Efficient Algorithms for Fixed-Rate Entropy-Coded Vector Quantization

A. K. Khandani*, P. Kabal†,‡ and E. Dubois†
*Dept. of Elec. and Comp. Eng., University of Waterloo, Waterloo, Ont., N2L 3G1
†INRS–Telecommunications, 16 Place du Commerce, Verdun, PQ., H3E 1H6
‡McGill University, 3480 University, Montreal, PQ., H3A 2A7

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 presents us with the problem of the variable data rate. A solution in a space of dimensionality N is to select a subset of elements in the N-fold cartesian product of a scalar quantizer and represent them with code-words of the same length. 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. The *search/addressing* of this scheme can no longer be achieved independently along the one-dimensional subspaces. In the case of a memoryless source, the selected subset has a high degree of structure which can be used to substantially decrease the 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 stages. 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 negligible.

1 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 straight-forward 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 a subset of the N-fold symbols and represent them with code-words 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 output and also the corresponding *addressing/reconstruction* processes can no longer be achieved independently along the one-dimensional (one-D) subspaces. The basic idea is to select the subset of points in such a way that these processes can be simplified.

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.

Another class of schemes are 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 symmetry which can be used to substantially reduce the complexity. A method for reducing the complexity of such a quantizer based on using a state diagram (with the states corresponding to the length of the code-words) is introduced by Laroia and Farvardin in [4]. Subsequently, Balamesh and Neuhoff in [5], introduce some complementary techniques to further reduce the complexity. In the present work, we introduce some more advanced techniques showing improvement with respect to the schemes of [4], [5].

We discuss a dynamic programming approach. The key point is to use the additive property of the self-information, in conjunction with the additive property of the distortion measure, to decompose the search/addressing 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 stages where each stage involves the cartesian product of two lower dimensional subspaces. This results in several benefits over the conventional trellis-based approach used in [4], [5]. By effectively quantizing the state space, we obtain suboptimum methods with low complexity and negligible performance degradation.

2 Basic Structure

Consider a memoryless source and a scalar quantizer composed of M partitions. In the N-fold cartesian product of this quantizer, we obtain M^N, N-D partitions. The final vector quantizer is selected as a subset of the N-D partitions composed of T elements. Each partition is represented by a code-word composed of $\lceil \log_2 T \rceil$ bits. The N-D reconstruc-

0-7803-1825-0/94 $4.00 © 1994 IEEE

tion vectors are denoted as $\mathbf{r}_i, i = 0, \ldots, T-1$. For a given source vector $\mathbf{x}$, the quantization rule (decoding) is to find the reconstruction vector $\mathbf{r}_i$ which has the minimum square distance to $\mathbf{x}$, addressing is to produce the index i when $\mathbf{r}_i$ is selected, and reconstruction is to reproduce $\mathbf{r}_i$ from the index i.

Assume that the induced self-information and the expected value of the symbols mapped to the j'th one-D partition are equal to c_j and r_j, respectively. The self-information associated with a one-D point is considered as a cost associated with that point. The selection rule for the N-D symbols is to keep the points with the lowest overall additive cost. The N-D reconstruction vectors are obtained by concatenating the corresponding one-D reconstruction levels, namely r_j's. The search operation is formulated as:

$$\begin{array}{ll} \text{Minimize} & \sum_{i=0}^{N-1}(x_i - r_{j_i})^2 \\ \text{Subject to:} & \sum_{i=0}^{N-1} c_{j_i} \leq C_{\max} \end{array} \qquad (1)$$

The immediate approach to solving (1) is to perform an exhaustive search.

For the addressing/reconstruction, we need a one-to-one mapping between the set of the code-words and the set of the integer numbers $0, \ldots, T-1$ 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 look-up table.

In a high dimensional space, as the number of the symbols is usually quite high, one can not make use of the immediate approaches based on exhaustive search and lookup table. The main idea is to use the high degree of structure, which is mainly due to the symmetry of the problem in (1), to reduce the complexity of the involved operations.

