An Efficient Approach to Lattice-based Fixed-rate Entropy-coded Vector Quantization

S. Nikneshan and A. K. Khandani

Department of Electrical and Computer Engineering
University of Waterloo
Waterloo, Ontario, Canada, N2L 3G1

Technical Report UW-E&CE#2001-11

September 2, 2001

Abstract: In the absence of channel noise, variable-length quantizers perform better than fixed rate Lloyd-Max quantizers for sources with a non-uniform density function. However, channel errors can lead to a loss of synchronization resulting in a propagation of error. To avoid having variable rate, one can use a vector quantizer selected as a subset of high probability points in the Cartesian product of a set of scalar quantizers and represent its elements with binary code-words of the same length (quantizer shaping). We choose these elements from a lattice resulting in a higher quantization gain in comparison with simply using the Cartesian product of a set of scalar quantizers. We introduce a class of lattices which have a low encoding complexity, and at the same time result in a noticeable quantization gain. We combine the procedure of lattice encoding with that of quantizer shaping using hierarchical dynamic programming. In addition, by devising appropriate partitioning and merging rules, we obtain sub-optimum schemes of low complexity and small performance degradation. The proposed methods show a substantial improvement in performance and/or reduction in complexity with respect to the best known results.

1 Introduction

Optimum fixed-rate scalar quantizers, introduced by Max [2] and Lloyd [3] (LMQ), minimize the average distortion for a given number of threshold points. To increase the resolution of quantization in high probability region, the threshold points are closely spaced in those regions, and widely spaced where the probability is small. In spite of the gain of LMQ in comparison with uniform quantization, there is still a big gap left to the rate distortion bound.

To improve the performance of scalar quantizers, one could use variable-rate encoding of the quantizer output. Optimal variable-rate (entropy-constrained) scalar quantizers minimize the average distortion for a given output entropy and are known to asymptotically (at high rates) perform within 1.53 dB of the rate-distortion bound [4]. The performance of optimum entropy constrained quantizers is studied for a wide class of memoryless sources in [5]. A well known method for entropy-constrained quantization, introduced in [6], is based on using a distance measure which combines the effects of entropy and distortion by considering a linear combination of the two using fixed Lagrangian coefficients.

To take advantage of the entropy coding, while avoiding the dis-advantages associated with using variable rate codes (including error propagation and buffering problems), one can use fixed-rate entropy-coded vector quantization. This is based on selecting a subset of points of high probability in the Cartesian product of a set of scalar quantizers and representing its elements with binary code-words of the same length. This approach is traditionally denoted as the geometrical source coding [7]. In this case, one can further improve the quantization performance by exploiting a high dimensional lattice which offers some extra quantization (packing) gain due to using quantization regions which are close to a sphere.

One class of schemes are based on selecting the N-D quantizer points with the lowest additive self-information (typical set). In this case, the selected subset has a high degree of structure which can be used to reduce the complexity. A method for exploiting this structure based on using dynamic programming is proposed by Laroia and Farvardin in [8]. The core idea in the schemes of [8] is to use a state diagram with the transitions corresponding to the one-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 one-D symbol(s) of cost c. Consequently, the states in the nth stage, $n=0,\ldots,N-1$, represent the accumulative cost over the set of the first n dimensions. The links connecting two successive stages are assigned a metric corresponding to the one-D distortions. Then, the Viterbi algorithm is used to find the path of the minimum overall additive distortion through the trellis. In [8], cost is defined as a scaled version of self information where entropy coding is achieved by discarding the states with a cost higher than a given threshold. This Scalar-Vector Quantizer (SVQ) [8] is a vector quantizer derived from a variable-rate scalar quantizer and can achieve a large portion of the boundary gain by selecting the code-vectors inside the typical set.

Reference [9] uses a different approach to dynamic programming showing improvement with respect to the schemes of [8]. The key point in [9] is 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 which is formed in a hierarchy of levels where each level involves the Cartesian product of two lower dimensional subspaces.

