백터양자화기의 구조적 코더 찾기

STRUCTURED CODEWORD SEARCH FOR VECTOR QUANTIZATION

우홍체
대구대학교, 정보통신공학부
경북경산시 진량읍 내리리

Hong Chae Woo
School of Computer and Comm.
Taegu Univ., Kyungpook, Korea

ABSTRACT

Vector quantization (VQ) is widely used in many high-quality and high-rate data compression applications such as speech coding, audio coding, image coding and video coding. When the size of a VQ codebook is large, the computational complexity for the full codeword search method is a significant problem for many applications. A number of complexity reduction algorithms have been proposed and investigated using such properties of the codebook as the triangle inequality. This paper proposes a new structured VQ search algorithm that is based on a multi-stage structure for searching for the best codeword. Even using only two stages, a significant complexity reduction can be obtained without any loss of quality.

I. INTRODUCTION

Vector quantization (VQ) is a powerful data compression method that has been widely applied. As more high-quality and high-rate speech, audio, image, and video applications have been developed, the importance of VQ methods has grown significantly. The development of powerful low-cost computers and digital signal processing chips has made VQ applications relatively straightforward. However, there are still serious storage and complexity problems for VQ applications that use large size codebooks [1].

A codebook stores codewords (or code vectors) that are used to encode an input vector. The VQ search algorithm chooses the codeword that is minimally distant from the input vector and is the best representative. The criteria for making the selection is based upon a distortion measure that is often the minimum Euclidean distance between the input vector and the codeword. VQ can be regarded as a mapping from a k dimensional input vector set X to the codebook C with the same dimension. The encoder of VQ can be expressed as:

$$Encoder: X \rightarrow C$$

where $X=\{x_i\}, i=1,\ldots,L$, $C=\{c_i\}, i=1,\ldots,N$, x_i is an input vector, there are L input vectors, c_i is a codeword and there are N codewords . The number N is 2^b in which the b bits are allocated for encoding. The decoding is a simple table look-up routine using the decoded VQ index. In VQ, the complexity of encoding is much higher than that of decoding because of the requirement to find the best (minimum distortion) codeword. In real-time applications, the reduction of encoding complexity is often essential.

II. Fast Codeword Search Algorithms

When a codebook has N codewords, and the dimension of a vector is k, and the minimum mean square error (MMSE) distortion measure is selected, the conventional full search algorithm of VQ requires $N \times k$ multiplications, $N \times (2k-1)$ additions and subtractions, and $N-1$ comparisons. As the dimension of a vector or the size of a codebook is increased, these numbers become very large and the full search method is often not viable. A variety of fast codeword search algorithms for reducing computational complexity have been designed and studied. The basic idea of all these algorithms is to reduce the number of candidate codewords to which the full search routine is applied by using some form of pre-selection technique.

The partial distortion elimination (PDE) algorithm rejects the codeword as a candidate for full search if the partially accumulated distortion of the first j

elements ($1 \leq j < k$) is larger than the current minimum distortion in the best codeword search routine [2]. If the current minimum distortion is near the final minimum distortion early in the search routine, the algorithm can be effective. If a prediction of the initial minimum distortion is included in the PDE algorithm, it can be more effective. This is called the *predictive PDE* [3].

Using the triangle inequality relation, unlikely codewords can be eliminated as candidates. That is, the codeword c_i is rejected as a candidate if

$$d(c_i, c_j) > 2 \cdot d_{\min}$$

where $d(c_i, c_j)$ is the distortion measure, c_j is the current best match, and $d_{\min}$ is the current minimum distortion. Note that the distances between every pair of codewords in the codebook are calculated and stored, and this requires an extra memory with $N \times (N-1)$ locations [4].

The *partial search partial distortion* (PSPD) algorithm stores the mean of each codeword. Only codewords whose mean values are in the range from the maximum, $m_{\max}$ to the minimum, $m_{\min}$ are searched. The range is given as follows:

$$m_{\max} = m_{xi} + d_{\min} / \sqrt{k}$$

$$m_{\min} = m_{xi} - d_{\min} / \sqrt{k}$$

where m_{xi} is the mean of an input vector x_i. The PDE algorithm is then applied to the codewords in the specified mean range [5].

In the *classified pre-selection* method, the codewords are grouped in clusters. The centroid of each cluster is stored. Instead of searching over all the codewords, the method first finds 3 or 4 clusters near the input vector by computing distances between the input vector and the cluster centeroids. A full search for the best codeword is performed on only the codewords belonging to the pre-selected clusters [6].

