5/8-32
272334
18.

N90-21898

TDA Progress Report 42-100

February 15, 1990

An Adaptive Vector Quantization Scheme

K.-M. Cheung
Communications Systems Research Section

Vector quantization is known to be an effective compression scheme to achieve a low bit rate so as to minimize communication channel bandwidth and also to reduce digital memory storage while maintaining the necessary fidelity of the data. However, the large amount of computations required in vector quantizers has been a handicap in using vector quantization for low-rate source coding. In this article an adaptive vector quantization algorithm is introduced that is inherently suitable for simple hardware implementation because it has a simple architecture. It allows fast encoding and decoding because it requires only addition and subtraction operations.

I. Introduction

Vector quantization can be used as a source coding technique for speech and images by mapping a sequence of continuous or discrete vectors into a digital sequence suitable for communication over a digital channel or storage in a digital medium. The goal is to reduce the volume of the data for transmission over a digital channel and also for archiving to a digital storage medium, while preserving the required fidelity of the data. This is particularly important to many present and future NASA missions, as the volume of speech and image data in the foreseeable future would become prohibitively large for many communication links or storage devices. It was shown in [1] that a well-designed vector quantization scheme can provide a high compression ratio with good reconstructed quality in the mean-square error sense.

Unlike scalar quantization, where the actual coding of continuous or discrete samples into discrete quantities is done on single samples, the input data of a vector quantization encoder are multi-dimensional blocks of data (input vectors). Most traditional vector quantization schemes [1,2] work as follows: A codebook, which consists of codeword vectors of the same dimension as the input data vectors, is first generated by training a subset of the source data by the LBG (Linde-Buzo-Gray) algorithm [1]. The codebook is then transmitted through the channel. An input vector is encoded by first comparing it with codewords in the codebook. The codeword closest to the input vector is chosen as the quantization vector to represent the input vector. The index of this chosen codeword is then transmitted through the channel. Compression is achieved since fewer bits are used to represent the codeword index than the quantized input data. Notice that the generation of the codebook and the choice of the best codeword for each input vector involve computing the distortion between the input vector and each codeword in the codebook; these processes are usually computationally intensive, especially when the codebook size is large. In this article, we introduce an adaptive vector quantization scheme that does not require codebook generation and is relatively computationally simple.

Bentley et al. [3] and Elias [4] describe a simple heuristic move-to-front protocol to do lossless high rate data compression on text data. This protocol maintains a sequential list of vectors so that the more frequently accessed vectors are near the front. In this article, we modify this protocol and apply it to the design of an adaptive vector quantization scheme to do lossy low rate data compression on speech or image data. This vector quantization scheme exploits the locality of reference by assuming the same vectors are used more frequently over short intervals and then fall into a long period of disuse. It uses a self-organizing list, which will be explained in Section II, as an auxiliary data structure such that the vectors that are more likely to be transmitted are near the front of the list. This dynamic data structure has two advantages: the data are presorted to allow the encoder to efficiently and quickly match the input vectors with the codewords in the list. It also provides inherent adaptability to local statistics by keeping the more recently used codewords in the front of the list while shifting the long unused codewords out of the list. This scheme also has implementation advantages. It has a simple architecture for hardware implementation, and it only requires addition and subtraction operations to allow fast encoding and decoding. This scheme is particularly suitable for real-time compression of audio and video data (both intraframe coding and interframe coding) that will be a large part of communications in future NASA missions. This scheme also has potential uses in commercial areas like high-definition television and audio/video communications using telephone lines. The results reported here are, however, preliminary.

Section II outlines the basic algorithm and some variations of it. Section III describes the performance analysis method and the experimental results, and Section IV gives the concluding remarks.

II. Basic Algorithm

Unlike the traditional vector quantization schemes, this scheme does not require the computationally intensive process of generating a fixed codebook in the first place. The encoder and decoder keep an identical codeword table of the same size. The table is constructed on the fly by exploiting the local statistics of the data as the vector quantization encoder is reading in the input vectors. The codewords in the table, and the positions of codewords in the table, are constantly updated according to a self-organizing protocol such that the more recently used codewords are near the front of the table. Codewords that have not been used for a long time would eventually reach the end of the list and be discarded. The following is a more quantitative discussion of this algorithm.