Trellis Coded Quantizer (TCQ) introduced by Marcellin and Fischer in [10] is also derived from a scalar quantizer and uses Ungerboeck idea of coding by set partitioning to realize a significant packing gain. Reference [10] studies the quantization gain of several Ungerboeck one-dimensional trellis codes based on partitioning the integer lattice Z into the four cosets of $4Z$. In spite of the fact that no explicit attempt is made to realize the boundary gain, the trellis coded quantization performs remarkably well at rates of 3 bits/sample and less [10]. Trellis Coded Vector Quantizer (TCVQ) studied in [11] and [12] tries to realize the boundary gain by using the coding by set partitioning idea on a vector quantizer codebook. The Entropy-Constrained TCQ (ECTCQ) of Fischer and Wang [13] uses variable rate coding (using fixed Lagrangian [6]) in conjunction with a TCQ and performs within 0.5 dB of the rate distortion bound. In [14], Laroia and Farvardin combine the idea of SVQ [8] with TCQ [10] and propose a fixed-rate quantizer with packing which they call the Trellis-Based Scalar-Vector Quantizer (TB-SVQ).

In this work, we combine the hierarchical dynamic programming approach of [9] with lattice quantization and propose a new method for state quantization based on a binary tree. Reference [14] is the only known fixed-rate method with packing prior to the current work. We will later provide a comparison between our proposed scheme and [14] showing a substantial reduction in the complexity for a similar performance.

The article is organized as follows: Section 2 contains a brief description of the proposed Lattice quantizer. In Section 3, we talk about the basic structure of the proposed quantization scheme. Section 4 talks about the hierarchical dynamic programming for fixed-rate, entropy-coded quantization, and shows how it can be combined with encoding algorithm of the employed lattice structure. Section 5 talks about our approach to state quantization used to reduce the complexity. Finally, in Section 6, we compare the performance of the proposed method with some other relevant works reported in the literature.

2 Lattices Based on Hadamard Matrix

In the following, we present a lattice construction which is motivated by the ideas behind squaring construction [16]. We aim at generalizing the squaring construction to involve gluing of more than two identical sections with the objective of increasing the quantization gain without excessive increase in the complexity.

Define a set of vectors $V_L = \{v_1, v_2, \ldots, v_L\}$ such that v_i represents the ith row of an $L \times L$ Hadamard matrix. The vector $v_i^c \in V_L^c$ (complement of the vector $v_i \in V_L$) is obtained by replacing all the 1s in v_i by -1, and vice versa. Consider the set $W_L = V_L \cup V_L^c$ which is composed of $2L$ vectors in an L-D space. We define another set called $W_L^{(i)}$ whose members are obtained by concatenating the members of W_L for i times, where each vector is concatenated with itself or with its complement. The new set $W_L^{(i)}$ is composed of vectors of dimensionality $i \times L$, and has $2^i \times L$ members. For example (v_1, v_1^c, v_1, v_1^c) is a member of $W_L^{(4)}$ for $v_1 \in V_L$. Note that we can take advantage of Hadamard matrix structure and represent the set $W_L^{(i)}$ with a tree. As we will see later, this tree structure matches with the tree structure of hierarchical dynamic programming, facilitating the quantization operation. Note that the case $i = 2$ corresponds to squaring construction resulting in the Barnes-Wall lattices.

Assume that in the elements of $W_L^{(i)}$, all the 1s are substituted by $e(even)$ and all the -1s are substituted by $o(odd)$. The set $\Lambda(L, N = i \times L)$ is defined as the set of points in an N-D space obtained from $W_L^{(i)}$ by replacing all the e's with the set of the one-D threshold points with an even index, and all the o's with the set of the one-D threshold points with an odd index. Note that if we use an unbounded set of integer points along each dimension, the set Λ obtained in this manner will be a lattice. In occasions that no confusion can arise, we refer to Λ as a lattice. Note that $L = 8$ and $i = 1, 2$ correspond to Barnes-wall lattices E_8 and Λ_{16}, respectively.

