $\mathcal{C}^{j+1} \subset \mathcal{C}^j$. The search procedure continues to be the same as the ECPNN algorithm. The operation of the quantizer in the search process is given also by (3).

IV. Implemention Aspects

A. ECPNN Algorithm

According to Section III the strategy to choose the new quantizer $Q^{j+1}(a,b) = \mathcal{M}_{ab}[Q^j]$ is to search all possible s ratios using (23). At this point we can conclude that this procedure is quite computationally intensive when we operate with a large training set. This problem of large computation is solved by Equitz [3] using a data structure called $\mathcal{K}$-d trees. The aim of using this data structure is to develop a method which allows for $\log(N)$ multidimensional "nearest-neighbor" searches to be accomplished. As pointed out by Equitz, this method introduces an error due to the pre-classification of data and consequent suboptimal merges in the PNN sense. Here, we are not interested in this solution; our aim is to use the full search procedure. This problem of large computation can be circumvented if we initialize the design procedure with a high rate codebook instead of the training sequence, since, generally, VQ's are used with a maximum codebook size of 4096 codevectors. There are several good choices for an initial codebook, among them being the PNN algorithm plus $\mathcal{K}$-d trees and a recent algorithm reported by Hsieh in [14]. Instead of using one of these algorithms, we shall adopt an initial codebook designed by GLA, because we shall later compare the distortion-rate performance between ECPNN codebooks and ECVQ codebooks. The ECVQ algorithm is initialized with the GLA codebook, because this codebook is the maximum rate ECVQ solution for the given codebook size. This initial codebook, $\mathcal{C}^0 = \mathcal{C}^{GLA}$, defines a local minimum distortion quantizer of the original training set. With this initialization the initial rate R^0 is equal to H^{GLA} (the entropy of the GLA codebook), and the initial distortion D^0 is equal to D^{GLA}. We are now in position to describe the algorithm.

Algorithm

- Step 1 (Initialization)

  Set $j = 0; A_0 = M; \mathcal{C}^0 = \mathcal{C}^{GLA}; \mathbf{y}_m^0 = \mathbf{y}_m^{GLA}, n_m^0 = n_m^{GLA}, m = 1, \ldots, M; D^0 = D^{GLA}; R^0 = H^{GLA};$

- Step 2 (Find the best minimum PNN entropy quantizer $Q^{j+1}(a,b)$)

  Get the pair of indices (α, β) for the slope $s_{\alpha\beta}^{j+1}$ such that

$$s_{\alpha\beta}^{j+1} = \min_{a,b \in \mathcal{A}^j, a \neq b} [s_{a,b}^{j+1}]$$

- Step 3 (Quantizer and partition cell set update)

$$\begin{aligned} Q^{j+1} &= \mathcal{M}_{\alpha\beta}[Q^j] \\ D^{j+1} &= D^j + \Delta D_{\alpha\beta}^{j+1} \end{aligned}$$

$$R^{j+1} = R^j - \Delta R^{j+1}_{\alpha\beta}$$

- Step 4 (Recursion Counter)

  Set $j = j + 1$

- Step 5 (Stopping Rule)

  If stopping criterion is not met, return to Step 2, else stop.

One choice for the stopping rule at Step 5 is, for instance: stop if $R^{j+1} \leq R^t$ where R^t is a target rate. There are, of course, other choices. The ECPNN and AECPNN algorithms both have the same structure, only differing in the distortion increase equation and the choice of the codevector for the partition cell $\mathcal{P}_{ab}$ resulting from the $\mathcal{M}_{ab}$ operation. The AECPNN codebooks are faster to design because it is not necessary to compute new centroids.

Here we point out that three look-up tables are used to evaluate the distortion and rate increments in equations (18) and (20) in Step 2. Particularly, the factors $\frac{1}{n_a+n_b}$, $(n_a+n_b)\log_2(n_a+n_b)$ and $n_{a,b}\log_2 n_{a,b}$ can be computed and stored in arrays outside of Step 2. The use of these tables represents a very low memory cost since the algorithm does not work with the training sequence as do generalized Lloyd type algorithms. Another point is that divisions can be eliminated by modifying the strategy of choosing the next quantizer $Q^{j+1}(a,b)$. Recall that this corresponds to finding

$$s^{j+1}_{\alpha\beta} = \min \left[\frac{\Delta D^{j+1}_{ab}}{\Delta R^{j+1}_{ab}}\right] \ \forall\ a,b \in \mathcal{A}^j,\ a \neq b,$$

which may be achieved by pairwise progressive sorting such that

$$\frac{\Delta D^{j+1}_{ab}}{\Delta R^{j+1}_{ab}} \leq \frac{\Delta D^{j+1}_{cd}}{\Delta R^{j+1}_{cd}},$$

for distinct pairs (a,b) and (c,d). The latter comparison is equivalent to

$$\Delta D^{j+1}_{ab}\Delta R^{j+1}_{cd} \leq \Delta D^{j+1}_{cd}\Delta R^{j+1}_{ab}, \tag{27}$$

since all quantities are non-negative. Based on [10], additional savings in computation can be obtained by introducing a so-called distance table $\Delta\mathbf{D}$ that is defined as,

$$\Delta\mathbf{D} = \begin{pmatrix} 0 & \Delta D_{12} & \Delta D_{13} & \dots & \Delta D_{1A_0} \\ \Delta D_{12} & 0 & \Delta D_{23} & \dots & \Delta D_{2A_0} \\ \Delta D_{13} & \Delta D_{23} & 0 & \dots & \Delta D_{3A_0} \\ \vdots & \vdots & \ddots & \dots & \vdots \\ \Delta D_{1A_0} & \Delta D_{2A_0} & \Delta D_{3A_0} & \dots & 0 \end{pmatrix}, \tag{28}$$

where ΔD_{ab} is the increase of distortion as defined before. We can incorporate this matrix in our design algorithm in the initialization step and consequently eliminate the distortion calculation inside Step 2. Note that this symmetric matrix stores all weighted distances, and the null diagonal reflects the fact that $\Delta D_{aa} = 0$. In the Step 3 of the algorithm, it is necessary to update this matrix only for the row and column corresponding to the pair of merged partitions cells.

