A Hierarchical Dynamic Programming Approach to Fixed-Rate, Entropy-Coded Quantization

A. K. Khandani, *Member, IEEE*

***Abstract*—In quantization of any source with a nonuniform probability density function, the entropy coding of the quantizer output can result in a substantial decrease in bit rate. A straightforward entropy coding scheme faces us with the problem of variable data rate. A solution in a space of dimensionality N is to select an appropriate subset of elements in the N-fold Cartesian product of a scalar quantizer and represent its elements with codewords of the same length. The drawback is that the *search/addressing* of this scheme can no longer be achieved independently along the one-dimensional subspaces. A reasonable rule is to select the N-fold symbols of the highest probability. For a memoryless source, this is equivalent to selecting the N-fold symbols with the lowest *additive* self-information. In this case, due to the additivity property of the self-information, the selected subset has a high degree of structure which can be used to substantially decrease the search/addressing complexity. In this work, a dynamic programming approach is used to exploit this structure. We build our recursive structure required for the dynamic programming in a hierarchy of levels. This results in several benefits over the conventional trellis-based approaches. Using this structure, we develop efficient rules (based on aggregating the states) to substantially reduce the search/addressing complexities while keeping the degradation in performance negligible.**

***Index Terms*—Scalar quantization, vector quantization, fixed-rate, entropy coding, hierarchical dynamic programming, addressing, decoding.**

I. Introduction

Consider the problem of quantizing a source with a nonuniform probability density function. If the dimensionality of the quantizer is not high enough, the entropy coding of the output can result in a substantial decrease in bit rate. A straightforward entropy coding method presents us with the problem of variable data rate. Also, if the bit rate per quantizer symbol is restricted to be an integer, we are potentially subject to wasting up to one bit of data rate per quantizer output. A solution in a space of dimensionality N is to code the N-fold Cartesian product of a scalar quantizer. To avoid having a variable data rate, one can select an appropriate subset of the N-fold symbols and represent its elements with codewords of the same length. In such a block-based source-coding scheme, as some of the elements in the N-fold Cartesian product space are not allowed, the *search* for the quantizer partition (decoding) and also the corresponding *addressing, reconstruction* processes (to be defined later) can no longer be achieved independently along the one-dimensional (1-D) subspaces. Obviously, this results in an increase in the complexity of these operations.

One class of schemes are based on using a subset of points from a lattice (quantization lattice) bounded within the Voronoi region around the origin of another lattice (shaping lattice) [1]. In this case, the selected subset forms a group under vector addition modulo the shaping lattice. This group property is used to facilitate the complexity of the underlying operations.

Another class of schemes is based on selecting the N-fold symbols with the lowest additive self-information. This approach is traditionally denoted as the geometrical source coding [2], [3]. In this case, the selected subset has a high degree of structure which can be used to substantially reduce the complexity. A method for exploiting this structure based on using a dynamic programming approach with the states corresponding to the length of the codewords is used by Laroia and Farvardin in [4] and [5]. Subsequently, Balamesh and Neuhoff, in [6], employ some complementary techniques to further reduce the complexity. In the present work, we use a more advanced approach to dynamic programming showing improvement with respect to the schemes of [4]–[6].

The key point is to use the additivity property of the self-information, in conjunction with the additivity property of the distortion measure, to decompose the underlying operations into the lower dimensional subspaces. This decomposition avoids the exponential growth of the complexity. The core of the scheme, as in any problem of dynamic programming, is a recursive relationship. We build our recursive structure in a hierarchy of levels where each level involves the Cartesian product of two lower dimensional subspaces. This results in several benefits over the conventional trellis-based approaches used in [4]–[6]. By effectively quantizing the state space, we obtain suboptimum methods with low complexity and negligible performance degradation.

II. Basic Structure

Consider a memoryless source and a scalar quantizer composed of M points. In the N-fold Cartesian product of this quantizer, we obtain M^N, N-D points. The final vector quantizer is selected as a subset of the N-D points composed of T elements. Each N-D point is represented by a codeword composed of $\lceil \log_2 T \rceil$ bits. The N-D reconstruction vectors are denoted as $\boldsymbol{r}_i$, $i = 0, \cdots, T-1$. For a given source vector $\boldsymbol{x}$, the quantization rule (decoding) is to find the reconstruction vector $\boldsymbol{r}_i$ which has the minimum square distance to $\boldsymbol{x}$, addressing is to produce the index i when $\boldsymbol{r}_i$ is selected, and reconstruction is to reproduce $\boldsymbol{r}_i$ from the index i.

