Variable Dimension VQ Encoding and Codebook Design

Anamitra Makur and K. P. Subbalakshmi

***Abstract*—A variable dimension vector quantizer (VDVQ) has codewords of unequal dimensions. Here, a trellis-based sequential optimal VDVQ encoding algorithm is proposed. Also, a VDVQ codebook design algorithm based on splitting a node with equal or reduced dimensions is proposed that does not require any codebook parameter to be prespecified unlike known schemes. The VDVQ system is shown to outperform a few known VQ systems for AR(1) sources.**

***Index Terms*— Source coding, variable-rate coding, vector quantization.**

I. Introduction

A CONVENTIONAL vector quantizer (VQ) [1] uses a fixed dimension for every codeword (hence, every input vector). On the other hand, all codewords of a variable dimension VQ (VDVQ) do not have the same dimension. A VDVQ system may only improve the performance due to its generality. Further, a VDVQ system possesses the advantages of a variable rate coder such as better instantaneous distortion values.

Most attempts to use variable dimensions in VQ [2]–[5] are limited to systems that adapt to instantaneous input statistics to choose the dimension, not addressing the VDVQ encoding and codebook design issue as a whole. Optimal VDVQ encoding should involve parsing the input sequence and nearest neighbor search. Although the dictionary-based lossy coder of [6], [7] encodes from a variable dimension codebook (dictionary), it does not explicitly address this issue since the encoder aims at finding the maximum dimensional match not exceeding a certain distortion threshold. This coder does not require a codebook design either. VDVQ coders addressing the optimal encoding and codebook design may be found in [8]–[10]. The vector dimension in [9] is fixed, while the supervector dimension is variable, which necessitates optimal parsing during encoding.

II. Trellis-Based VDVQ Encoding

Given a codebook containing N codewords of varying dimensions belonging to the set $\mathcal{K} = \{k_{\min}, \cdots, k_{\max}\}$, optimal VDVQ encoding achieves a parsing of the input samples together with a matching of input vectors to codewords, such as to minimize a cost function $D + \lambda R$ between the input samples and the output samples. D denotes the total distortion

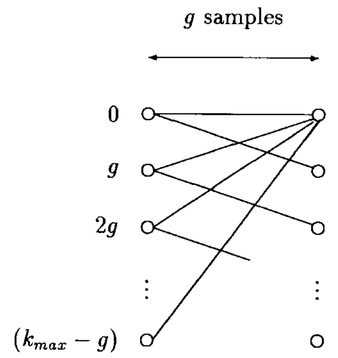

Fig. 1. One stage of the encoding trellis.

(MSE) and R is the total rate (bits). We present a sequential (delayed-decision) optimal VDVQ encoding algorithm using the Viterbi trellis, or dynamic programming.

If g is the greatest common divisor of $\mathcal{K}$,

$$g = \gcd\{k : k \in \mathcal{K}\} \tag{1}$$

then an input sequence cannot be coded entirely unless the number of samples is a multiple of g. Consequently, we only need to consider g samples at a time. During an intermediate stage of encoding when a multiple of g samples are received, these may be parsed in the following ways: the partitions consist of an integral number of vectors (complete partitions, *degree of incompleteness* 0); last g samples are not yet part of any vector—it is to be combined with future samples to form a vector (incomplete partitions, degree of incompleteness g); last $2g$ samples are incomplete (degree $2g$);$\cdots$; last $k_{\max} - g$ samples are incomplete (degree $k_{\max} - g$). A partition of degree $i \cdot g$ at some stage gives rise to a partition of degree $(i+1)g$ at the next stage (provided $i+1$ is less than $k_{\max}/g$), and also to a partition of degree 0 if $(i+1)g \in \mathcal{K}$. Therefore, all possible partitions are the paths of a regular trellis, one stage (g samples) of which is shown in Fig. 1.