B. Computational Complexity

In this section we evaluate the computational complexity of the ECPNN algorithm. This study is applicable both to the ECPNN and AECPNN algorithms. We present a comparison between the ECPNN and ECVQ algorithms based on the number of additions, multiplications, logarithms and comparisons denoted respectively by ν_a, ν_m, ν_l and ν_c. Recall that divisions have been entirely replaced with the comparisons in (27) for ECPNN. Both the ECPNN and ECVQ algorithms are assumed to be initialized by the same codebook, which can be designed by various means, such as GLA, splitting algorithm, product of scalar codebooks, or truncated lattice points. It is important, however, to obtain a good initial codebook for both algorithms. We have not studied the sensitivity of ECPNN or ECVQ to initial codebook selection, but surmise that ECPNN may be the more sensitive of the two, as a poor initial choice of partitions may propagate when their unions form the partitions at lower rates.

The most computationally expensive part of the ECPNN algorithm is the search for the best pair to merge in Step 2. In this step the numbers ν_a, ν_m, ν_l and ν_c of additions, multiplications, divisions and comparisons are proportional to the number of non-distinct pairs to search, $\frac{1}{2}\,(A_j^2 - A_j)$, where A_j represents the number of reproduction vectors in the j-th iteration. When we proceed normally to find $A_0 = M$ (initial codebook size) points on the operational distortion-rate curve, $\hat{D}_{LM}(R)$, the total number of non-distinct pairs to search through M iterations in Step 2 is given by:

$$S_1 = \frac{1}{2} \sum_{A_j=1}^{M} (A_j^2 - A_j) = \frac{M^3 - M}{6} \approx \frac{M^3}{6}. \tag{29}$$

Note that we are assuming that the ECPNN is implemented using a progressive search where the search decreases at each iteration j by checking the zero values that are assigned to the unscaled probability mass array n_m^j for $m = 1, ..., A_0$.

Utilizing the symmetry of the matrix $\Delta\mathbf{D}$, the number of elements that require updating in a row and column in Step 3 is $A_j - 1$ for each iteration j. The total updated elements for the M iterations in Step 3 is:

$$S_2 = \sum_{A_j=1}^{M} (A_j - 1) = \frac{M(M-1)}{2} \approx \frac{M^2}{2}.$$

The total enumeration for each merit figure is given by:

$$\begin{aligned}
\nu_a &= 3S_1 + 2LS_2 \approx \frac{M^3}{2} + LM^2, \\
\nu_m &= 2S_1 + (L+3)S_2 \approx \frac{M^3}{3} + \frac{(L+3)M^2}{2}, \\
\nu_l &= 0, \\
\nu_c &= S_1 + \ M(M^2 - M) + M^2 \approx \frac{7M^3}{6};
\end{aligned} \tag{30}$$

The enumeration of the number of comparisons, ν_c, reflects the fact that we need to count the comparisons due to (27) and exclude the partitions that have n_a^j and $n_b^j = 0$ in Step 2, which corresponds respectively to the first two terms. The updating of $\Delta\mathbf{D}$ adds M^2 more comparisons in ν_c above.

Developing a similar analysis of the ECVQ algorithm by first evaluating the merit figures per iteration, we obtain $2NML$ additions, NML multiplications, M logarithms and NM comparisons, with N the number of training vectors (see [4]). If we let P denote the desired number of points on the $\hat{D}_{LM}(R)$ curve (number of desired codebooks), and F the number of iterations for convergence to a single point of the ECVQ algorithm, the merit figures will be given by:

$$\begin{aligned}
\nu_a &= 2FPNML, \\
\nu_m &= FPNML, \\
\nu_l &= FPM, \\
\nu_c &= FPNM.
\end{aligned} \tag{31}$$

Here, unlike ECPNN, we see that the complexity of ECVQ depends on the size of the training set. In order to make a comparison in terms of computational complexity between ECVQ and ECPNN, we would like to make the following observations about the operation of the ECVQ algorithm (see, e.g., [4], [5] and [15]):

(1) when designing a codebook to meet a given target rate R^t, one must start ECVQ with a high rate codebook and may have to run the algorithm repeatedly for different values of λ and initial sizes of codebooks, while searching for a codebook entropy that satisfies the target rate. This guessing of Lagrange multipliers and initial codebooks will generate other (R, D) points, before finding the target rate, R^t;

(2) generally for testing purposes or other applications, it is necessary to generate multiple codebooks with distortions obtainable over a whole range of rates. Hence, one must run ECVQ repeatedly on an increasing sequence of λ's, beginning with $\lambda^0 = 0$ that is associated with the high rate codebook, and increasing in any manner consistent, with obtaining the desired codebooks. The final codebook for λ^j is the initial codebook for λ^{j+1}.

In both ways of operation, multiple codebooks are generated. Since the first method of operation is very tedious, we have defined our complexity numbers in terms of the second method of operation.

Let us now make a comparison between ECPNN and ECVQ in terms of computational complexity using one example taken from [5]. In [5], ECVQ codebooks were used to compress motion compensation frame differences in a video coding application. The design parameters for the ECVQ algorithm were: 50 frames of spatial resolution of 348 pixels by 288 lines divided into blocks $L = 6 \times 6$, with an initial codebook size $M = 1024$ and with number of desired codebooks $P = 7$ between rates $R = 0.25$ and 0.05 (b/s). The factor of convergence F is assumed to be equal to three (this value is a typical average number, see, e.g, [4, pp. 35]). Because of its mode of operation, starting with the same initial codebook and covering the same range of rates, $M = 1024$ codebooks are generated using the ECPNN algorithm.

We can see by Table I that the ECPNN algorithm can indeed design $M = 1024$ codebooks in less time than the ECVQ algorithm takes to design $P = 7$. Considering the sum $\nu_a + \nu_m + \nu_l + \nu_c$ as a total merit figure, the numbers in Table I suggest that the ECPNN algorithm is roughly 160 times faster than ECVQ if the initialization is not counted. In our simulations, the initial codebooks were designed by the GLA algorithm, which has been the common procedure with ECVQ, since GLA achieves a local minimum of the distortion for a fixed size codebook. This type of initialization requires $2FNML$ multiplies, $FNML$ adds, and FNM comparisons, with F=10 being a typical maximum number of iterations for this algorithm. The resulting numbers for these operations are 1.02×10^{11} multiplies, 5.13×10^{10}, and 1.42×10^{9} comparisons, all of which are larger than the ECPNN figures for designing $P = 7$ codebooks. Adding these numbers to those in Table 1, the total number of operations would indicate that the ECPNN algorithm is still roughly 3 times faster than ECVQ. This figure should be considered a lower bound on comparative computational speed, because there are several other initialization procedures, as mentioned, all of which are faster than GLA. Therefore, one may conclude that ECPNN is 3 to 160 times faster in designing the seven codebooks, depending on the initial codebook design. The upper bound of 160 demonstrates the computational speed advantage of the procedural steps of ECPNN compared to ECVQ. This advantage would increase with the number of codebooks to be designed.