Assume that the induced self-information and the expected value of the symbols mapped to the jth 1-D point are equal to c_j and r_j, respectively. The self-information associated with a 1-D point is considered as a cost associated with that point. The selection rule in the N-D space is to keep the N-D points with the lowest overall additive cost. The N-D reconstruction vectors are obtained by concatenating the corresponding 1-D reconstruction levels, namely r_j's. Assuming that the performance is measured in terms of the mean-square distance, the search operation is formulated as

$$\begin{aligned} \text{Minimize} \quad & \sum_{i=0}^{N-1} (x_i - r_{j_i})^2 \\ \text{Subject to:} \quad & \sum_{i=0}^{N-1} c_{j_i} \le C_{\max} \end{aligned} \tag{1}$$

where j_i is the index of the point selected along the ith dimension and $C_{\max}$ is the maximum value of the allowable cost in the N-D space. The objective of the search operation is to find an allowable N-D point (satisfying the cost constraint) resulting in the minimum distortion. The immediate approach to solving (1) is to perform an exhaustive search.

Manuscript received June 14, 1994; revised January 26, 1996. This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC). The material in this work was presented in part at the 1993 Conference on Information Sciences and Systems, The Johns Hopkins University, Baltimore, MD, and in part at the 1993 Canadian Workshop on Information Theory, Rockland, Ont, Canada..

The author is with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, Ont., N2L 3G1 Canada.

Publisher Item Identifier S 0018-9448(96)04014-X.

0018–9448/96$05.00 © 1996 IEEE

For the addressing/reconstruction, we need a one-to-one mapping between the elements of the selected subset of the N-D space and the set of the integer numbers $0, \cdots, T-1$ such that the mapping (addressing) and its inverse (reconstruction) can be easily implemented. The immediate approach to obtain such a mapping is to use a lookup table.

In a high-dimensional space, as the cardinality of the selected subset of points is usually quite high, one cannot use the immediate approaches based on the exhaustive search and/or the lookup table and an algorithmic approach is needed. The basic strategy is to use the high degree of structure of the problem in (1) to reduce the complexity of the involved operations. This structure is due to the fact that both the objective function and the constraint in (1) are composed of the sum of some terms corresponding to the 1-D subspaces. This results in a recursive structure for the resulting quantizer which is explained in the following.

III. Recursive Merging of Shells: Hierarchical Dynamic Programming

Dynamic programming is a multistage optimization procedure based on an inductive principle. It makes use of a recursive relationship to decompose a complicated problem into a sequence of easier subproblems. In the following, we introduce our approach to dynamic programming. As the schemes of [4]–[6] are also based on dynamic programming, we have focused our explanation on a comparison between the methods.

The core idea in the schemes of [4]–[6] is to use a state diagram with the transitions corresponding to the 1-D symbols. This results in a trellis composed of N stages where N is the space dimensionality. The states s and $s+c$ in two successive stages are connected by a link corresponding to the 1-D symbol(s) of cost c. Consequently, the states in the nth stage, $n = 0, \cdots, N-1$, represent the accumulative cost over the set of the first n dimensions. The links connecting two successive stages are labeled by the corresponding 1-D distortions. Then, the Viterbi algorithm is used to find the path of the minimum overall additive distortion through the trellis.

Unlike [4]–[6] which are based on a component-by-component analysis, we build our recursive structure in a hierarchy of levels where each level involves the Cartesian product of two lower dimensional subspaces. To explain this structure, let $F_N(C)$ denotes the set of the N-D points of the overall (additive) cost C (shell of cost C). We have the following *recursive* relationship:

$$F_N(C) = \cup\,[F_{N_1}(C_1) \otimes F_{N_2}(C_2)] \tag{2}$$

where $\otimes$ denotes the Cartesian product, $N = N_1 + N_2$, and the union is computed over all the pairs (C_1, C_2) satisfying $C_1 + C_2 = C$. We are specially interested in the case that $N_1 = N_2 = N/2$. We refer to each Cartesian product element in (2) as a *cluster.*

The shells of different costs in a given subspace are considered as the states of the system in that subspace. Using such states, we can concatenate two subspaces of dimensionality (N_1, N_2) together and obtain a space of dimension N. The key point is that in such a concatenation, by using (2), the states in an N-D space can be expressed as the union of the element in the Cartesian product of the states in the lower dimensional subspaces. This property enables us to continue with the concatenation of the subspaces in a hierarchical manner. This approach is specially effective when the space dimensionality is equal to $N = 2^u$, where u is an integer. In this case, the hierarchy is composed of u levels where the ith level, $i = 0, \cdots, u-1$, involves the pairwise Cartesian product of the 2^i-D subspaces (there are 2^{u-i} such pairs in the ith level). All our following discussions are based on this structure.