3 Recursive merging of shells

If $F_N(c)$ denotes the set of N-D points of cost c (shell of cost c), we have the following recursive relationship:

$$F_N(c) = \bigcup [F_{N_1}(c_1) \otimes F_{N_2}(c_2)] \qquad (2)$$

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 refer to each cartesian product element in (2) as a *cluster*. We are specially interested in the case that $N_1 = N_2 = N/2$.

For a given input vector $\mathbf{x}$, by decoding of a shell we mean the process of finding the element of the shell which has the minimum distance to $\mathbf{x}$. Using (2), we can decode a shell recursively. To do this, $\mathbf{x}$ is split into two parts $\mathbf{x}_1$ and $\mathbf{x}_2$ of lengths N_1 and N_2. Assume that the nearest vectors of $F_{N_1}(c_1)/F_{N_2}(c_2)$ to $\mathbf{x}_1/\mathbf{x}_2$ are equal to $\hat{\mathbf{x}}_1/\hat{\mathbf{x}}_2$ with the minimum distances d_1/d_2. The nearest vector of $F_{N_1}(c_1) \otimes F_{N_2}(c_2)$ to $\mathbf{x}$ is equal to $(\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2)$ with the minimum distance $d_1 + d_2$. The minimum distance of a shell is equal to the smallest of the minimum distances of its clusters.

For $N_1 = N_2 = N/2$, if we know the minimum distance and the nearest vector for all the shells of the $N/2$-D subspaces, we can decode all the N-D shells. The lower is the number of shells in the $N/2$-D subspaces, the simpler will be the decoding process.

One can also use the recursive structure of the shells to develop an 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 [6] and elaborated in [7], [8]. 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 assuming that the points in a higher order cluster have a larger label. 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 be an integral power of two (integral bit rate). The key point behind the simplicity is as follows: Consider two sets of cardinalities 2^{t_1} and 2^{t_2}. The cartesian product of these sets is composed of $2^{t_1+t_2}$ elements. To address an element of the cartesian product, the input bit stream composed of $t_1 + t_2$ bits is split into two parts of lengths t_1 and t_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 can be easily obtained by concatenating the addresses of its constituents lower dimensional elements. The effect of merging is reflected through some additional bits which are stored in a block of memory.

4 Hierarchical dynamic programming

Dynamic programming is a multi-stage 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] and [5] are also based on a dynamic programming, we have focused our explanation on a comparison between the methods.

The core of the idea in the schemes of [4], [5] is to use a state diagram with the transitions corresponding to one-D symbols. This results in a trellis composed of N stages. The states s and $s + c$ in two successive stages are connected by a link corresponding to the one-D symbol(s) of cost c. Consequently, the states in the nth stage, $n = 1, \ldots, N$, represent the accumulative cost over the set of the first n dimensions. The links connecting two successive stages are labeled by the corresponding distortions. Then, the viterbi algorithm is used to find the path of the minimum overall distortion through the trellis.

The straight-forward approach is to assign an independent

state to each possible value of cost at a given stage. Let K denote the number of the distinct values of cost along a dimension. Number of distinct values of cost in N dimensions can be as large as:

$$C_N = \sum_{\sum_{i=0}^{K-1} n_i = N} \frac{N!}{\prod_i n_i!} \quad (3)$$

The general term in (3) represents the total number of N-tuples where the one-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 of 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 (after the natural merge) 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], the self-information associated with the one-D symbols are rounded to rational numbers with a common denominator. In [5], to reduce the complexity with respect to [4], these are rounded to integer numbers.

Unlike [4] and [5] which are based on a component-by-component analysis, we use a hierarchy of stages where each stage involves the cartesian product of lower-dimensional subspaces. This approach is specially effective when the space dimensionality is equal to $N = 2^u$. In this case, the hierarchy is composed of u stages where the ith stage, $i = 0, \ldots, u-1$, is based on the (pair-wise) cartesian product of the 2^i-D subspaces (there are 2^{u-i} identical pairs of cartesian product in the ith stage). 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 [9]). 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.