Encoding consists of retaining the minimum cost path of each degree, $P_d(s), d = 0, g, \cdots, k_{\max} - g$, at some stage s of the trellis. Each path $P_d(s)$ carries the total cost $C_d(s)$ of itself except the incomplete part, if any. In the next stage, all incomplete paths are updated simply by copying earlier paths:

$$P_d(s+1) = P_{d-g}(s), \qquad \text{for } d = g, 2g, \cdots, k_{\max} - g \tag{2}$$

and the cost is also copied. Since a number of paths converge at degree 0, one full search of the codebook is performed at each stage to determine the minimum cost path:

$$m = \mathrm{argmin_n}\{\mathrm{C_d(s) + D(x(k_n), c_n)} + \lambda r_n : d + g = k_n\} \tag{3}$$

$$P_0(s+1) = P_{k_m - g}(s) \star c_m \tag{4}$$

Paper approved by E. Ayanoglu, the Editor for Communication Theory and Coding Application of the IEEE Communications Society. Manuscript received May 18, 1995; revised September 15, 1996.

A. Makur is with the Department of Electrical Communication Engineering, Indian Institute of Science, Bangalore 560012, India.

K. P. Subbalakshmi is with the School of Engineering Science, Simon Fraser University, Burnaby, B.C. V5A 1S6, Canada.

Publisher Item Identifier S 0090-6778(97)05737-1.

$$C_0(s+1) = C_{k_m - g}(s) + D(x(k_m), c_m) + \lambda r_m \quad (5)$$

where c_n are the codewords with dimensions k_n and index lengths r_n, $x(k)$ are k most recent input samples, $D(,)$ is the (total) distortion measure, and $\star$ denotes concatenation. At any stage, if the first part of all retained paths is found to be identical, then that part is decided (corresponding indexes transmitted). At the end of an input sequence, the degree 0 path is decided as the optimal path. While the encoding delay for this algorithm is observed to be acceptable (e.g., maximum 13 samples for $k_{\max} = 5$), a suboptimal fixed-delay encoding algorithm is obtained if a decision is forced (by majority) on the past sequence that is more than a given number of samples away.

The trellis-based VDVQ encoding proposed here is more efficient than the variable-to-variable VQ (VVVQ) encoding of [8] (which is also used in [9]) in two ways. Since the examples of [8] has $g = 1$, the role of g is not exploited, and one full search is required for every sample. Since the proposed algorithm requires one full search for every g samples, it provides an advantage when $g > 1$. Second, the VVVQ encoding is done by backtracking after the entire input sequence is obtained, essentially meaning infinite delay. The proposed trellis structure allows sequential encoding since a decision is possible for the remote past.

III. Split-Based VDVQ Codebook Design

Tree growing involving splitting a node that produces a minimum distortion to rate slope [11] is extended here to design a VDVQ codebook. Inputs to the algorithm are the maximum dimension $k^0_{\max}$ and the minimum dimension $k^0_{\min}$. The tree begins with a single codeword of dimension $k^0_{\max}$. During the jth iteration, a leaf of the tree will be split. Besides conventional splitting, we consider splitting a codeword of dimension k into two codewords of reduced dimensions l and $k-l$ when $l, k-l \geq k^0_{\min}$. The children are segments of the parent codeword, so that the change in distortion is always nonpositive. A codeword may be split in $k+2-2k^0_{\min}$ ways, producing dimensions $(k,k), (k^0_{\min}, k-k^0_{\min}), \cdots, (k-k^0_{\min}, k^0_{\min})$. For each candidate split for each leaf, GLA with VDVQ encoding is run until convergence. Let D_j and R_j be the total distortion and rate of the tree before the jth iteration. Let D_s and R_s be the resulting distortion and rate after a candidate split is effected. Then this split produces a slope

$$\frac{\Delta D}{\Delta R} = \frac{D_s - D_j}{R_s - R_j}. \quad (6)$$

The split producing a minimum slope is accepted. The algorithm is terminated once some termination condition (maximum rate, or minimum distortion, or maximum number of codewords) is reached.