Suppose the original p-dimensional digital data file size is $L_1 \times L_2 \times \ldots \times L_p$ pixels and each pixel's intensity is represented by k bits. Let each vector represent a block of $l = s_1 \times s_2 \times \ldots \times s_p$ pixels, $s_i | L_i$, for $i = 1, \ldots, p$. Let the codeword table have size N, $N = 2^m - 1$, and let the all ones' m-tuple (i.e., m ones'), denoted by a, be reserved to indicate the transmission of an uncoded vector while the rest of the m-tuples are used to represent the indices of codewords in the table. Let $\mathbf{r}(t)$ be the input vector and $\mathbf{c}_i(t)$ be the ith element in the codeword table at time t. Both $\mathbf{r}(t)$ and $\mathbf{c}_i(t)$ have the same dimensionality $s_1 s_2 \ldots s_p$. Let $D(\mathbf{r}(t), \mathbf{c}_i(t))$ be the distortion function associated with the vector quantization coder. The encoder computes $D(\mathbf{r}(t), \mathbf{c}_i(t))$ for $i = 0, \ldots, N-2$. Let s be the index of the codeword corresponding to the minimum distortion with $\mathbf{r}(t)$. That is,

$$s = \text{index achieving} \min_{i=0,\ldots,N-2} D(\mathbf{r}(t), \mathbf{c}_i(t))$$

The minimum distortion $D(\mathbf{r}(t), \mathbf{c}_s(t))$ is then compared to a preset threshold T. If $D(\mathbf{r}(t), \mathbf{c}_s(t)) \leq T$, the index s is sent. The codeword table is then updated as follows:

$$\text{tmp} \leftarrow \mathbf{c}_s(t) \tag{1}$$

$$\mathbf{c}_i(t+1) \leftarrow \mathbf{c}_{i-1}(t) \qquad i = s, \ldots, 1 \tag{2}$$

$$\mathbf{c}_0(t+1) \leftarrow \text{tmp} \tag{3}$$

If $D(\mathbf{r}(t), \mathbf{c}_s(t)) > T$, the reserved all ones' m-tuple a is sent, which is followed by the transmission of the uncoded input vector $\mathbf{r}(t)$. The codeword table is then updated as follows:

$$\mathbf{c}_i(t+1) \leftarrow \mathbf{c}_{i-1}(t) \qquad i = N-2, \ldots, 1 \tag{4}$$

$$\mathbf{c}_0(t+1) \leftarrow \mathbf{r}(t) \tag{5}$$

The decoder on the other end maintains the codeword table, which is identical to the encoder's. When a legitimate index s of the codeword table is received, the decoder outputs the codeword corresponding to s and updates the table as in Eqs. (1), (2), and (3). When the reserved m-tuple a and the uncoded input vector $\mathbf{r}(t)$ are received, the decoder outputs $\mathbf{r}(t)$ and updates the tables using Eqs. (4) and (5). The encoding and the decoding algorithms are summarized in Fig. 1 and Fig. 2, respectively.

The speed of the vector quantization encoder can be greatly enhanced by performing a partial search for the codeword of minimum distortion rather than a full search. One efficient and simple way is to compute $D(\mathbf{r}(t), \mathbf{c}_i(t))$ sequentially along the self-organized list of codewords and to pick the first codeword $c_s(t)$ in the list with $D(\mathbf{r}(t), \mathbf{c}_s(t)) \leq T$. Another partial search algorithm is to search the first n codewords in the list, where $n < N - 1$, and pick the minimum if it is less than T. In most cases these partial search schemes can give the codeword of minimum distortion. This is because the codewords in the table, and the positions of codewords in the table, are constantly updated according to a self-organizing protocol such that the more recently used codewords, thus the most likely codewords with minimum distortion, are near the front of the table. The performance degradation from using partial search algorithms instead of full search is in most cases unnoticeable. The speed of the encoder can also be increased by choosing a simpler distortion measure like the sum of absolute difference between pixels in $\mathbf{r}(t)$ and $\mathbf{c}_i(t)$.