We use an iterative training algorithm (similar to LBG) to adjust the reconstruction levels along each dimension. Fig. 1 shows the quantization gain (refer to [15] for definition) of these lattices (obtained through simulation) as a function of the space dimension for

$L = 4, 8, 16, 32$ (rank of the Hadamard matrix). In the following, we derive an expression for the computational requirements of the proposed Lattice Vector Quantizer (denoted as LVQ).

Suppose we use a $L = 2^l$-ranked Hadamard, the dimension is $N = 2^n$ and the number of threshold points along each dimension is M. The encoding can be implemented using a tree diagram which has $i = 1, 2, ..., n+1$ stages. At the first stage, we have to find the distance of each input sample to its nearest *even, odd* points. By replacing d_e^2 with $d_e^2 - d_o^2 = 2d \simeq d$, where d is the distance to the mid-point between the closets even and odd points, and d_o^2 with $d_o^2 - d_o^2 = 0$, we can avoid the squaring operation and also save the addition operation for the odd points. At three stages of hierarchy, we need comparison. First, at the beginning that the number of them is $\simeq N \log_2 M$ (for finding the nearest *even/odd* points). Second, at the l-th stage that the number of them is equal to $L \times \frac{N}{L}$ (for comparison between a given row of the Hadamard matrix and its complement). Finally, in the last stage, that the number of them is equal to L. Thus, the total number of comparisons is:

$$A_{Total} = (\log_2 M + 1)N + L \tag{1}$$

There are N subtraction required at the first stage of the hierarchy to calculate d (recall that d is the distance of input sample from the mid-point between the nearest odd and even points). In other stages, this number is $(2^i - 3) \times \frac{N}{2^{i-1}}$ where $1 < i \leq l+1$ (there are 3 additions with zero which are not counted). After level l, the number of additions remains constant at L, thus $L \times \frac{N}{2^{i-1}}$ additions are required for $i > l+1$. This results in the following expression for the total number of additions per dimension.

$$1 + \sum_{2}^{l+1} \frac{2^i - 3}{2^{i-1}} + L \sum_{l+2}^{n} \frac{1}{2^{i-1}} \tag{2}$$

Table 1 shows the complexity of LVQ for different values of L. Referring to Table 1 and Fig. 1, we conclude that the case of $L = 8$ results in the lowest encoding complexity among the choices achieving the maximum gain (0.85 dB). For this reason, we will concentrate on $L = 8$ in all our following discussions.

3 Basic Structure for Fixed-rate Entropy Coding

Consider a memoryless source and the N-fold Cartesian product of a scalar quantizer composed of M points. The final vector quantizer is selected as a subset of points from $\Lambda(L, N)$ composed of T elements, denoted as $\mathbf{r}_i, i = 0, \ldots, T-1$, where each N-D point is represented by $\lceil \log_2 T \rceil$ bits.

Assume that the induced self-information and the expected value of the symbols mapped to the jth one-D point along the ith dimension are equal to c_{ij} and r_{ij}, respectively. The N-D reconstruction vectors are obtained by concatenating the corresponding one-D reconstruction levels, namely r_{ij}s. The self-information associated with a one-D point is considered as a cost associated with that point. We select the N-D points for which the overall additive cost is less than a given threshold. This results in choosing points from the high probability region in the N-D space (typical set).

For a given source vector $\mathbf{x}$, the quantization rule (encoding) 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. In practice, as the cardinality of the selected subset of points is very high, one needs an algorithmic approach for the underlying operations (using exhaustive search and/or look up table is not practical).

4 Hierarchical Dynamic Programming

We build a recursive structure using a hierarchy of levels where each level involves the Cartesian product of two lower dimensional subspaces. To explain this structure, let $F_N(C)$ denote the set of the N-D points of the overall (additive) cost C (shell of cost C). We have the following *recursive* relationship:

$$F_N(C) = \bigcup \left[F_{N_1}(C_1) \otimes F_{N_2}(C_2) \right] \tag{3}$$

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 interested in the case that $N_1 = N_2 = N/2$. We refer to each Cartesian product element in Eq. (3) as a *cluster*.