The immediate benefit of this approach is the possibility of using a parallel processing system. Another benefit is that this structure can be easily combined with the state diagram of a lattice (used to decode the lattice [10]). This provides a means to easily use the scheme in conjunction with a quantization lattice. More importantly, as we will see later, this approach provides the basis for an effective state-space quantization rule. In the following, we explain how this structure of states can be used to achieve the decoding and the addressing.

A. Recursive Decoding

For a given input vector $\boldsymbol{x}$, by decoding of a shell we mean the process of finding the element of the shell which has the minimum distance to $\boldsymbol{x}$. Using (2), we can decode a shell recursively. To do this, the N-D input vector $\boldsymbol{x}$ is split into two parts $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, each of length $N/2$. Assume that the nearest vectors of $F_{N/2}(C_1)$, $F_{N/2}(C_2)$ to $\boldsymbol{x}_1$, $\boldsymbol{x}_2$ are equal to $\hat{\boldsymbol{x}}_1$, $\hat{\boldsymbol{x}}_2$ with the distortions d_1, d_2, respectively. The nearest vector of the cluster $F_{N/2}(C_1) \otimes F_{N/2}(C_2)$ to $\boldsymbol{x}$ is equal to $(\hat{\boldsymbol{x}}_1, \hat{\boldsymbol{x}}_2)$ with the distortion $d_1 + d_2$. The distortion of a shell is equal to the smallest of the distortions of its clusters. Note that if we know the distortion and the nearest vector for all the shells of the $N/2$-D subspaces, we can decode all the N-D shells. This means that by starting from the 1-D subspaces and progressing in a recursive way, one is able to decode an $N = 2^u$ dimensional space in u steps.

B. Recursive Addressing

One can also use the recursive structure of the shells to develop algorithmic addressing, reconstruction procedures. The basic idea is that the addressing within each cluster can be achieved independently along its lower dimensional shells. This results in the same decomposition principle as proposed for the first time in [7] and elaborated in [8], [9]. To complete the recursion, it remains to select a single cluster within a shell. This is achieved by arranging the clusters within a shell in a preselected order and assigning a larger label to the points in a higher order cluster. Based on this ordering, a cluster is selected according to the range of the index and the corresponding residue with respect to the start of the range is used for the addressing within the cluster.

The procedure of recursive addressing becomes specially simple if all the cardinalities are restricted to an integral power of two. The key point behind the simplicity is as follows: Consider two sets of cardinalities 2^{c_1} and 2^{c_2}. The Cartesian product of these sets is composed of $2^{c_1+c_2}$ elements. To address an element of this Cartesian product, the input bit stream composed of $c_1 + c_2$ bits is simply split into two parts of lengths c_1 and c_2. Each part is subsequently used to select a point within one of the two sets. In other words, the address of a composite symbol is easily obtained by concatenating the addresses of its constituents lower dimensional components.

The reconstruction process is achieved by reversing the order of the steps taken for the addressing.

IV. State-Space Quantization: Aggregation of States

The straightforward approach is to assign an independent state to each possible value of cost at a given level. Let K denote the number of the distinct values of cost along a dimension. In this case, the number of distinct values of cost in N dimensions can be as large as

$$D = \sum_{\substack{\sum_{i=0}^{K-1} n_i = N}} \frac{N!}{\prod_i n_i!}. \tag{3}$$

The general term in (3) represents the total number of N-tuples where the 1-D symbol with the ith value of cost has occurred for n_i times. If two different combinations in (3) result in the same value for the additive cost, the corresponding states *merge* together. This is denoted

as a *natural merge*. Even for a moderate value of K, the number of distinct states in N-D can be impractically large. The solution is to synthetically *aggregate* distinct states into a smaller number. This is denoted as the *state-space quantization* and is the key point to the effectiveness of any dynamic programming approach. In [4], [5], the self-information associated with the 1-D symbols are rounded off to rational numbers with a common denominator. In [6], to reduce the complexity with respect to [4], [5], these are rounded off to integer numbers.

The major question is how we can aggregate the shells into *macro-shells* while keeping the degradation in performance negligible. Obviously, after aggregation, the points of the macro-shells will no more be of exactly the same cost. Based on our hierarchy in an $N = 2^u$-D space, we consider the following rules for the aggregation of states.

Recursive Aggregation Rule: The macro-shells in 2^i-D subspaces are composed of the union of the elements in the Cartesian product of the 2^{i-1}-D macro-shells. In this case, the macro-shells act as the states of the system and the same recursive addressing and decoding methods explained earlier are applicable.

In devising a specific merging rule, we should keep the following three implicit objectives in mind:

1) As truncation is achieved by discarding some of the macro-shells, while the objective is to discard a given number of points of the highest cost, we should try to minimize the overlap between the range of the costs of different macro-shells.
2) The number of the macro-shells should be as small as possible. This suggests that we should try to put an equal number of points in different macro-shells. As we will see later, in the case that the macro-shells have an equal number of points, the addressing is also much simpler than the general case.
3) Aggregation rule should be compatible with our recursive structure mentioned earlier.