This scheme has the following advantages. It is inherently suitable for simple hardware implementation. The encoder and decoder maintain their codeword tables, which can be implemented using a set of registers, by performing only simple shifting operations. The computation of distortion between input vector and codeword vector at the encoder side requires only simple addition and subtraction operations. It allows fast encoding and decoding, and unlike many traditional vector quantization schemes that take two or more passes over the data, this scheme requires only one pass.

This scheme has one disadvantage for communications systems requiring rigid data formatting. Unlike most conventional vector quantization encoding schemes that keep the compression rate of the input data files constant and let the distortion drift, this scheme bounds the distortion and lets the compression rate vary according to the degree of activity in the data. Thus the sizes of the compressed data files cannot be a priori specified, and this would create difficulties in specifying the data format of the overall communication system. One way to circumvent this problem is to iteratively re-encode an input data file using different distortion thresholds until the compressed file size matches a certain preset data format. Another solution is to restrict the number of uncoded vectors per data file to be sent such that the compressed file size remains constant for each data file. The first solution increases the overall computation time and the second solution degrades the overall distortion performance.

III. Performance Analysis and Experimental Results

The compression ratio of a vector quantization scheme on a data file is defined to be the ratio of the original data file size to the compressed data file size. A distortion measure D is an assignment of a cost $D(\mathbf{r}(t), \mathbf{c}_i(t))$ of reproducing any input vector $\mathbf{r}(t)$ by a code vector $\mathbf{c}_i(t)$. Given such a distortion measure, the performance of a vector quantization scheme can be measured in terms of the compression ratio versus the average distortion $\mathrm{E}[D(\mathbf{r}(t), \mathbf{c}_i(t))]$ between the input and the final reproduction.

A good distortion measure should be computationally tractable to allow easy analysis and subjectively meaningful so that large or small measured distortions closely correspond to bad and good subjective quality. In this article, we choose to use the conventional squared error distortion measure, which is defined as follows:

$$\begin{aligned} D(\mathbf{r}(t), \mathbf{c}_i(t)) &= \|\mathbf{r}(t) - \mathbf{c}_i(t)\| \\ &= \sum_{j=0}^{l-1} (r^j(t) - c_i^j(t))^2 \end{aligned} \qquad (6)$$

It is also common practice to measure the performance of a system by the signal-to-noise ratio (or signal-to-quantization-noise ratio) in dB. That is,

$$\mathrm{SNR} = $$

$$10 \log_{10} \frac{2^{2k}}{\frac{1}{L_1 L_2} \sum_{i=0}^{L_1 - 1} \sum_{j=0}^{L_2 - 1} (x_{orig.}(i,j) - x_{decoded}(i,j))^2} \qquad (7)$$

where $x(i,j)$ is the intensity value of the (i,j)-th pixel.

Both planetary and nonplanetary images (8 bits per pixel) with various degrees of activity were used in this data compression experiment, and the compression ratio, the root-mean-square-error per pixel, and SNR performance are tabulated in Table 1. The original picture and a decoded picture (compression ratio 8.325 and SNR 34.68 dB) of "Mercury" are given in Fig. 3 and Fig. 4, respectively. The original picture and a decoded picture (compression ratio 4.163 and SNR 31.57 dB) of "USC411" are given in Fig. 5 and Fig. 6, respectively.

An inherent disadvantage of low-rate vector quantization image coding is the annoying presence of blockiness in the decoded image. Because of the mismatch between the pixel values of two spatially adjacent codewords, discontinuity is developed at the block boundaries and a staircase effect shows at the edges. This degradation is usually the most noticeable and most annoying quantization error. Simple image enhancement techniques such as prefiltering and postfiltering can be applied to the decoded image to eliminate the obvious blockiness in the decoded version of the picture.

IV. Concluding Remarks

In this article, a locally adaptive vector quantization scheme, which has a simple architecture to facilitate hardware implementation, was introduced. This scheme adapts efficiently to local statistics by using a simple heuristic method for self-organizing sequential search introduced in [3] and [4]. This scheme is fast, simple and robust, and has potential use for real-time compression of audio and video data in future NASA missions and in commercial areas such as high-definition television, facsimile, and audio/video communications using telephone lines. Performance in noise is a big, open issue.