For a given input vector $\mathbf{x}$, by encoding of a shell we mean the process of finding the element of the shell which has the minimum distance to $\mathbf{x}$. To do this, the N-D input vector $\mathbf{x}$ is split into two parts $\mathbf{x}_1$ and $\mathbf{x}_2$ each of length $N/2$. Assume that the nearest vectors of the shells $F_{N/2}(C_1), F_{N/2}(C_2)$ to $\mathbf{x}_1, \mathbf{x}_2$ are equal to $\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2$ with the distances d_1, d_2, respectively. The nearest vector of the cluster $F_{N/2}(C_1) \otimes F_{N/2}(C_2)$ to $\mathbf{x}$ is equal to $(\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2)$ 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 encode all the N-D shells. In other words, one is able to encode an $N = 2^n$ dimensional space in n steps by starting from the one-D subspaces and progressing in a recursive way.

As we are selecting the points from a lattice, we should impose another constraint on the Cartesian products of the shells when we are building the clusters in the next level of the hierarchy. This is achieved by sub-dividing the shells according to the evenness and oddness of their one-D points (following the structure of the Hadamard matrix) and discarding the combinations which do not satisfy the Hadamard conditions in the corresponding Cartesian products. We refer to the subset of points within a given shell which correspond to a specific sequence of *even* and *odd* combinations (determined by the rows of the Hadamard matrix) as a sub-shell and to the corresponding Cartesian product as a sub-cluster. In this case, the states of the system correspond to distinct sub-shells at each level of the hierarchy.

5 State Quantization

The straight-forward approach is to assign an independent state to each possible value of cost of sub-shells at a given level of the hierarchy. Let K denote the number of distinct values of cost along a dimension. Even for a moderate value of K, the number of distinct states in N-D can be impractically large. The solution is to *aggregate* distinct states into a smaller number. This is denoted as the *state-space quantization.*

In the following, we discuss two methods for the state quantization. The first method is from reference [9] and we will briefly explain how it can be used in conjunction with the lattice quantizer proposed here. Then, we present another method for state quantization based on *non-uniform merging* of sub-shells which offers a much lower complexity.

5.1 Uniform Merging

Consider an $N = 2^n$-dimensional space and assume that the hierarchical dynamic programming is achieved in n stages where the ith stage, $i = 0, \ldots, n-1$, involves the Cartesian product of the 2^i-D subspaces. Assume that there are $K_i = 2^{k_i}$ sub-shells of equal cardinality in the ith level of the hierarchy. In the Cartesian product of two of the 2^i-D subspaces, we obtain 2^{2k_i} sub-clusters of equal cardinality where those which do not satisfy the Hadamard constraints are discarded. The remaining sub-clusters are arranged in the order of increasing average cost and a proper number of subsequent sub-clusters are aggregated into a higher level sub-shell. Then, the whole process is repeated recursively where the final quantizer is selected as an appropriate number of the N-D combinations of the lowest average cost.

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

5.2 Non-uniform Merging Using a Binary Tree

Discarding the points of higher cost induces a non-uniform probability distribution on the lower dimensional subspaces such that the points of the lower cost are used more frequently. This fact is in favor of using a higher resolution in the areas of lower cost. These observations suggest that uniform merging as discussed earlier does not result in the best performance. To take this property into account, we assign a different number of points to each sub-cluster.

Assume that there are 2^k sub-shells of equal cardinality at a given stage of our hierarchy. In the 2-fold Cartesian product space, we obtain 2^{2k} sub-clusters which are merged into K sub-shells of integer bit rate, but with non-equal number of points. This is achieved by using a set of integer numbers ℓ_is satisfying $\sum_i 2^{-\ell_i} = 1$ where $2^{-\ell_i}$ is the fraction of the sub-clusters aggregated in the ith sub-shell, $i = 0, \ldots, K-1$. A simple argument shows that the ℓ_is can be selected as the lengths of different paths in any binary tree with K final nodes. This configuration allows to use a set of prefix codes for the underlying addressing.