Concerning the first objective of this list, the best approach is to partition the dynamic range of the cost into nonoverlapping segments. Then, each macro-shell is considered as the set of elements with the cost in one of these subranges. By appropriately selecting the subranges, one can even put an equal number of points in each macro-shell and satisfy the second objective. This sounds excellent, however, unfortunately, no recursive structure is known for this type of aggregation. As we will see later, by partitioning the space into macro-shells of increasing *average cost*, it is possible to remain compatible with our recursive structure.

In the following, we propose two rules for the state-space quantization which partially fulfill the aforementioned objectives. In the first method, the aggregation is limited to the 1-D subspaces. This is based on a similar principle as used in the context of constellation shaping in [11]. In the second method this is achieved sequentially in different levels of our hierarchy. This is based on a similar principles as used in the context of the constellation shaping in [8], [12], and [13]. As we will see later, the second method is specially effective and results in a simple addressing procedure.

A. Aggregation On a 1-D Basis, Macro-Shells of Identical Sum of the Indices

The effect of natural merging of shells is specially pronounced when the costs of the 1-D shells are affine functions of their indices (cost of the ith shell is equal to $c_0 + i\Delta$). This results in a set of $1 + N(K-1)$ distinct shells in an N-D space where K is the number of 1-D shells. Based on this observation, in our first method, the 1-D symbols are aggregated into K information macro-shells with a fixed spacing (increment in the self-information) Δ. In this case, the probabilities of the points in the ith 1-D macro-shell satisfy $0 < -\log_2 p \le c_0$ for $i = 0$ and $c_0 + (i-1)\Delta < -\log_2 p \le c_0 + i\Delta$, for $i = 1, \cdots, K-1$. Obviously, some of the 1-D macro-shells may remain empty. The higher dimensional macro-shells are considered as the set of the symbols with a fixed sum of the indices. This results in a recursive structure. The final subset is selected as the union of the N-D macro-shells with the sum of the indices less than a given value $L_{\max}$. This results in $\min[2^i K, L_{\max}]$ states in the ith level of the hierarchy. This approximation method can be considered as a more general formulation for the schemes of [4]–[6] which are based on approximating the costs on a 1-D basis.

From the three objectives listed earlier, this method just fulfills the last one, namely, the recursive structure. In the following, we introduce another method which is more compatible with these three objectives.

B. Aggregation on a Sequential Basis, Macro-Shells of Increasing Average Costs and Identical Cardinalities

In our second method, the quantization of the state space is based on a sequential aggregation of the macro-shells in the 2^i-D subspaces, $i = 0, \cdots, u-1$. In other words, the state-space quantization is achieved gradually at different levels of the hierarchy. The subspaces involved at each level are partitioned into a number of macro-shells of increasing average costs and identical cardinalities. The key point is to approximate the costs of all the points within a given macro-shell by their average value.

Consider an $N = 2^u$-dimensional space and assume that there are $K_i = 2^{k_i}$ macro-shells of equal cardinality in the $N_i = 2^i$-D subspaces, $i = 0, \cdots, u-1$. In the Cartesian product of two of the N_i-D subspaces, we obtain 2^{2k_i} clusters of equal cardinality. The clusters are arranged in the order of increasing average costs. A number equal to $2^{2k_i - k_{i+1}}$ of subsequent clusters are aggregated into a higher level ($2N_i = N_{i+1}$-D) macro-shell. Then, the whole process is repeated recursively. The final subset is obtained by keeping some of the N-D clusters of the lowest average cost. Note that this whole operation is done just once and the result is stored for subsequent uses. In this case, the total number of states (macro-shells) in the ith level of the hierarchy is equal to $(N/2^i) \times 2^{k_i} = 2^{u+k_i-i}$ where $N/2^i$ is the number of 2^i-dimensional subspaces involved in the ith level.

Using macro-shells of *integral, equal bit rate* results in a specially simple addressing scheme. This is discussed in the following: Consider the case that the macro-shells in a given level of our hierarchy, say at dimensionality N', are composed of 2^{c_1} elements. Also, assume that a higher level macro-shell (dimensionality $2N'$) is obtained by aggregating 2^{c_2} clusters in the two-fold Cartesian product of the N'-D macro-shells. The addressing of the $2N'$-D macro-shells requires $2c_1 + c_2$ bits. The address of an $2N'$-D element is computed by concatenating the addresses of its constituent components in the N'-D macro-shells and concatenating the result with an additional c_2 bits which are selected as the label of the cluster within its corresponding $2N'$-D macro-shell.