V. Experimental Results

In this section we present a discussion of the results obtained when the ECPNN and AECPNN codebooks are used to quantize the output of a discrete-time i.i.d Gaussian source with zero mean ($m_X = 0$) and unit variance ($\sigma_X^2 = 1$), subject to the mean square error criterion. The distortion versus rate results are compared to the results obtained with ECVQ and AECVQ algorithms and with the well known rate-distortion function $R(D)$, which is the minimum rate achievable for a given average distortion D. For this source and distortion measure, this function is [13]:

$$R(D) = \max[0, \frac{1}{2}\log_2(\frac{\sigma_X^2}{D})], \tag{32}$$

in b/s, where D is the mean squared error. To generate the results shown in this section, the size of the training sequence and test sequences were 40,960 samples each. The initial codebook used by ECPNN, AECPNN, ECVQ and AECVQ algorithms was designed using the GLA with codebook size of 256 codevectors. In this study 4-dimensional quantizers were utilized. We present in Figure 2 the *signal-to-quantization noise ratio* $\mathrm{SQNR} = 10\log_{10}(\frac{\sigma_X^2}{D})$ in dB versus rate of vector quantizers using ECPNN, AECPNN, ECVQ and AECVQ codebooks along with the $R(D)$ curve. The operational points were found, by quantizing the test sequence using the criterion in (3). Clearly, ECPNN and ECVQ have indistinguishable performance. AECPNN and AECVQ have almost the same performance, except for an insignificant advantage to AECVQ below 0.4 b/s. Another interesting point, is that both AECPNN and AECVQ for the rate $R = 0$ reach an SQNR slightly less than 0 dB. This effect can be explained by the fact that both algorithms have constrained alphabets where the centroid of the entire training set is not the reproduction level when there is just one partition. Consequently the resultant vector is not a good estimate of the centroid of data outside the TS. We can see by this Figure, that the double constrained VQ's have very close performance with the single constrained VQ's.

Results for a first-order Gaussian autoregressive (AR(1)) given by the model

$$X(n) = aX(n-1) + W(n),$$

where the correlation coefficient $a = 0.9$, unit variance ($\sigma_X^2 = 1$) and $W(n)$ is an i.i.d. zero mean Gaussian random sequence with variance $\sigma_W^2 = (1 - a^2)\sigma_X^2$, are shown in Figure 3 for the case of a 4-dimensional VQ. The training and test sequences each contained 40,960 sample vectors. The initial codebook used by ECPNN, AECPNN, ECVQ and AECVQ algorithms was designed using the GLA with codebook size of 256 codevectors. The rate-distortion function for stationary Gaussian sources [13] is evaluated from the parametric equations given by:

$$R_\theta = \frac{1}{2\pi} \int_{-\pi}^{\pi} \max \left[0, \frac{1}{2} \log_2 \frac{S_{XX}(\omega)}{\theta} \right] d\omega, \quad (33)$$

$$D_\theta = \frac{1}{2\pi} \int_{-\pi}^{\pi} \min \left[\theta, S_{XX}(\omega) \right] d\omega, \quad (34)$$

where $S_{XX}(\omega)$ is the discrete-time source power spectral density (psd) and the nonzero portion of the $R(D)$ (or $D(R)$) curve is generated as the parameter θ traverses the interval $0 \leq \theta \leq$ *ess sup* $S_{XX}(\omega)$, where *ess sup* denotes the essential supremum of a function.

This experiment shows the same trends as with the i.i.d. source. There is essentially no difference in performance between the ECPNN approach and the ECVQ approach; AECPNN and AECVQ have close performances to each other and are only slightly below those of their non-alphabet constrained counterparts; and there is a small degradation of AECPNN below 0.4 b/s.

In Figure 4, we plot for this autoregressive source the codebook size versus the rate. By this Figure we can note that ECPNN codebooks are smaller than ECVQ codebooks. This behavior could be explained by the fact that ECPNN design necessarily decreases the size of the previous codebook using the merge operator. The same does not happen with the ECVQ algorithm because it always tries to populate the previous set of partitions defined by the previous codebook used. This difference in codebook sizes gives to the ECPNN codebooks operational advantages over ECVQ codebooks. For example, consider the rate of 1.0 b/s (and a SQNR of $\approx$ 10 dB) the ECPNN codebook has $\frac{1}{3}$ the number of codevectors of the ECVQ codebook. We can note that AECPNN and AECVQ have almost identical behavior in terms of numbers of vectors used for encoding.

We should also note that the AECPNN codebook is larger than the ECPNN codebook, while the AECVQ codebook is smaller than that of ECVQ. A merge of given regions in AECPNN produces the same decrease in rate, but a larger increase in distortion as the same merge in ECPNN. More codevectors are then required to lower the increase in distortion in a merge to track the distortion-rate performance of ECPNN. The opposite behavior of AECVQ and ECVQ is the result of re-assigning probabilities among the same number of partitions at each rate. The operation of the AECVQ algorithm shows a greater tendency to assign zero probability partitions than ECVQ, resulting in a smaller codebook. The reason for this tendency would require delving into the details of AECVQ, which is too far afield from the subject at hand.

VI. A Multi-rate Pyramid Coder

The excellent results obtained by Woods and O'Neil [16] and Gharavi and Tabatabai [17] for scalar DPCM and PCM coding of image sub-bands initiated the current flurry of activity of subband or hierarchical coding. The subband splitting schemes above differ in that one involves a pyramid splitting of the image [17], producing an octave-step division of the frequency axes, whereas the other uses equal size subbands with uniform divisions of the frequency axes [16].

In our scheme the image is split into a pyramid, and each pyramidal subband is coded independently. In this work we use the ECPNN and AECPNN design algorithms for designing the entropy-constrained codebooks which are used for encoding the subbands. An ECPNN system for coding an image pyramid in a multirate fashion requires the design of multiple codebooks of different rates for every image subband. An AECPNN system for coding an image pyramid requires only one codebook (set of codevectors) per subband, but to achieve the codebook entropy at every desired rate, a different set of binary codewords is required. Therefore an AECPNN system requires one codebook and a set of codewords for each subband.