References

[1] R. M. Gray, "Vector Quantization," *IEEE ASSP Magazine*, April 1984, vol. 1, pp. 4–29.

[2] N. Jayant and P. Noll, *Digital Coding of Waveforms–Principles and Applications to Speech and Video*, Prentice-Hall, 1984.

[3] J. Bentley, D. Sleator, R. Tarjan, and V. Wei, "A Locally Adaptive Data Compression Scheme," *Communications of the ACM*, April 1986, vol. 29, pp. 320–330.

[4] P. Elias, "Interval and Recency Rank Source Coding: Two On-Line Adaptive Variable-Length Schemes," *IEEE Transactions on Information Theory*, January 1987, vol. 33, pp. 3–10.

Table 1. Compression ratio versus distortion performances

File name	File size	C.R.	RMSE/pixel	SNR, dB
Mercury	256 x 256	8.32	4.72	34.68
		5.49	3.66	36.90
USC411	256 x 256	4.16	6.75	31.57
		2.58	4.60	34.85
Saturn6	800 x 800	18.75	7.40	30.75
		12.54	3.70	36.80
		9.29	3.00	38.63
Lena	512 x 512	15.94	20.88	21.77
		6.91	14.12	25.16
		3.91	10.08	28.10
		2.58	7.42	30.75

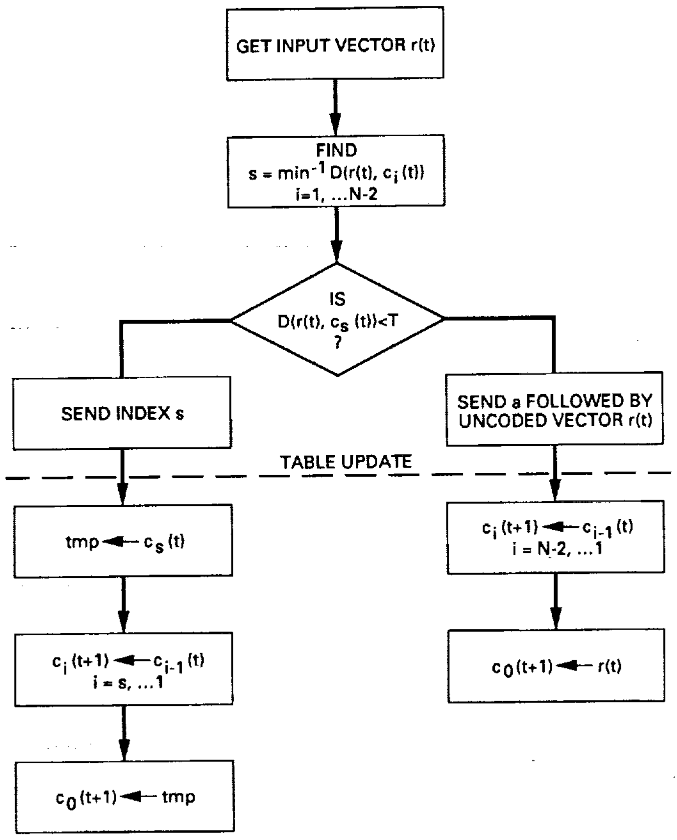
GET INPUT VECTOR r(t)
FIND
$s = \min^{-1} D(r(t), c_i(t))$
$i=1, \ldots N-2$
IS
$D(r(t), c_s(t)) < T$
?
SEND INDEX s
SEND a FOLLOWED BY UNCODED VECTOR r(t)
TABLE UPDATE
$tmp \leftarrow c_s(t)$
$c_i(t+1) \leftarrow c_{i-1}(t)$
$i = s, \ldots 1$
$c_0(t+1) \leftarrow tmp$
$c_i(t+1) \leftarrow c_{i-1}(t)$
$i = N-2, \ldots 1$
$c_0(t+1) \leftarrow r(t)$

Fig. 1. Encoding algorithm.