1) Computation of Complexity: The number of additions–comparisons[1] required to decode each pair of the subspaces involved in the ith level is equal to 2^{2k_i} for $i = 0, \cdots, u-2$ and $2^{2k_{u-1} - r_s}$ for $i = u-1$, where r_s denotes the redundancy associated with the selection of the final N-fold symbols as a subset of the Cartesian product space. Note that r_s is equal to the logarithm of the ratio of the employed number of points per dimension to the

[1] By addition–comparison we mean the combination of one addition and one comparison. Note that in dynamic programming, these two operations always occur together.

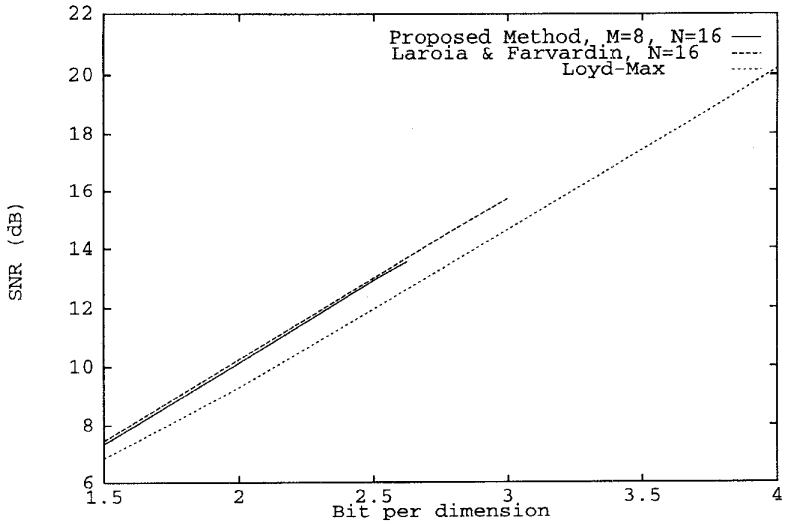

Fig. 1. Quantization SNR for an i.i.d. Gaussian source. $N = 16$ (dimensionality), $M = 8$ (number of points per dimension).

minimum necessary number of points per dimension. Multiplying by the number of pairs in each level, namely, 2^{u-i-1} for levels $i = 0, \cdots, u-1$, we obtain the following expression for the total number of additions–comparisons:

$$A_{\text{Total}} = 2^{2k_{u-1}-r_s} + \sum_{i=0}^{u-2} 2^{2k_i+u-i-1}. \tag{4}$$

In the case of using parallel processing, all the pairs in a given level are computed simultaneously and the factor 2^{u-i-1} will not be present in the general term of the summation in (4).

For addressing in an $N = 2^u$-dimensional space, all we need is a set of u blocks of ROM (read-only memory) to store the components of each macro-shell in the Cartesian product of the macro-shells in its lower dimensional subspaces. The ith addressing level, $i = 0, \cdots, u-2$, requires a lookup table with $2k_i \times 2^{2k_i}$ bits. The last level requires $2k_{u-1} \times 2^{2k_{u-1}-r_s}$ bits. The total size is equal to

$$M_{\text{ROM}} = (k_{u-1} \times 2^{2k_{u-1}-r_s+1}) + \sum_{i=0}^{u-2} k_i \times 2^{2k_i+1}. \tag{5}$$

We also need a block of RAM (random-access memory) to store the result of the computations. These results are: i) cost of the survivor corresponding to each state at levels $i = 0, \cdots, u-1$, and ii) 1-D components corresponding to each such survivor. We need k_0 bits to represent each 1-D macro-shell. Recall that there are 2^{u+k_i-i} states (macro-shells) in the ith level of the hierarchy. We need to store 2^i 1-D components for the survivor corresponding to each state at level $i = 0, \cdots, u-1$. This results in a block of RAM of size

$$M_i = k_0 \times 2^{u+k_i} \tag{6}$$

bits at level $i = 0, \cdots, u-1$.

Next, we compute the size of the memory required to store the survivors values. We assume that each 1-D cost is represented by R_0 bits. Noting that the survivor values at a given level are obtained by adding two survivor values from a lower level, we conclude that one needs $R_0 + i$ bits to represent each of the survivor values at levels $i = 1, \cdots, u-1$. Multiplying by the number of states, we conclude that the total memory size required to store the survivor values at level i is equal to

$$M_i' = 2^{u+k_i-i} \times (R_0 + i). \tag{7}$$

In practice, to decode the level $u-1$, it is enough to have full storage for one of the two subspaces in level $u-1$ (denoted as the right subspace) and for the two subspaces in level $u-2$ building the other subspace in level $u-1$ (denoted as the left subspace). This is based on immediately moving to the final level of the decoding after the computations over a given macro-shell in the left subspace at level $u-1$ is completed (assuming that the macro-shells in the left subspace at level $u-1$ are decoded one by one).