A. Designing of Codebooks of Image Pyramids

A pyramid of three levels was constructed by splitting the image into four subbands, and then recursively splitting two more times the lowpass subband. The spectral decomposition is shown in Figure 5. The nine-tap filter of [18] was used in the decomposition to obtain this 3-level pyramid. At each level the filtering of the lowpass image generates four components, a lowpass (LL), a vertical highpass (LH), a horizontal highpass (HL) and a diagonal highpass (HH) component.

The encoder design procedure for all subbands is the same. The initial codebooks for the

ECPNN and AECPNN design algorithms are obtained using the GLA. Different initial size codebooks are chosen for each subband according to the estimate of the highest required rate. Once we have obtained the GLA codebook (slope $\lambda^0 = 0$), along with its associated input set partition and codevector probabilities, we use it for the initialization of the ECPNN and AECPNN algorithms, as mentioned in Section IV.

For the design of the multiple rate codebooks for each subband, we use a training set of 10 (512 x 512) images. As pointed out in [19], since images differ in their spectral properties, normalized codebooks were designed for each subband by scaling all training vectors by the standard deviation of its associated subband. All the images in the TS were converted to zero mean sources by subtracting out their means. Therefore, the mean of the image to be coded and the variances of its ten subbands need to be sent to the decoder. This bookkeeping information corresponds to a negligible increase in the final rate reported here, which is the code rate alone.

We designed 10 codebooks for each subband using the ECPNN algorithm and 10 sets of codewords for each subband using the AECPNN algorithm. These codebooks will be used by the rate allocation algorithm for coding the input image with a desired target rate. We have designed, for the first and second levels of the pyramid, codebooks with entropy rates ranging from 2.0 to 0.0 b/pixel, with a rate decrement of 0.4 b/pixel between codebooks with vector dimension equal to 4. For the subbands, {LH, HL, HH } of the third level we designed codebooks ranging from 4.0 to 0.0 b/pixel with the same rate decrement as the first and second levels of the pyramid but with vector dimension equal to 2. Finally, for the subband LL3 we designed scalar codebooks with rates ranging from 7.0 to 0.0 b/pixel with rate decrements of 1.0 b/pixel. We have chosen smaller vectors for the level 3 because these subbands have to be coded at high rates. Generally the level 3 contains the highest activity (highest variance). The lowest frequency band of level 3, LL3, is coded by scalar quantizers designed by the ECPNN and AECPNN algorithms. For this band, perhaps a more suitable scheme is a DPCM or a DCT system to remove the interpixel correlation. The latter approach for coding the LL3 subband was not adopted, because we wanted to test our quantizers in all bands.

B. Rate Allocation

The problem of finding an optimal bit assignment for subband vector quantization can be stated as follows:

• **Find the minimum distortion D over the set of subband rates $\{R_l\}$ such that the overall rate has a given value R**

In order to solve the problem above, a set of distortion-rate points must be found for each subband, as has been done by the ECPNN and AECPNN design algorithms. Chou *et al.* [20] suggested an operational method which finds an optimal solution for this situation. In [20], the idea is to organize the distortion-rate points on a tree and "prune" the tree according the generalized Breiman, Friedman, Olshen, and Stone (BFOS) algorithm, in order to find the best combination of quantizers to encode the data, given a target rate or distortion. Recently, Riskin reported this algorithm in [21], which corresponds to the algorithm used here.

In our context of bit-allocation the BFOS algorithm can be used as follows: construct a tree T with l subtrees (actually each subtree is a unary tree) where each subtree represents a subband or class. In each subtree we have k nodes, where each node is represented by $(R, D, \lambda)^k$ point found by the ECPNN or AECPNN algorithm operating on the training image set. The topmost node in the subtree corresponds to the zero-rate codebook; i.e, $\lambda = \infty$. Moving down in the subtree, we decrease the λ values and at the bottom of this unary tree we find the (R, D) point that corresponds to the minimum λ value (found by the ECPNN or AECPNN algorithm); this node will be called leaf node. According to this organization, if we move downward along the subtree (or the tree), the value of rate of each node increases and the distortion decreases. Besides the $(R, D, \lambda)^k$ point stored in each node, we also store a parameter called Λ. This parameter is defined to be the ratio $\Delta D/\Delta R$, where ΔD and ΔR respectively denote the magnitudes of the difference between the distortion and rate at the current node and the leaf node. The value Λ is the slope that trades off D and R in each class or subband S. In Figure 6 the tree organization is displayed. Note that, as we proceed downward along the tree, the value of rate at each node increases while the value of distortion decreases.

If we denote the full tree by T_I, the generalized BFOS algorithm will prune off the branches of the tree T_I in order to form the final pruned tree T_F. In this pruning operation the algorithm obtains a sequence of trees, where each intermediate tree T^{i+1} is obtained by pruning the node that belongs to a particular subtree, S, having the smallest Λ in the T^i tree. After pruning, the subtree, S, has a new leaf node and consequently the Λ ratios must be re-calculated for this particular subtree. The algorithm ends when the sum of the leaf node rates drops below the target rate (actually one can use other stopping rules). The codebook used to encode each subband corresponds to the codebook specified by the leaf nodes of the final pruned tree T_F. The distortion and slope λ values used in this computation were renormalized by the variances of each subband. Here, we want to emphasize that the value λ is not necessary for the algorithm. In Figure 7 is illustrated the encoding process for a given subband signal. We would like to point out that, due to the inevitable mismatch between the statistics of the training sequence and the source image, there is a mismatch between the rate and distortion estimated by the BFOS allocation procedure and the actual quantization rate and distortion of the input image using the codebooks selected by the BFOS. But, with the training set of ten images used in our simulations, this mismatch proved to be very small for a variety of test images. The interested reader is directed to [8] for further results.

C. *Experimental Results*

Computer simulations have been implemented to evaluate the proposed coding system. The encoding rate reported here corresponds to the weighted sum of the individual encoding rates of each subband and is given by the following counting

$$R = \sum_{k=1}^{3} \sum_{l=1}^{3} \frac{H_{Qkl}}{4^k} + \frac{H_{QLL3}}{4^3}, \tag{35}$$

where k indicates the level of the pyramid and l a particular subband in the set, {LH, HL, HH }, H_Q is first-order entropy per pixel obtained through the VQ encoding. As the objective fidelity criterion, we have adopted the peak-to-peak SNR between the original and the reconstructed image. The images *Lena* and *Boats*, which are outside the training sequence, were used to test the performance of the system. *Lena* is a 512×512 USC database image and *Boats* is a 512×512 ISO image. The *Boats* image originally had a size of 576×720, which we have truncated for convenience. In Table II is reported the peak-to-peak SNR with the respective rates for *Lena*. The target rates given as input to the BFOS algorithm were $\frac{1}{3}$, $\frac{1}{2}$ and 1.0 b/pixel.