This non-uniform merging rule is applied in the $(n-2)$th stage of the hierarchy. The corresponding merging rule for the $(n-1)$th stage is the same as uniform merging. Using the non-uniform merging at one of the middle stages enable us to reduce the number of combinations at last stage of hierarchy. Note that by using this strategy the cardinalities of all the subsets are guaranteed to be an integer power of two resulting in a simple addressing scheme. The ℓ_is we use for the binary tree are $(1, 1, 2, 3, 4, 5, ..., 2k-1)$.

6 Numerical Results

In the following, we present some numerical results for the performance and complexity of the proposed method. In all cases, a sequence of 20000 source vector is used to design the quantizer and a different sequence of the same length is used to measure the resulting performance. Input samples are from a memoryless Gaussian source. The quantization performance is measured in terms of the mean square distance. In all comparisons, the memory size is in byte (8 bits) per N dimensions and the computational complexity is the number of additions/comparisons per dimension.

Fig. 2 (a) provides a comparison between our Lattice-based Fixed-rate Entropy-coded Vector Quantizer (L-FEVQ), reference [9] and rate distortion function where we have about 0.85 dB improvement with respect to [9] which is the quantization gain due to using lattice. Fig. 2 (b) provides a comparison between L-FEVQ, LVQ and rate distortion function reflecting the gain achieved through entropy coding for L-FEVQ vs. LVQ. Figure 3 provides a comparison between proposed method and TB-SVQ from reference [14]. Both methods show the same performance, however, as will be explained in the following, our method has a much smaller complexity.

Table 3 shows an approximation of complexity for the proposed method (L-FEVQ) using non-uniform merging of states. The entries of this table are computed using the expressions given in [9] where we have accounted for the effect of the non-uniform merging of states and the memory and computational requirements at each stage of the hierarchy are multiplied by proper factors to reflect the increase in the complexity due to the embedded lattice structure (refer to [1] for more details).

Reference [14] does not provide numerical results concerning the complexity of their method (TB-SVQ). However, TB-SVQ is based on the method presented in [8] where the packing gain and the shaping gain are achieved by two embedded trellis diagrams. In general, using a trellis diagram for packing results in a linear increase in the complexity proportional to the number of states for the method of [8]. The curve in Fig.3 (extracted from [14]) are based on using a trellis with 32 states for the packing, resulting in an increase by a factor of about 32 for the complexity of the TB-SVQ as compared to the values given for [8] in Table 2. Referring to Tables 2, 3, and including the increase in the complexity due to the packing trellis for TB-SVQ [14], we can obtain an estimate of the reduction in complexity for the proposed method as compared to [14] while the performances are very close (refer to Fig. 3). Another drawback of the TB-SVQ vs. our method is the increase in the quantization delay. The results from [14] given in Fig. 3 correspond to a block length of 32 samples for the shaping trellis, but the overall delay in [14] is generally larger due to the extra delay required by the packing trellis. The results for our method given in the same figure correspond to a total delay of 32 samples only.

7 Summary

In this paper, we have combined the hierarchical dynamic programming approach for fixed rate quantization [9] with lattice quantization and have used a binary tree to non-uniformly partition the quantizer space. Numerical results are presented showing that the proposed method bridges a major part of the gap between the LMQ and the rate distortion bound while maintaining fixed-rate outputs and a reasonable overall complexity. The proposed method shows a substantial improvement in performance and/or reduction in the complexity with respect to the best known results reported in [10, 8, 14, 9].

References

[1] S. Nikneshan and A. K. Khandani, "Lattice-based fixed-rate entropy-coded vector quantization," *Proc. The Sixth Canadian Workshop on Information Theory,* CWIT'99, pp. 59–62, June 1999.

[2] J. Max, "Quantizing for minimum distortion," *IRE Trans. Inform. Theory,* vol. IT-6, pp. 7-12, March 1960.

[3] S. P. Lloyd, "Least square quantization in PCM," *IEEE Trans. Inform. Theory,* vol. IT-28, pp. 129–137, Jan. 1982.