A different approach in complexity reduction of VQ is to search an unconstrained VQ using a constrained search pattern. Historically, this is done for constrained VQs (tree-structured VQs, residual VQs, and multi-stage VQs) with great success, but the associated constraints on the VQ itself limits the overall performance. In our approach only the search is constrained – the VQ codebook is not.

III. Structured Codeword Search

A. Design of Codeword Cluster

The design of a VQ codebook is an NP-hard problem so that sub-optimal methods are normally used. A well-known and commonly used algorithm is the Linde-Buzo-Gray (LBG) codebook design [1]. For a given training data, the algorithm works as follows:

- Select N initial codewords in the codebook.
- Cluster the training data around each codeword using an Euclidean distance measure. This is called the nearest neighbor condition.
- Compute the centeroid of each cluster to create a new codeword.
- Repeat step 2 and 3 until the changes in codewords are relatively small.

There are many variations of the LBG algorithm for selecting a good set of initial codewords, reducing the design complexity, or obtaining a globally optimal codebook. However, the LBG algorithm is very simple and it is popular for VQ codebook design even though it is only locally optimal and it often converges slowly to a local optimum.

The full search on VQ codebook can be broken into multi-stage search. This is, in effect, like zooming in on the best codeword in stages. We call this the multi-stage fast search (MSFS) algorithm. Given an input, the MSFS algorithm can search for the best matched codeword more efficiently without introducing additional distortion. In the MSFS algorithm, there can be any number of search stages, but two stage structures were investigated in this research.

Our basic assumption is that we begin with a one-stage optimal codebook. Our goal is to design a fast search algorithm that will still find the best codeword without doing a full search of the whole codebook. In our technique, we first design a smaller codebook that is created by using a clustering algorithm on the codewords of the optimal full VQ codebook. Then, rather than searching all codewords in the full codebook, the distance between an input vector and vectors of the new smaller codebook is first computed, and the nearest cluster to the input vector is then selected. This is the first stage. In the second stage, a full-search is applied only to codewords belonging to that cluster. In this research, the centroids $\{g_1, g_2, \Lambda, g_{N/2}\}$ of codeword clusters in the first stage are designed with a modified LBG algorithm.

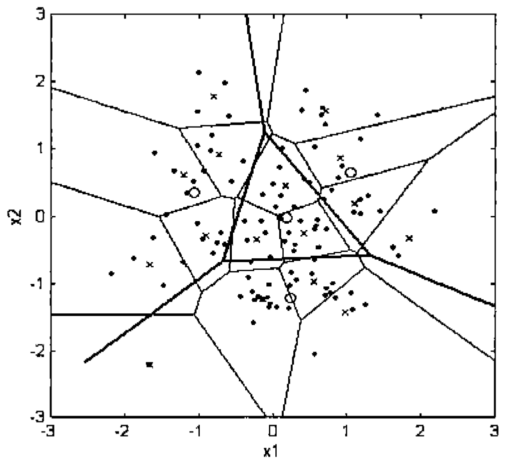
Suppose that N codewords, $\{c_i\}$ with dimension k, are already generated by the LBG algorithm. Then, the regions, V_i which are optimal partitions of the input training data around each codeword are also known. For the first stage of the MSFS, the codewords are partitioned into $N/2$ clusters. The centroids of codeword clusters are generated as follows.

- Select $N/2$ initial centeroid, $\{g_i\}$.
- Cluster the codewords $\{c_i\}$ around each centeroid g_i using an Euclidean distance measure.
- Compute the new centeroid, g_i of each cluster with the original training data belonging to all the V_i regions of the codewords in the current codeword cluster.
- Repeat steps 2 and 3 until the changes in the centeroids of the codeword clusters are relatively small.

Note that step 3 is different from the usual LBG algorithm. The modified LBG algorithm that has codewords $\{c_i\}$ as its training data uses the same nearest condition as the traditional VQ, but in computing the centeroids it uses the original data in the V_i regions instead of codewords. These centeroids need an additional $N/2$ memory locations to be stored.

B. Fast Codeword Search

Using an example, the details in the MSFS algorithm are illustrated in this section. First, zero mean and unit variance random Gaussian data are generated as training data in a VQ design. The codebook in the VQ is designed with the general LBG algorithm, and the



centeroids of the codeword clusters are designed with the above modified LBG algorithm.

Figure 1. The V_i regions and the boundaries of codeword clusters (100 Gaussian data -- •, 16 codewords -- ×, 4 codeword clusters -- O)

Figure 1 shows 100 Gaussian data (marked as •), 16 codewords (marked as ×), the centeroids (marked as O) of the codeword clusters, the border lines (thin solid lines) of V_i regions of the codewords, and the border lines (dark solid lines) of the codeword clusters.