The results in Table II can be considered excellent in terms of PSNR. The subjective quality of these images are excellent between rates 0.5 to 1.0 b/pixel. Some degradation can be noticed in the 0.33 b/pixel image, especially the smoothing of the background and some ringing of the edges on the hat, but the image is still pleasing. We should have in mind that in order to achieve this low rate some subbands have zero rate allocation, especially those that belong to the first level of the pyramid. The ECPNN and AECPNN images do not have any visible differences. In Figure 8 the original *Lena* and the reconstructed images using ECPNN codebooks with rates $R = 0.34$ b/pixel, $R = 0.51$ b/pixel and $R = 1.1$ b/pixel are displayed. The image reconstructions for AECPNN are not displayed due to space constraints.

The comparisons of different coding techniques is generally a difficult task. In the following, we compare our PSNR values with the best results yet reported using subband techniques upon the *Lena* image and the JPEG standard implementation of [22] coded at rates approximately 0.5 b/pixel. Unhappily, we are unable to compare to the excellent results in [23] achieved with AECVQ applied to pyramid decompositions of smaller (256×256) images.

In comparison with other results from the literature, we find that our subband coder is quite competitive, achieving PSNR's of 36.1 and 36.2 dB at rates of 0.51 b/pixel and 0.55 b/pixel for ECPNN and AECPNN, respectively. The adaptive entropy coded subband coder of Kim and Modestino [24], which utilizes predictive entropy-constrained vector quantizers designed by the ECVQ algorithm, achieves a PSNR of 35.98 dB, at a rate of 0.48 b/pixel, for encoding the monochrome version of *Lena* (512×512) image. The subband coder based on entropy-constrained trellis coded quantizers proposed by Sriram and Marcellin [25] achieves a PSNR value of 36.6 dB at 0.48 b/pixel. Two points should be emphasized about this work: the lowest subband is DCT coded and the entropy-constrained trellis coded quantizers in the other subbands give an effective block length much larger than dimension 4, the highest vector dimension used in our work. In the embedded hierarchical image coder of Shapiro [26] the PSNR value is 35.97 dB with 0.5 b/pixel as encoding rate. Through improvements of Shapiro's coder, Said and Pearlman [27] have obtained a PSNR of 36.8 dB at 0.5 b/pixel. Both schemes perform a 5-level hierarchical subband decomposition, where the latter scheme employs multiple arithmetic coders with different adaptive models and the former employs a single arithmetic coder with an adaptive model. The JPEG results for *Lena* using adaptive variable length Huffman codes were 34.9 dB at 0.51 b/pixel. Here, we would like to point out that in our opinion the JPEG scheme is a medium bit-rate coder and comparisons in a low-bit rate region might be unfair. The JPEG image looks very blocky in the background and on the face. These characteristics are not adequately reflected in the PSNR measures.

In other experiment we have encoded the image *Boats* shown in Figure 9. This image is far more difficult to compress with high quality. Among the impediments are the thin lines, which represent the masts of the boats. We present results with a target rate of 1.0 b/pixel. Using the ECPNN codebooks we obtained 38.9 dB at 1.1 b/pixel. The AECPNN, as was expected, is very close to the ECPNN result, 38.9 dB at 1.2 b/pixel. The images are subjectively very good when compared to the original at this rate. In Figure 10 is displayed the result using ECPNN codebooks. Again, due to space constraints we do not show the AECPNN result, which attains the same subjective quality. In lowering the bit rate below 0.70 b/pixel, some expected ringing occurs and the sky loses its granularity. We have compared our images with the JPEG compressed image having 38.4 dB PSNR at 1.1 b/pixel. Comparing subjectively our image and the JPEG image, the JPEG image may have a slight edge in one aspect: the middle-left mast of a boat seems to be more faithfully reconstructed. However, we would like to emphasize that for lower rates ($<$ 0.70 b/pixel), our scheme performs better than the JPEG scheme.

Concluding, we would like to point out that we have used an M-ary non-adaptive arithmetic coder designed by Popat [12] for the noiseless coding of the codevector indices. The biggest difference between the entropies reported in this section and corresponding output bit-rates of this coder is 0.008 b/pixel.

VII. Conclusions

A new design technique for entropy-constrained vector quantization has been proposed. The algorithm, called ECPNN, is based on the same partitional clustering technique used by Equitz [3] in his PNN algorithm. The sequence of quantizers recursively generated by ECPNN algorithm has an operational distortion-rate function that closely approximates the performance of ECVQ, the iterative entropy-constrained quantizer design method proposed by Chou, Lookabaugh and Gray [4]. This fact has been verified by quantizing Gaussian memoryless and autoregressive sources. It was verified, that the size of the ECPNN codebook generally is smaller than the ECVQ codebook when compared at the same distortion-rate performance. The main advantage of the ECPNN algorithm is the faster design when compared with ECVQ. We show, that it is possible to design alphabet- and entropy-constrained vector quantizers using the pairwise nearest neighbor partitional technique, which we named AECPNN. The results of AECPNN quantizing synthetic sources show performance as good as the AECVQ [6] algorithm introduced by Rao and Pearlman [6]. The AECPNN and AECVQ algorithms simplify the encoding-decoding process because they decrease the amount of memory necessary for operating a multi-rate coder and permit simplifications on the search procedure, since the same codebook is used at all rates for a given source. We have used our design techniques to develop codebooks for a multi-rate coder for image pyramids where it is especially effective for low bit rates. The results show that the AECPNN codebooks performs as well as the ECPNN codebooks in terms of subjective and objective quality. Future developments can incorporate memory in our design technique considering high-order statistics, as Chou and Lookabaugh [15] propose in the case of the ECVQ algorithm in the form of a conditional entropy-constrained vector quantizer design called CECVQ.

References

[1] 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.

[2] S. P. Lloyd, "Least squares quantization in PCM", *IEEE Trans. on Information Theory*, vol. IT-28, pp. 129–137, March 1982.

[3] W. H. Equitz, "A new vector quantization clustering algorithm", *IEEE Trans. on Acoustics, Speech and Signal Processing*, vol. ASSP-37, pp. 1568–1575, Oct. 1989.

[4] P. A. Chou, T. Lookabaugh and R. M. Gray, "Entropy-constrained vector quantization", *IEEE Trans. on Acoustics, Speech and Signal Processing*, vol. ASSP-37, pp. 31–42, Jan. 1989.