4.1 State-space quantization, aggregation of states

As already mentioned, a straight-forward approach results in a large number of distinct states (shells). The major question is how we can aggregate the shells into *macro-shells* while keeping the degradation negligible. Obviously, after aggregation, the points of the macro-shells are no more of the same cost (each macro-shell has a range of costs). Based on our hierarchy in an $N = 2^u$-D space, we consider the following recursive structure.

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 devising a specific 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 afore-mentioned objectives. In the first method, the aggregation is limited to the one-D subspaces. This is based on a similar approach as used for the first time in the context of constellation shaping in [10]. In the second method this is achieved sequentially in different stages of our hierarchy. As we will see later, the second method is specially effective and results in a simple addressing procedure.

4.2 Aggregation on a one-D basis, Macro-shells of identical sum of the indices

The effect of natural merging of shells is specially pronounced when the cost of the one-D shells is an affine function of their indices (cost of the ith shell is equal to $c_0 + i\Delta$). This results in a set of KN distinct shells in an N-D space where K is the number of one-D shells.

Based on this observation, in our first method, the one-D symbols are aggregated into K information macro-shells with a fixed spacing (increment in the self-information) Δ. The probabilities of the points in the ith macro-shell satisfy $0 < -\log_2 p \leq c_0$ for $i = 0$ and $c_0 + (i-1)\Delta < -\log_2 p \leq c_0 + i\Delta$, for $i = 1, \ldots, K-1$. Obviously, some of the one-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 stage of the hierarchy. This approximation method can be considered as a more general formulation for the schemes of [4] and [5] which are based on approximating the costs on a one-D basis.

From the three objectives in the afore-mentioned list, this method just fulfills the last one, namely the recursive structure. In the following, we introduce another method which is more compatible with these objectives.

4.3 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, $i = 0, \ldots, u-1$. In other words, the state-space quantization is achieved gradually at different stages of the hierarchy. The subspaces involved at each stage of the hierarchy 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 in the $N_i = 2^i$-D subspaces, $i = 0, \ldots, u-1$. In the cartesian product of two of the N_i-D subspaces, we obtain K_i^2 clusters of equal volume. The clusters are arranged in the order of increasing average costs. A number equal to K_i^2/K_{i+1} of subsequent clusters are aggregated into a higher level ($2Ni = N_{i+1}$-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.

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 stage of our hierarchy, say at dimensionality N', are composed of 2^{t_1} elements. Also, assume that a higher level macro-shell (dimensionality $2N'$) is obtained by aggregating 2^{t_2} clusters in the two-fold cartesian product of the set of the N'-D macro-shells. The addressing of the $2N'$-D macro-shells requires $2t_1 + t_2$ 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 t_2 bits which are selected as the label of the corresponding cluster within the $2N'$-D macro-shell.

For addressing in an $N = 2^u$ dimensional space, all we need is a set of u memory blocks to store the components of each macro-shell in the cartesian product of the macro-shells of the lower dimensional subspaces. The ith addressing stage, $i = 0, \ldots, u-2$, requires a lookup table with $2k_i \times 2^{2k_i}$ bits. The last stage requires $2k_{u-1} \times 2^{2k_{u-1} - r_s}$ bits where r_s denotes the redundancy associated with the selection of the final N-fold symbols as a subset of the cartesian product space. This is defined as the logarithm of the ratio of the employed number of points per dimension to the minimum necessary number of points per dimension. By using a relatively small number of macro-shells in lower dimensional subspaces and imposing an appropriate constraint on r_s, one can provide a tradeoff between performance and complexity.

4.4 Comparison with other methods

Figures (1), (2) show the Signal-to-Noise-Ratio (SNR) obtained by using our sequential aggregation rule in conjunction with an independent identically distributed (iid) Gaussian source. Table (1) presents a comparison between our method and the scheme of [4] in terms of performance and complexity. It is difficult to have a fair comparison with the scheme of [5] because in their case the space dimensionality is usually quite high which results in a longer delay.

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. IT-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. IT-35, pp. 137–145, January 1989.

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

[5] 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.

[6] G. R. Lang and F. M. Longstaff, "A leech lattice modem," *IEEE J. Select. Areas Commun.,* vol. SAC-7, pp. 968–973, Aug. 1989.