Obviously, we do not need to assign a separate block of RAM for each level and the same block can be used several times as we go higher in the hierarchy. Our computation shows that for the values of k_i's and u of interest to us, if one has enough RAM for the right subspace at level $u-1$, and for the two subspaces at level $u-2$ building the left subspace at level $u-1$, then that memory is enough to carry out all the computations. This results in a total size of

$$M_{\text{RAM}} = \tfrac{1}{2}(M_{u-1} + M_{u-1}' + M_{u-2} + M_{u-2}'). \tag{8}$$

2) Comparison with Other Methods: First, we talk about the selection of the parameters, namely, M and k_i's, $i = 0, \cdots, u-1$. In general, to keep the complexity and/or the performance at reasonable levels, there are not too many choices available for these parameters and one can simply select them using the trial and error method. The value of M should be selected to support the required bit rate per dimension, plus the selected value for the redundancy per dimension. A larger value for M results in a better performance but at the same time increases the complexity. We always select the value of M to obtain 0.5 bit of redundancy per dimension. Our experience shows that this selection results in a reasonable compromise between the complexity and the performance.

In the selection of k_i's, our experience shows that the following rules of thumb result in a good tradeoff between performance and complexity: i) The value of k_0 is selected such that there are two points in each 1-D macro-shell.[2] ii) The values of k_i, $i = 1, \cdots, u-2$ are selected to be close to each other. iii) The value of k_{u-1} is selected larger compared to the rest of k_i's. Note that discarding of clusters is achieved at level $u-1$, and consequently, having a larger value for k_{u-1} results in a noticeable improvement

[2]Due to symmetry, the 1-D partitions of the original scalar quantizer have in pair (positive and negative points) the same value for the cost. Such two points are aggregated in one macro-shell. Note that this aggregation does not result in any approximation in the first level.

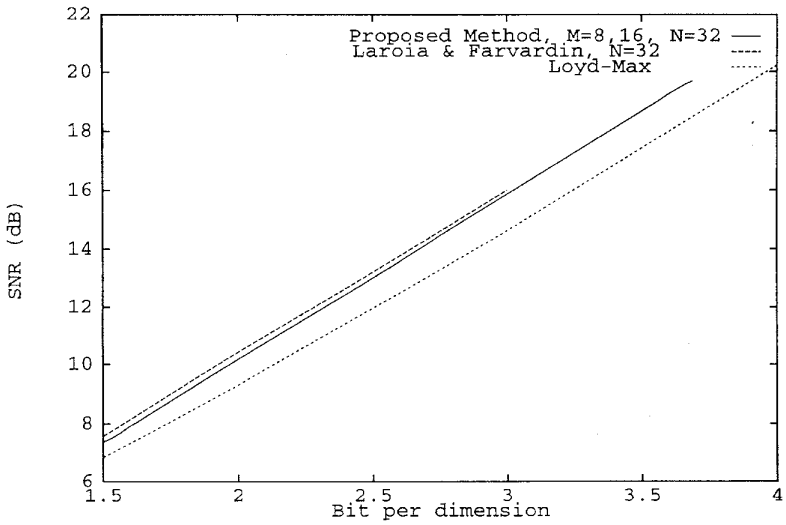

Fig. 2. Quantization SNR for a Gaussian source. $N = 32$ (dimensionality), $M = 8, 16$ (number of points per dimension).

TABLE I
COMPARISON BETWEEN THE PROPOSED METHOD BASED ON THE SEQUENTIAL AGGREGATION OF SHELLS (DENOTED BY SMS) WITH THE SCHEME OF [4], [5] (DENOTED BY L-F)
(The quantities N, M, and R are the space dimensionality, the number of points per dimension, and the rate (in bits) per dimension, respectively. The memory size is in byte (8 bits) per N dimensions and the computational complexity is the number of additions/comparisons per dimension. The value of R_0 in (7) is selected as $R_0 = 8$ bits. The values inside parentheses are the computational complexities of our method in the case of using a parallel processing system.)

Method	N	M	$(k_i, i = 0, u-1)$	R	M_{RAM}	M_{ROM}	$M_{\text{RAM}} + M_{\text{ROM}}$	Computation	SNR (dB)
SMS	16	4	(1, 2, 4, 8)	1.5	0.6	0.8	1.4 k	50 (30)	7.43
L-F	16	4	————	1.5	——	——	8 k	6×10^2	7.47
SMS	16	8	(2, 4, 5, 8)	2.5	1.0	2.0	3.0 k	220 (100)	12.91
L-F	16	8	————	2.5	——	——	21 k	2×10^3	13.00
SMS	32	16	(3, 5, 6, 6, 10)	3.5	8	13	21 k	1100 (300)	18.7
L-F	32	16	————	3.5	——	——	300 k	1×10^4	18.8†

in performance while the corresponding increase in complexity is moderate (due to the subtraction of r_s from the exponent of the corresponding terms in (4) and (5)).