[5] T. Lookabaugh, E. A. Riskin, P. A. Chou and R. M. Gray, "Variable rate vector quantization for speech, image and video compression", *IEEE Trans. on Communications*, vol. COM-41, pp. 186–199, Jan. 1993.

[6] R. P. Rao and W. A. Pearlman, "Alphabet-constrained vector quantization", *IEEE Trans. on Information Theory*, vol. IT-39, pp. 1167–1179, July 1993.

[7] W. A. Finamore, D. P. de Garrido and W. A. Pearlman, "A clustering algorithm for entropy-constrained vector quantizer design", *in Visual Commun. Image Processing '90, Proc. SPIE 1360*, pp. 837–846, Lausanne, Switzerland, Nov. 1990.

[8] D. P. de Garrido, W. A. Pearlman and W. A Finamore, "Vector quantization of image pyramids with the ECPNN algorithm", *in Visual Commun. Image Processing '91: Visual Communications, Proc. SPIE 1605*, pp. 221–232, Boston, MA, November 1991.

[9] D. P. de Garrido and W. A. Pearlman, "Multirate codebook design with the entropy-constrained pairwise nearest neighbor (ECPNN) algorithm", *in Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, pp. 373–376, 1992.

[10] D. Wishart, "An algorithm for hierarchical classifications", *Biometrics*, vol. 22, pp. 165–170, March 1969.

[11] D. A. Huffman, "A method for the construction of minimum-redundancy codes", *Proc. IRE*, vol. 40, pp. 1098–1101, Sept. 1952.

[12] A. C. Popat, "Scalar quantization with arithmetic coding", Master's thesis, Electrical Engineering and Computer Science Dept., Massachusetts Institute of Technology, 1990.

[13] T. Berger, *Rate Distortion Theory*, NJ: Prentice-Hall, Englewood Cliffs, 1971.

[14] C.-H. Hsieh, "DCT-based codebook design for vector quantization of images", *IEEE Trans. on Circuits and Systems for Video Technology*, vol. CSVT-2, pp. 401–409, Dec. 1992.

[15] P. A. Chou and T. Lookabaugh, "Conditional entropy-constrained vector quantization of linear predictive coefficients", *in Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, pp. 197–200, 1990.

[16] J. W. Woods and S. D. O'Neil, "Subband coding of images", *IEEE Trans. on Acoustic, Speech and Signal Processing*, vol. ASSP-34, pp. 1278–1288, Oct. 1986.

[17] H. Gharavi and A. Tabatabai, "Sub-band coding of monochrome and color images", *IEEE Trans. on Circuits and Systems*, vol. CAS-35, pp. 207–214, Feb. 1988.

[18] E. H. Adelson and E. Simoncelli, "Orthogonal pyramid transforms for image coding", *in Proc. SPIE Conf. on Visual Commun. Image Processing*, pp. 50–58, Cambridge, MA, Oct. 1987.

[19] B. Mahesh and W. A. Pearlman, "Multiple-rate structured vector quantization of image pyramids", *Journal of Visual Commun. and Image Representation*, vol. 2, pp. 103–113, June 1991.

[20] P. A. Chou, T. Lookbaugh and R. M. Gray, "Optimal pruning with applications to tree-structured source coding and modelling", *IEEE Trans. on Information Theory*, vol. IT-35, pp. 299–315, March 1989.

[21] E. A. Riskin, "Optimal bit allocation via the generalized BFOS algorithm", *IEEE Trans. on Information Theory*, vol. IT-37, pp. 400–402, March 1991.

[22] A. C. Hung, "Portable video research group (PRVG)-JPEG codec 1.1", Stanford University, 1993.

[23] R. P. Rao and W. A. Pearlman, "Alphabet- and entropy-constrained vector quantization of image pyramids", *Optical Engineering*, vol. 30, pp. 865–871, July 1991.

[24] Y. H. Kim and J. W. Modestino, "Adaptive entropy coded subband coding of images", *IEEE Trans. on Image Processing*, vol. IP-1, pp. 31–48, Jan. 1992.

[25] P. Sriram and M. W. Marcellin, "Wavelet coding of images using trellis coded quantization", *in Visual Information Processing '92, Proc. SPIE 1705*, pp. 238–247, Orlando, FL, April 1992.

[26] J. M. Shapiro, "An embedded wavelet hierarchical image coder", *in Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, pp. 657–660, 1992.

[27] A. Said and W. A. Pearlman, "Image compression using the spatial-orientation tree", *in Proc. IEEE Int. Symp. on Circuits and Systems*, pp. 279–282, 1993.

Table I

No. of operations

OPERATION	ECPNN	ECVQ
ν_a	0.57×10^9	215×10^9
ν_m	0.37×10^9	110×10^9
ν_l	0	21×10^3
ν_c	1.2×10^9	2×10^9

Table II

Simulation results for *Lena*

Codebooks	R(b/p)	PSNR (dB)
ECPNN	0.34	34.2
AECPNN	0.36	34.3
ECPNN	0.51	36.1
AECPNN	0.55	36.2
ECPNN	1.1	39.1
AECPNN	1.2	39.2

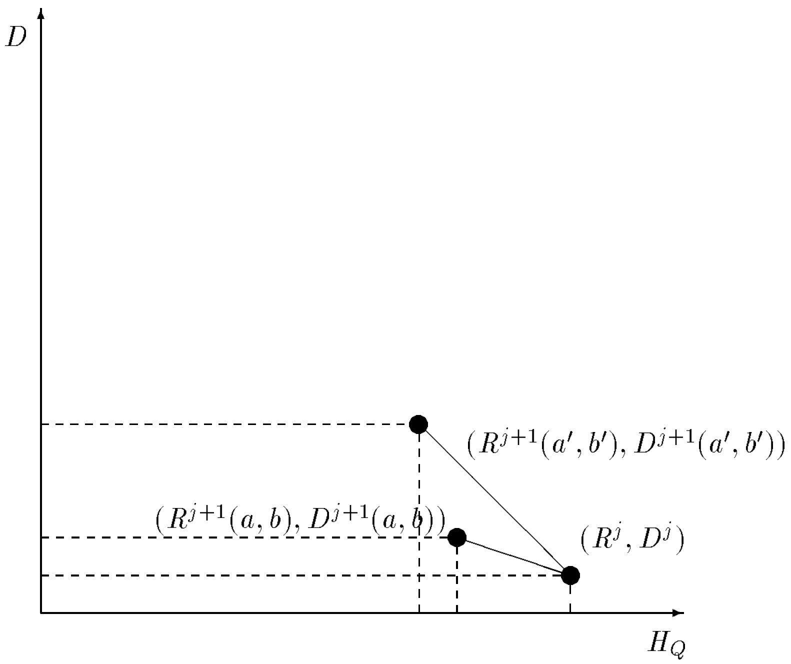


Fig. 1. Distortion-rate points.