[4] H. Gish and J. N. Pierce, "Asymptotically efficient quantizing," *IEEE Trans. Inform. Theory,* vol. IT-14, pp. 676–683, Sept. 1968.

[5] N. Farvardin and J. Modestino, "Optimum quantizer performance for a class of non-Gaussian memoryless sources," *IEEE Trans. Inform. Theory,* vol. IT-30, pp. 485–497, May 1984.

[6] P. A. Chou, T. Lookabaugh, R. M. Gray, "Entropy-constrained vector quantization," *IEEE Trans. Acoust., Speech, Signal Processing,* vol. ASSP-28, pp. 31–42, Jan. 1989.

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

[8] 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.

[9] A. K. Khandani, "A Hierarchical Dynamic Programming Approach to Fixed-rate, Entropy-Coded Quantization," *IEEE Trans. Inform. Theory,* vol. IT-42, pp. 1298–1303, July 1996.

[10] M. W. Marcellin, T. R. Fischer, "Trellis-coded quantization of memoryless and Gauss-Markov sources," *IEEE Trans. Commun.* vol. 38, pp. 82–93, Nov. 1990.

[11] T. R. Fischer, M. W. Marcellin, and M. Wang, "Trellis-coded vector quantization," *IEEE Trans. Inform. Theory,* vol. IT-37, pp. 1551–1566, Nov. 1991.

[12] H. S. Wang and N. Moayeri, "Trellis coded vector quantization," *IEEE Trans. Communication,* vol. 40, pp. 1273–1276, Aug. 1992.

[13] T. R. Fischer, and M. Wang, "Entropy-constrained trellis-coded quantization," *IEEE Trans. Inform. Theory,* vol. IT-38, pp. 413–425, March 1992.

[14] R. Laroia and N. Farvardin, "Trellis-based scalar-vector quantizer for memoryless sources," *IEEE Trans. Inform. Theory,* vol. IT-40, pp. 860–870, May 1994.

[15] J.H. Conway and N. J. Sloane, "Sphere Packings, Lattices and Groups," *Springer-Verlag,* New York , 1988.

[16] G. D. Forney, "Coset codes- Part I: Introduction and Geometrical Classification," *IEEE Trans. Inform. Theory,* vol. IT-34, pp. 1123–1151, Sept. 1988.