In the following, we present some numerical results concerning the performance and the complexity of the proposed method. In all cases, a sequence of 20 000 source vectors is used to design the quantizer and a separate sequence of the same length is used to measure the resulting performance.

Figs. 1 and 2 show the SNR (signal-to-noise ratio) obtained by using our sequential aggregation rule in conjunction with an independent and identically distributed (i.i.d.) Gaussian source. The quantization distortion is measured in terms of the mean-square distance. Table I presents a comparison between our method and the scheme of Laroia and Farvardin [4], [5] in terms of performance and complexity. The complexity of the method of Laroia and Farvardin is computed using the approximate formulas given in [4], [5]. It is difficult to have a fair comparison with the scheme of [6] because in their case the space dimensionality is usually quite high which results in a longer delay.

V. SUMMARY

We have presented an efficient method for the fixed-rate entropy coding of a memoryless source using dynamic programming. We build our recursive structure required for the dynamic programming in a hierarchy of levels. This results in several benefits over the conventional trellis-based approaches. Using this structure, we have developed efficient rules (based on aggregating the states) to substantially reduce the search/addressing complexities while keeping the degradation in performance negligible.

REFERENCES

[1] M. V. Eyuboglu and G. D. Forney, "Lattice and trellis quantization with lattice- and trellis-bounded codebooks—High-rate theory for memoryless sources," *IEEE Trans. Inform. Theory*, vol. 39, pp. 46–59, Jan. 1993.

[2] D. J. Sakrison, "A geometrical treatment of the source encoding of a Gaussian random variable," *IEEE Trans. Inform. Theory*, vol. IT-14, pp. 481–486, May 1968.

[3] T. R. Fischer, "Geometric source coding and vector quantization," *IEEE Trans. Inform. Theory*, vol. 35, pp. 137–145, Jan. 1989.

[4] R. Laroia and N. Farvardin, "A structured fixed-rate vector quantizer derived from variable-length encoded scalar quantizers," in *24th Annual Conf. on Information Sciences and Systems* (Princeton, NJ, Mar. 1990), pp. 796–801.

[5] ——, "A structured fixed-rate vector quantizer derived from variable-length scalar quantizer—Part I: Memoryless sources," *IEEE Trans. Inform. Theory*, vol. 39, pp. 851–867, May 1993.

[6] A. S. Balamesh and D. L. Neuhoff, "Block-constrained methods of fixed-rate, entropy coded, scalar quantization," submitted to *IEEE Trans.*

Inform. Theory, Sept. 1992.
[7] G. R. Lang and F. M. Longstaff, "A leech lattice modem," *IEEE J. Sel. Areas Commun.*, vol. 7, pp. 968–973, Aug. 1989.
[8] A. K. Khandani and P. Kabal, "Shaping multi-dimensional signal spaces—Part II: Shell-addressed constellations," *IEEE Trans. Inform. Theory*, vol. 39, pp. 1799–1808, Nov. 1993.
[9] G. D. Forney, "Coset codes—Part II: Binary lattices and related codes," *IEEE Trans. Inform. Theory*, vol. 34, pp. 1152–1187, Sept. 1988.
[10] A. R. Calderbank and L. H. Ozarow, "Nonequiprobable signaling on the Gaussian channel," *IEEE Trans. Inform. Theory*, vol. 36, pp. 726–740, July 1990.
[11] R. Laroia, N. Farvardin, and S. A. Tretter, "On optimal shaping of multi-dimensional constellations," *IEEE Trans. Inform. Theory*, vol. 40, pp. 1044–1056, July 1994.
[12] A. K. Khandani and P. Kabal, "Shaping of multi-dimensional signal constellations using a lookup table," *IEEE Trans. Inform. Theory*, vol. 40, pp. 2058–2062, Nov. 1994.

Vanishing Distortion and Shrinking Cells

Andrew B. Nobel, *Member, IEEE*

***Abstract*—We establish an asymptotic connection between vanishing rth-power distortion and shrinking cell diameters for vector quantizers with convex cells.**

***Index Terms*—Asymptotic quantization theory, high-rate quantization theory, vector quantization, shrinking cell diameters, convex sets.**

I. Introduction

The study of high-rate vector quantization is concerned with the performance of quantizers having large codebooks. One seeks bounds (or precise estimates) for the asymptotic distortion of a sequence vector quantizers in terms of codebook size, vector dimension, and the nature of the selected distortion measure. Gersho [4], Yamada *et al.* [14], and others have given heuristic derivations of formulas governing the distortion of vector quantizers with large codebooks. They assume that the underlying distribution has a smooth density, and that the cell diameters of the nth quantizer tend to zero as n tends to infinity. The latter condition, stipulating shrinking cells, is the subject of this correspondence.