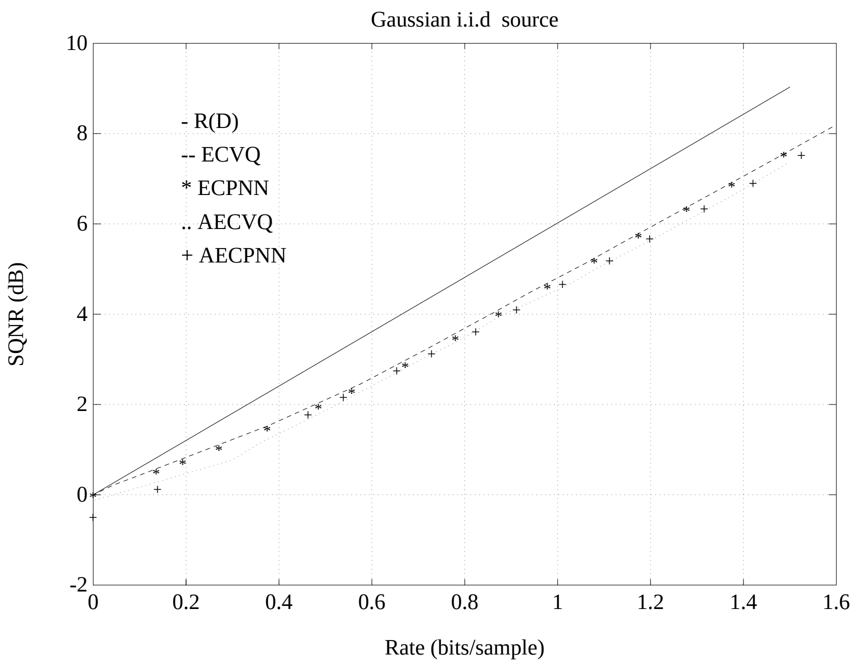


Fig. 2. Distortion-Rate performance for L=4 of ECPNN and ECVQ codebooks for i.i.d Gaussian source, outside the TS.

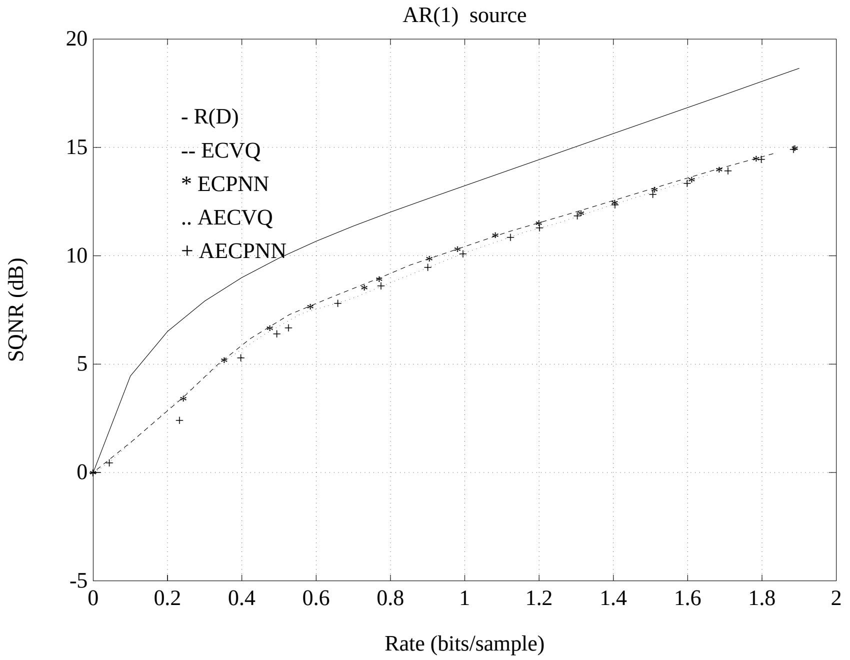


Fig. 3. Distortion-Rate performance for L=4 of ECPNN and ECVQ codebooks for AR(1) source, outside the TS.

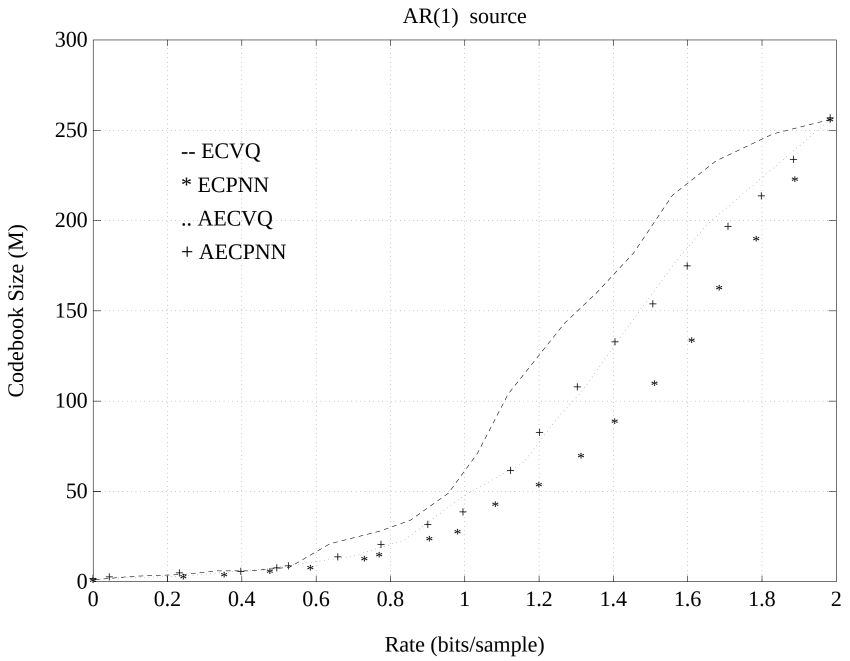


Fig. 4. Size of codebooks designed by ECPNN and ECVQ algorithms, for AR(1) source.