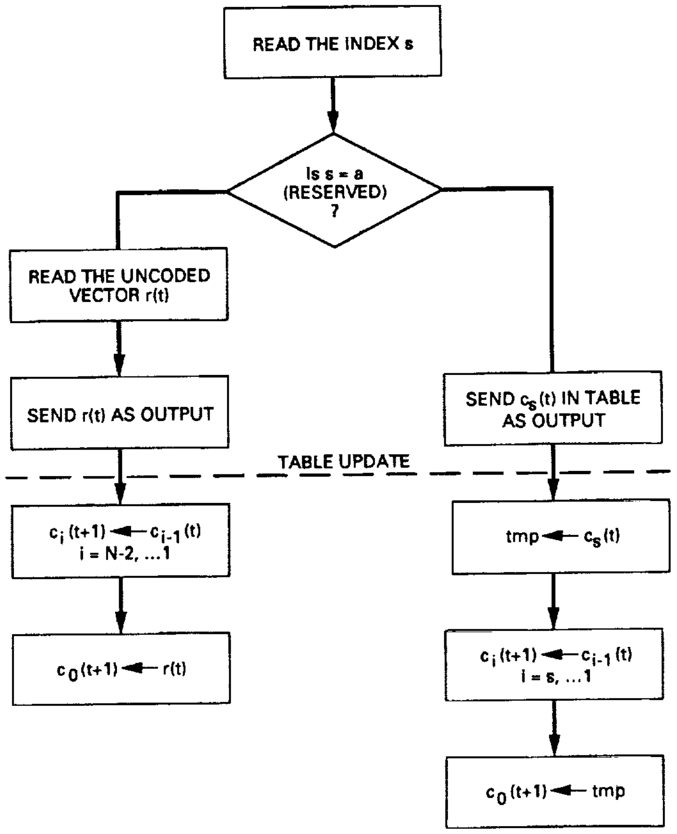
READ THE INDEX s
Is s = a
(RESERVED)
?
READ THE UNCODED VECTOR r(t)
SEND r(t) AS OUTPUT
SEND $c_s(t)$ IN TABLE AS OUTPUT
TABLE UPDATE
$c_i(t+1) \leftarrow c_{i-1}(t)$
$i = N-2, \ldots 1$
$c_0(t+1) \leftarrow r(t)$
$tmp \leftarrow c_s(t)$
$c_i(t+1) \leftarrow c_{i-1}(t)$
$i = s, \ldots 1$
$c_0(t+1) \leftarrow tmp$

Fig. 2. Decoding algorithm.

ORIGINAL PAGE
BLACK AND WHITE PHOTOGRAPH

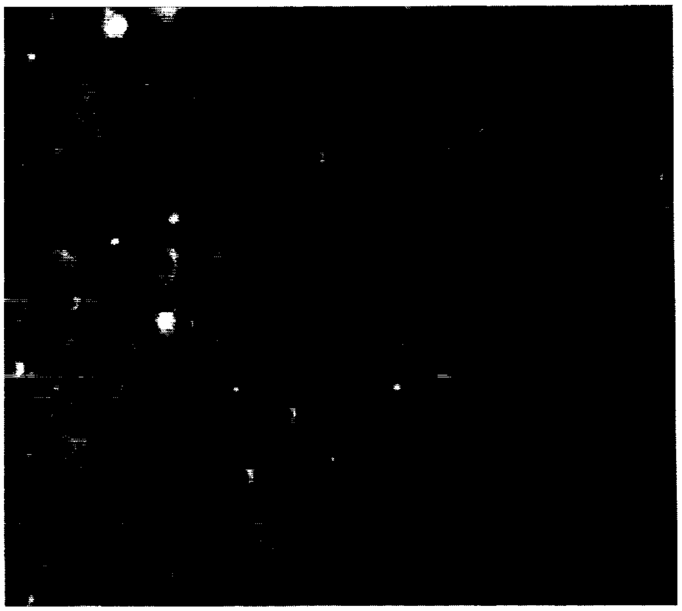
Fig. 3. Original image of "Mercury".