L	Adds	Compares
4	2.625	4.125
8	4.125	4.25
16	5.375	4.50
32	7.094	5.00

Table 1: Comparison in complexity of LVQ, for different L, $M = 8$, and $N = 32$

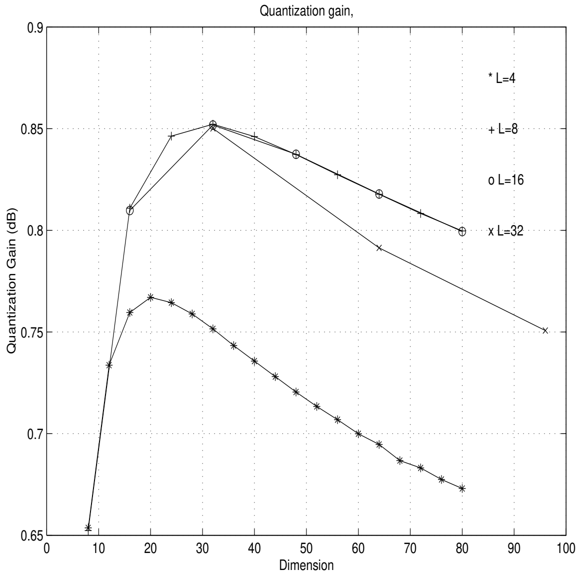
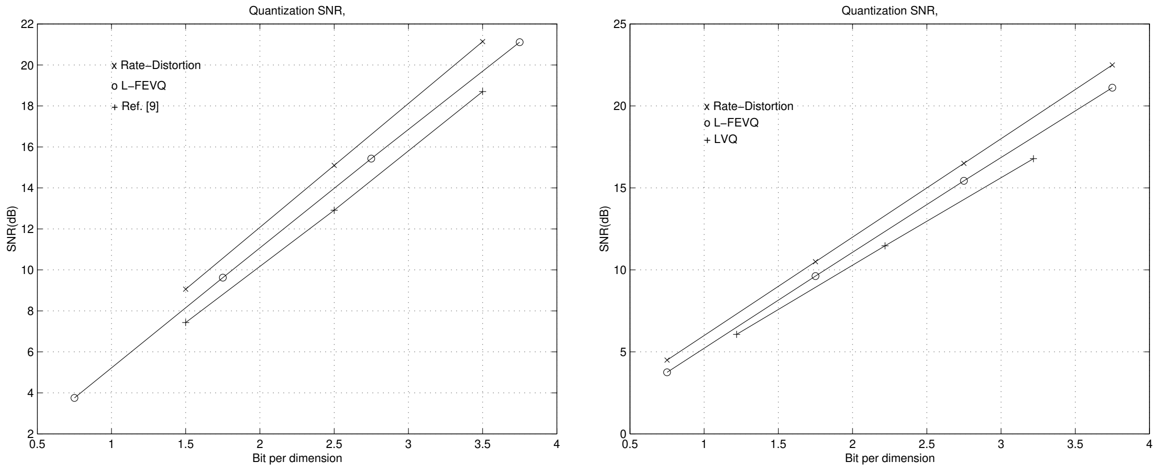


Figure 1: Quantization gain of different lattices as a function of the space dimension for $L = 4, 8, 16, 32$ (rank of the Hadamard matrix).



(a) (b)

Figure 2: Comparison of LVQ, L-FEVQ and reference [9].

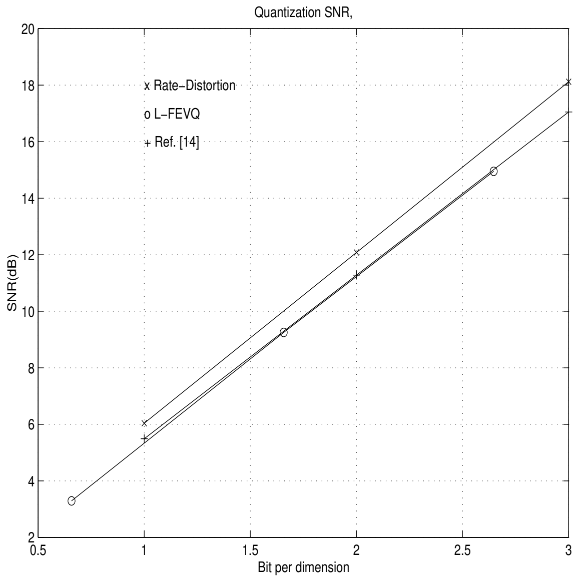


Figure 3: Comparison L-FEVQ (using non-uniform merging) and reference [14].

Method	N	M	R	M_{RAM}	M_{ROM}	$M_{\mathrm{RAM}} + M_{\mathrm{ROM}}$	Computation	SNR (dB)
Ref. [8]	16	4	1.5	N/A	N/A	8 k	600	7.47
Ref. [9]	16	4	1.5	0.6 k	0.8 k	1.4 k	50	7.43
Ref. [8]	16	8	2.5	N/A	N/A	21 k	2000	13.00
Ref.[9]	16	8	2.5	1.0 k	2.0 k	3.0 k	220	12.91

Table 2: Comparison of scheme of [8] and [9] for Gaussian source.

Rate	N	M	L	RAM	ROM	ROM+RAM	Computation	SNR(dB)
0.66	32	4	8	0.2 k	0.5 k	0.7 k	240	3.29
1.66	32	8	8	0.3 k	1.8 k	2.1 k	770	9.32
2.65	32	16	8	0.4 k	1.9 k	2.3 k	800	14.95

Table 3: Quantization SNR vs. rate of L-FEVQ using non-uniform state quantization for Gaussian source