Recently, several authors [5], [6], [10]–[13], have proposed using vector quantizers as the basis for multivariate histogram classification and regression schemes in higher dimensions. Verification of shrinking cell conditions is typically the key to establishing the consistency of such schemes (cf. [6], [10]). Although they do not figure explicitly in the rigorous derivation [2], [3], [15], [16] of bounds concerning the distortion of optimal nearest neighbor quantizers, shrinking cell conditions do appear in more general settings. Na and Neuhoff [7] require shrinking cells in their derivation of Bennett's integral for vector quantizers having convergent point densities and convergent inertial profiles. The same condition appears in recent work [8] on the asymptotic distribution of errors for high-rate vector quantizers.

The rth-power distortion of a quantizer is an average quantity, while its cell diameters measure its worst case local behavior. In many cases, the connection between quantizer design (which is typically distortion-based) and cell size is not readily apparent. For example, verifying a shrinking cell condition can be problematic when the quantizers under study are designed from finite data sets using iterative or recursive methods that seek to reduce empirical distortion, or when they are defined in terms of secondary quantities such as point densities and inertial profiles.

In Theorem 1 below we establish an asymptotic connection between vanishing rth-power distortion and shrinking cell diameters for quantizers with convex cells. As a consequence, a number of shrinking cell conditions may be easily verified by showing that the quantizers in question have distortion tending to zero. Theorem 1 also plays an important role in the asymptotic analysis [9] of a common greedy growing scheme for tree-structured vector quantizers.

II. Results

A vector quantizer is a mapping $Q\colon \mathbb{R}^d \to \mathcal{C}$, where $\mathbb{R}^d$ denotes d-dimensional Euclidean space, and $\mathcal{C} = \{c_1, \cdots, c_m\} \subseteq \mathbb{R}^d$ is a finite set of representative vectors known as the *codebook* of Q. Let P be a fixed probability distribution on $\mathbb{R}^d$. For each $r > 0$, the rth-power distortion of Q with respect to a random vector $X \sim P$ is given by

$$D_r(Q) = E\|Q(X) - X\|^r = \int \|Q(x) - x\|^r dP(x) \qquad (1)$$

where $\|\cdot\|$ is the ordinary Euclidean norm on $\mathbb{R}^d$. A sequence of quantizers $Q_1, Q_2, \cdots$ has vanishing rth-power distortion if $D_r(Q_n) \to 0$ as $n \to \infty$.

Every quantizer Q is associated with a finite partition $\{A_1, \cdots, A_m\}$ of $\mathbb{R}^d$, where $A_i = \{x\colon Q(x) = c_i\}$ is the cell contains those vectors assigned to the ith codeword. For each vector x the cell of Q containing x is defined by

$$Q[x] = \{u\colon Q(u) = Q(x)\}.$$

The *diameter* of a set $U \subseteq \mathbb{R}^d$ is the greatest distance between any two points of the set, namely

$$\operatorname{diam}(U) = \sup_{u,v \in U} \|u - v\|.$$

A sequence of quantizers $Q_1, Q_2, \cdots$ will be said to have *shrinking cells* if for every $\epsilon > 0$

$$P\{x\colon \operatorname{diam}(Q_n[x]) > \epsilon\} \to 0. \qquad (2)$$

Equivalently, $\operatorname{diam}(Q_n[X]) \to 0$ in probability when $X \sim P$. Note that $\operatorname{diam}(Q_n[x])$ accounts for all the points in the cell, not just those that lie in the support set of P.

Let $Q_1, Q_2, \cdots$ be vector quantizers such that i) each cell A_i of Q_n contains its corresponding codeword c_i, and ii) c_i is no worse a representative than the zero vector in the sense that

$$\int_{A_i} \|x - c_i\|^r dP \le \int_{A_i} \|x\|^r dP.$$

When $r = 2$ both conditions are satisfied if Q_n has convex cells and the representative of each cell is its centroid with respect to P. Under these conditions, it is readily verified that shrinking cells imply vanishing distortion.

Manuscript received July 10, 1995; revised December 18, 1995. This work was completed while the author was a Beckman Institute Fellow at the Beckman Institute for Advanced Science and Technology, University of Illinois, Urbana-Champaign. This work was supported in part by the NSF under Grant DMS-9501926.

The author is with the Department of Statistics, University of North Carolina, Chapel Hill, NC 27599-3260 USA.

Publisher Item Identifier S 0018-9448(96)03745-5.

0018–9448/96$05.00 © 1996 IEEE