Like tree growing, this algorithm is greedy since, at each stage, we try to pick the best alternative. However, it is expected to perform better than GLA. While conventional GLA iterates on N codewords, a TSVQ-like variation of GLA known as the split algorithm iterates on $2^0, 2^1, 2^2, \cdots$ codewords in stages. The split algorithm is known to perform better than GLA in avoiding local minima. One would expect

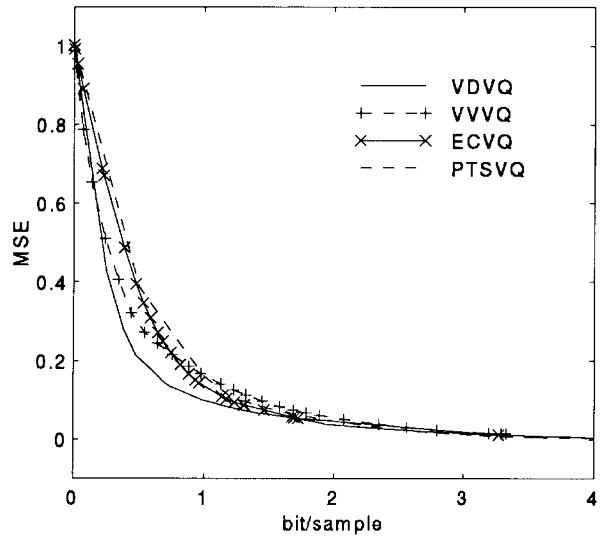

Fig. 2. Rate-distortion performance of VDVQ for AR(1) source, $\rho = 0.9$.

a variation of GLA, that splits one node at a time and iterates on $1, 2, 3, 4, \cdots$ codewords in stages, to perform even better. The proposed algorithm does just that. Each stage, however, converges fast since only one codeword (from a converged codebook) is split. For example, for $N = 64$, the proposed VDVQ codebook design (63 stages) took slightly more than seven times what is taken by GLA using split algorithm (six stages).

The proposed codebook design approach significantly differs from other known VDVQ codebook designs. Most existing schemes segment the codebook into equal dimensional parts, and design each segment independently. VVVQ, while following this principle for the initial codebook, runs GLA on the entire codebook eventually. All of these schemes either require *a priori* knowledge about the signal for designing the segmented codebooks, or require deciding some criteria (for classification) or parameters (such as the allowed dimensions and the number of codewords for each dimension in VVVQ) *a priori* using ad hoc techniques. It is intuitively more appealing to let the training sequence choose any such parameter. The proposed algorithm achieves this since the final values of $k_{\max} \leq k^0_{\max}$ and $k_{\min} \geq k^0_{\min}$, as well as the number of codewords for each dimension, are decided by the training sequence.

IV. Results and Conclusions

The performance of the proposed VDVQ system has been compared with the pruned tree-structured VQ (PTSVQ), entropy-constrained VQ (ECVQ), and VVVQ for identical training sequences. The cost function for the VDVQ is taken as D (i.e., $\lambda = 0$). For fair comparison with other schemes, each of which uses variable-length code, the VDVQ codeword indexes are entropy coded when its performance is measured (but not during codebook design).

Figs. 2 and 3 compare the rate-distortion performance of these systems for an AR(1) source with $\rho = 0.9$ and 0.7 ($\sigma^2 = 1$). VDVQ outperforms all other systems in Fig. 2. Its performance is only matched by VVVQ in Fig. 3, but VDVQ has

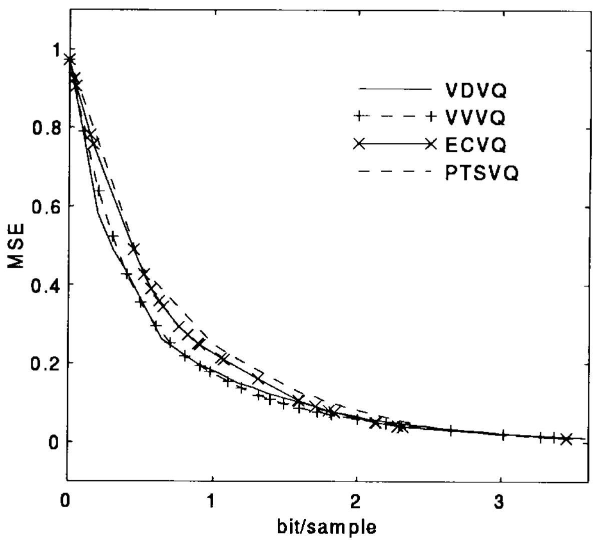

Fig. 3. Rate-distortion performance of VDVQ for AR(1) source, $\rho = 0.7$.