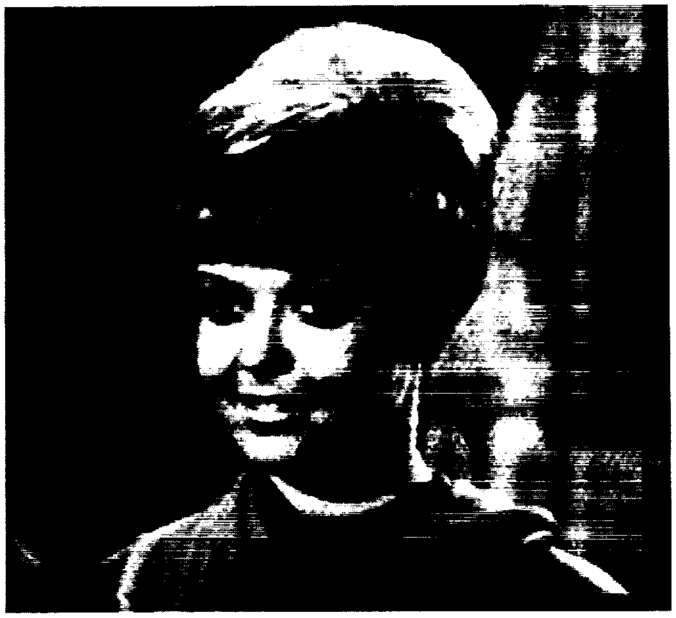
Fig. 5. Original image of "USC411".

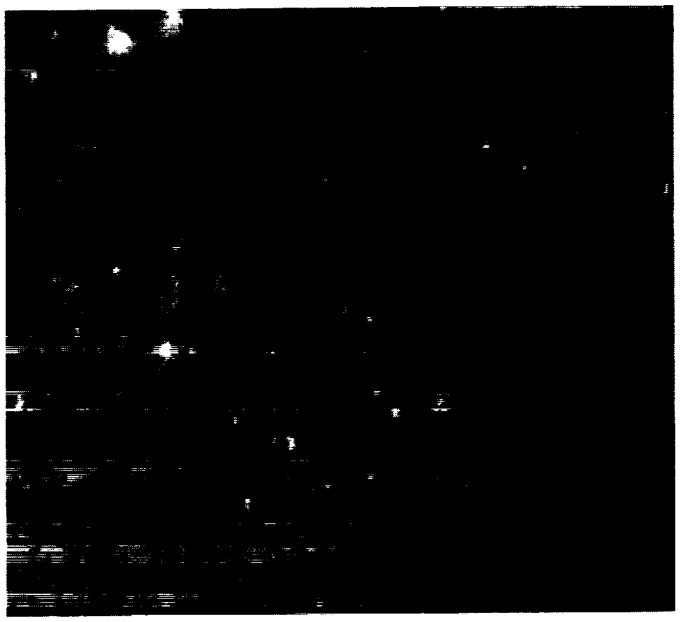
Fig. 4. Reconstructed image of "Mercury".

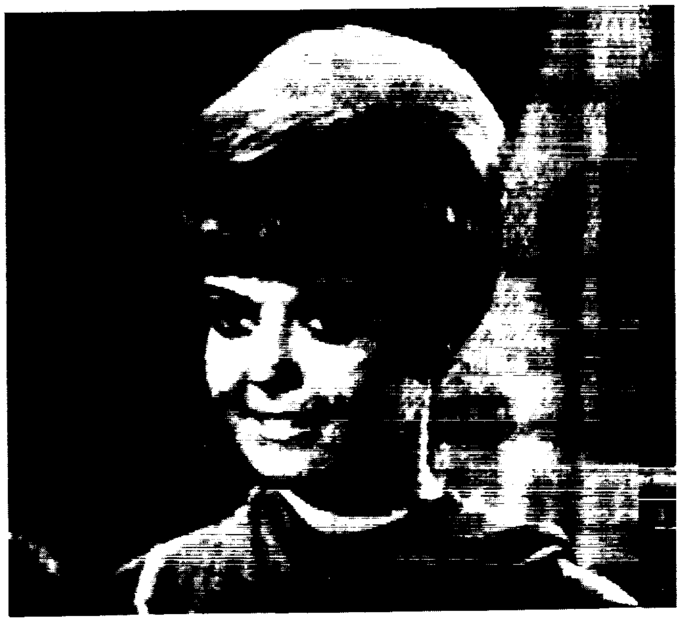
Fig. 6. Reconstructed image of "USC411".

ORIGINAL PAGE IS
OF POOR QUALITY