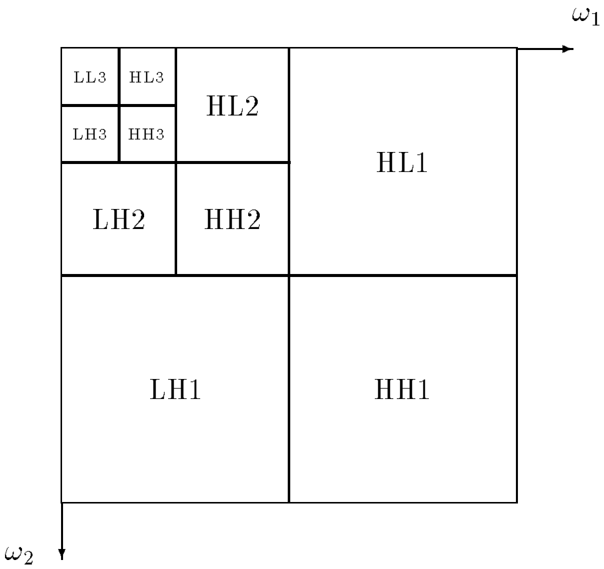


Fig. 5. Subband decomposition.

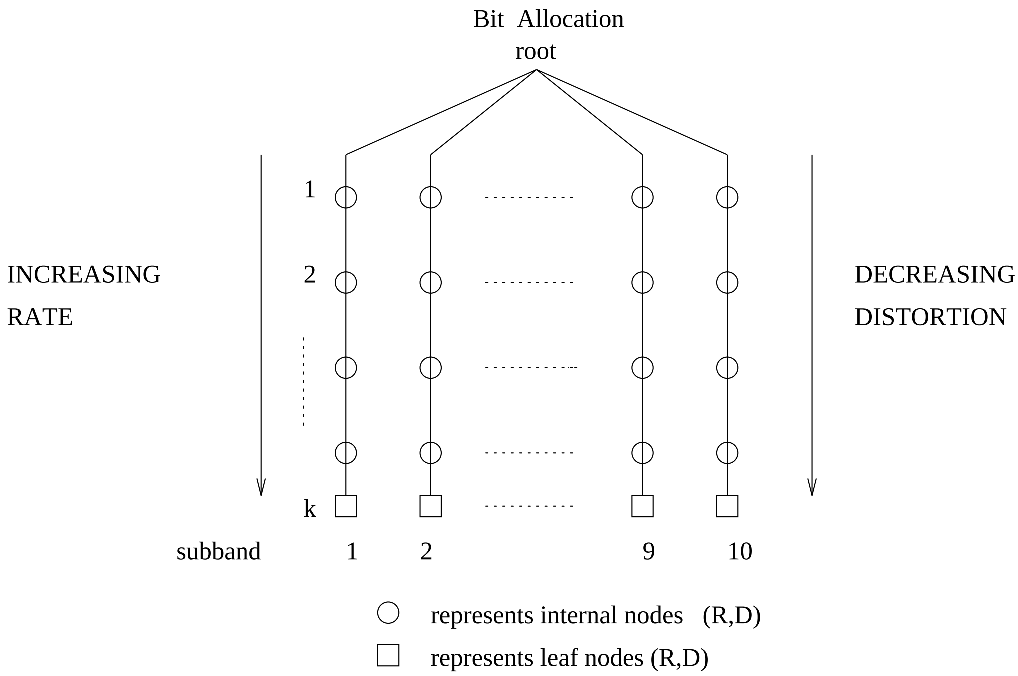
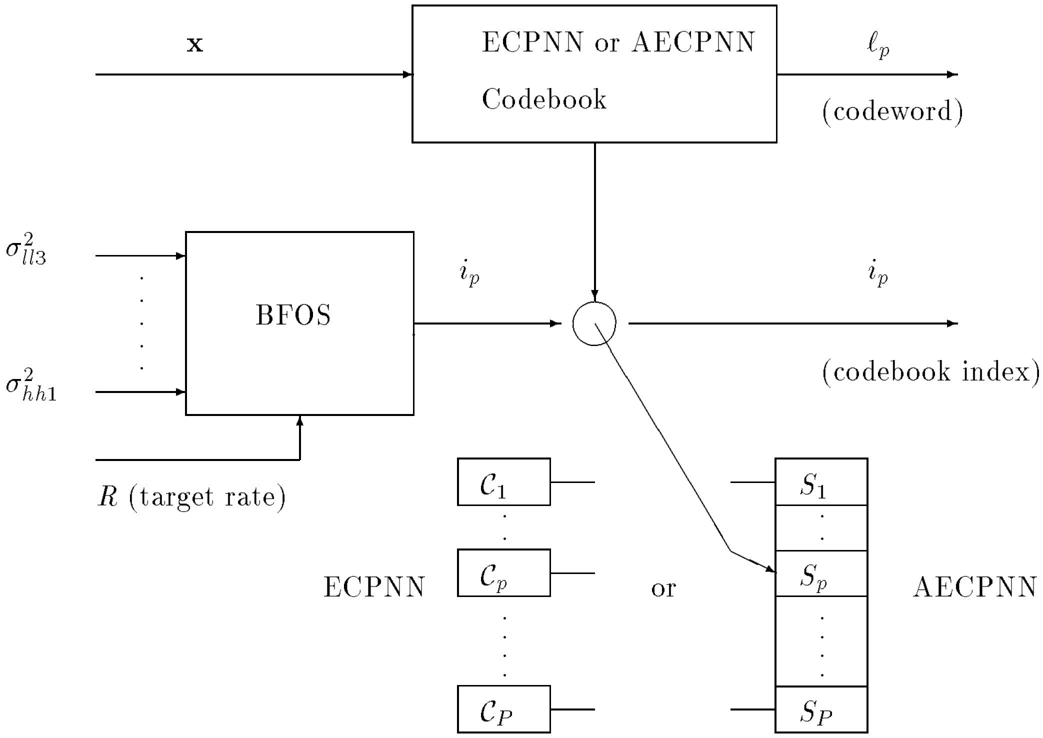


Fig. 6. Bit allocation using the BFOS algorithm.



Obs: ECPNN - P sets of codewords and P sets of codevectors

AECPNN - P sets of codewords and 1 set of codevectors

Fig. 7. VQ encoder.

Fig. 8. Top-left original *Lena* image,

top-right reconstructed image with $R = 1.1$ b/pixel,

bottom-left reconstructed image with $R = 0.51$ b/pixel and

bottom-right reconstructed image with $R = 0.34$ b/pixel.

Fig. 9. Original *Boats* image.

Fig. 10. *Boats* image coded at $R = 1.1$ b/pixel using ECPNN codebooks.