lower complexity and a smaller number of codewords than VVVQ. Table I shows the dimension, codebook size, and the number of distortion computations per sample for all systems for both figures. The parameters for each system are chosen such that the (average) dimensions of all systems are nearly equal. For each point in the figures, the parameters of Table I are averaged over the training sequence. For each curve, the minimum, average, and maximum value of each parameter is shown. It is observed that VDVQ encoding is more complex than only PTSVQ, which has a $\log N$ complexity. Other details about the simulation are: VDVQ $k^0_{\max} = 4(\rho = 0.9)$ and $5\,(\rho = 0.7), k^0_{\min} = 1$; VVVQ has 11, 11, and 10 codewords of dimension 1, 2, and 3; PTSVQ uses recursive optimal pruning from a grown tree at the maximum rate; to obtain points for higher rates, N for ECVQ and maximum N for PTSVQ are chosen large.

In this paper, a sequential optimal VDVQ encoding algorithm based on a trellis has been proposed, which is more suitable for implementation than the known VVVQ encoding. Also, a VDVQ codebook design algorithm based on tree growing has been proposed, which automatically determines the dimensions and the number of codewords for each dimension unlike the known schemes. For AR(1) sources, the proposed VDVQ system performs better than PTSVQ, ECVQ (both fixed dimension), and VVVQ (VDVQ). The VDVQ performance may be further improved by taking $D + \lambda R$ as the cost function.

TABLE I
VARIOUS PARAMETERS FOR FIGS. 2 AND 3

AR	system	dimension			codebook size			computation		
ρ		min	avg	max	min	avg	max	min	avg	max
0.9	VDVQ	1.10	2.09	4.00	1	22	43	0.00	20.20	43.00
	VVVQ	1.17	2.13	2.70	32	32	32	32	32	32
	ECVQ	2	2	2	256	256	256	128	128	128
	PTSVQ	2	2	2	1	164.5	328	0.00	3.18	4.00
0.7	VDVQ	1.06	2.21	5.00	1	13.5	26	0.00	13.18	26.00
	VVVQ	1.17	2.20	2.70	32	32	32	32	32	32
	ECVQ	2	2	2	128	128	128	64	64	64
	PTSVQ	2	2	2	1	87.5	174	0.00	2.74	3.49

ACKNOWLEDGMENT

The authors acknowledge the help of S. Sethu Selvi in running the simulations, and of the reviewers for pointing out the VVVQ work.

REFERENCES

[1] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression.* Boston, MA: Kluwer, 1992.
[2] C. Rutledge, "Variable block vector DPCM: Vector predictive coding of color images," in *Proc. IEEE ICC'87,* pp. 130–135.
[3] J. L. Boxerman and H. J. Lee, "Variable block-sized VQ of grayscale images with unconstrained tiling," in *Proc. IEEE ICASSP'90,* pp. 2277–2280.
[4] Y. Hussain and N. Farvardin, "Variable rate finite state vector quantization of images," in *Proc. IEEE ICASSP'91,* pp. 2301–2304.
[5] S. A. Mohamed and M. M. Fahmy, "Image compression using block pattern VQ with variable code vector dimension," in *Proc. IEEE ICASSP'92,* pp. 357–360.
[6] C. Constantinescu and J. A. Storer, "On-line adaptive vector quantization with variable size codebook entries," in *Proc. DCC'93,* pp. 32–41.
[7] C. Constantinescu and J. A. Storer, "Improved techniques for single-pass adaptive VQ," in *Proc. DCC'94,* pp. 410–419.
[8] P. A. Chou and T. Lookabaugh, "Variable dimension vector quantization of linear predictive coefficients of speech," in *Proc. IEEE ICASSP'94,* pp. I-505–I-508.
[9] M. Effros, P. A. Chou, and R. M. Gray, "Variable dimension weighted universal vector quantization and noiseless coding," in *Proc. DCC'94,* pp. 2–11.
[10] K. P. Subbalakshmi, "Variable dimension vector quantization," M.E. thesis, Dep. Elec. Comm. Eng., Indian Inst. Sci., Bangalore, June 1994.
[11] E. A. Riskin and R. M. Gray, "A greedy tree growing algorithm for the design of variable rate vector quantizers," *IEEE Trans. Signal Processing,* vol. 39, pp. 2500–2507, Nov. 1991.