[7] A. K. Khandani and P. Kabal, "Shaping multi-dimensional signal spaces—Part II: shell-addressed constellations," to appear in the *IEEE Trans. Inform. Theory*, Nov. 1993.

[8] A. K. Khandani and P. Kabal, "An efficient block-based addressing schemes for the nearly optimum shaping of multi-dimensional signal spaces," submitted to *IEEE Trans. Inform. Theory,* Aug. 1992, revised Oct. 1993.

[9] G. D. Forney, "Coset codes—Part II: Binary lattices and related codes," *IEEE Trans. Inform. Theory*, vol. IT-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. IT-36, pp. 726–740, July 1990.

Method	N	R	Memory	Computation	SNR (dB)
SMS	16	1.5	1.25 k	54 (33)	7.43
L-F	16	1.5	7.9 k	670	7.47
SMS	16	2.5	2.5 k	220 (97)	12.91
L-F	16	2.5	21.0 k	2240	13.00
SMS	32	3.5	14.3 k	1060 (290)	18.7
L-F	32	3.5	307 k	12500	18.8†

Table 1: Comparison between the method based on the sequential aggregation of shells (denoted by SMS) with the scheme of [4] (denoted by L-F). The memory size is in byte (8 bits) per N dimensions and the computational complexity is the number of additions/comparisons per dimension. The values inside parenthesis are the computational complexities of our method in the case of using a parallel processing system. (The value denoted by † is obtained using interpolation.).

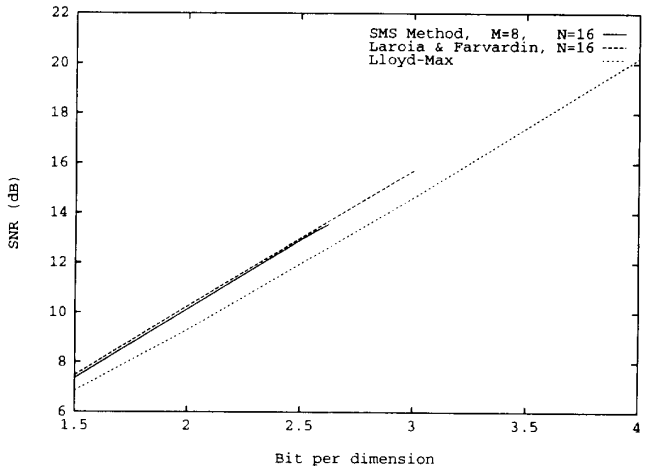

Figure 1: Quantization SNR for an iid Gaussian source, $N = 16$ (dimensionality), $M = 8$ (number of points per dimension) and $(k_1, k_2, k_3, k_4) = (3, 5, 6, 10)$.

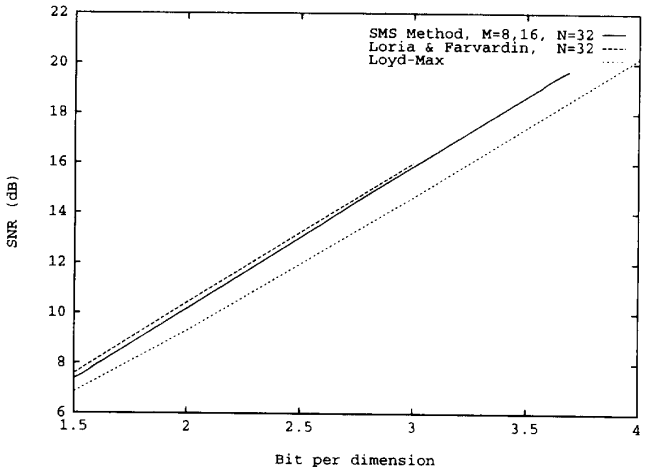

Figure 2: Quantization SNR for a Gaussian source, $N = 32$ (dimensionality), $M = 8, 16$ (number of points per dimension) and $(k_1, k_2, k_3, k_4, k_5) = (3, 5, 6, 7, 10), (4, 6, 7, 7, 10)$.