The problem that can be seen in Figure 1 is that the border lines of the V_i regions of the codewords are not the same as those of the codeword clusters. The disparity in border the lines causes some additional searching distortion in VQ. This is solved by identifying which V_i regions in a codeword cluster boundary are subsets of a large data set, U_i belonging to the codeword cluster boundary or which V_i regions are located across the boundaries of the codeword cluster. All the codewords in these two groups of V_i regions are related to the current centroid, g_i. After this procedure, each centeroid g_i of codeword cluster contains the set of codeword index to be searched in the second stage.

The summary of the MSFS algorithm is as follows:

- Search for the nearest centroid, g_i, to an input, x_i.
- Search for the nearest codeword, c_i, in the codeword index set of the current, g_i.

Note that each codeword index set of g_i needs additional memory locations, but only the codeword indices must be stored. Table 1, as an example, shows the number of codewords in each codeword cluster for the data from Figure 1.

Table 1. The number of codewords in each codeword cluster(16 codewords)

Code word cluster no.	No. of subsets V_i	No. of V_i across border	Total no. of codewords per cluster
1	4	1	5
2	5	0	5
3	1	2	3
4	4	1	5

Table 1 shows that, in a codebook of 16 codewords, when using the MSFS algorithm, a maximum of 9 distortion measure computations are needed in the codeword search algorithm, of which four are necessary in the first stage and a maximum five are required in the second stage. Note that the full search needs 16 distortion measure computations. In a small size codebook such as that shown in Table 1, the reduction of computations is not very great, but the advantage in large size codebook will be substantial.

In Table 2, the larger codebook of size 256 is investigated with vectors of dimensions 2 and 5. The MSFS algorithm here is applied to 16 codeword clusters. In the case of vectors of dimension 2, a the maximum of 45 distortion measure computations are required, in which 16 computations are necessary in the first stage and the maximum 29 computations are required in the second stage.

Table 2. The number of codewords in each codeword cluster(256 codewords)

Codeword cluster no.	Total no. of codewords (dim=2)	Total no. of codewords (dim=5)
1	21	23
2	25	28
3	20	34
4	29	35
5	21	30
6	17	34
7	25	31
8	26	33
8	20	33
10	23	34
11	20	37
12	21	35
13	23	34
14	19	41
15	20	30
16	22	33

In the full search method, 256 computations are required. In the case of vectors of dimension 5, the MSFS algorithm needs a maximum of 57 distortion measure computations in which the maximum 41 computations in the second stage are required in the 14th codeword cluster. Thus 18 percent of the full search computational burden is necessary in the 2 dimension case when the MSFS algorithm is applied and 22 percent of the full search is required for the 5 dimension 5. For both cases, an average of about 20 percent of the full search computational burden is required. Note that, in this example, there is no additional distortion introduced by the MSFS algorithm.

The MSFS requires 16 additional memory locations for storing the centeroids of the codeword clusters and about $2N$ memory locations for storing integer index sets in each codeword cluster. Also note that by combining the MSFS algorithm with other fast codeword search algorithms (discussed in Section 2) at each stage of the MSFS, additional complexity reduction is possible.

IV. SUMMARY

The multi-stage fast search algorithm for VQs was designed and tested with Gaussian training data. The centroids of codeword clusters in the first stage of the MSFS were designed with the modified LBG algorithm. In the codebook of size 256, the complexity with the MSFS of only two stages is reduced to 20 percentage of the full search complexity without introducing any additional distortion. Since this is very significant in real applications, the VQ method with the MSFS can be expected to give improved performance for many applications.

V. REFERENCES

[1] A. Gersho, and R.M. Gray, *Vector quantization and data compression*, Kluwer, Massachusetts, 1992.

[2] C.D. Bei, and R.M. Gray, "An improvement of the minimum distortion encoding algorithm for vector quantization," *IEEE Trans. on Comm.*, vol. 33, no. 10, pp. 1132-1133, 1985.

[3] J. Ngwa-Ndifor, and T. Ellis, "Predictive partial search algorithm for vector quantization," *IEE Electronic Letters*, vol. 27, pp.1722-1723, 1991.

[4] S.H. Huang, and S.H. Chen, "Fast encoding algorithm for VQ-based image coding," *IEE, Eletronic Letters*, vol. 26, pp. 1618-1619, 1990.

[5] G. Poggi, "Fast algorithm for full-search VQ encoding," *IEE Electronic Letters*, vol. 13, pp. 1141-1142, 1991.

[6] C.Q. Chen, S. H. Koh, and I.Y. Soon, "Fast codebook search algorithm for unconstrained vector quantization," *IEE Proc.-Vis. Image Signal Process*, vol. 145, no. 2, pp. 97-102, 1998.