Fast Codeword Search Algorithm for Vector Quantization Based on Hyperplane Partitioning Method

Ahmed Swilem[†], **Kousuke Imamura**[††] **and Hideo Hashimoto**[††]

[†] Graduate School of Natural Science & Technology, Kanazawa University
(2-40-20, Kodatsuno, Kanazawa-shi, Ishikawa, 920-8667, Japan)
[††] Faculty of Engineering, Kanazawa University
(2-40-20, Kodatsuno, Kanazawa-shi, Ishikawa, 920-8667, Japan)
TEL 076-234-4894

Abstract Vector quantization is the process of encoding vector data as an index to a dictionary or codebook of representative vectors. One of the most serious problems for vector quantization is the high computational complexity involved in searching for the closest codeword through the codebook. In this paper, we describe a new method allowing significant acceleration of codebook design and encoding processes for vector quantization. This method has feature of using a suitable hyperplane to partition codebook and image data. Experimental results are presented on image block data. These results show that our method performs better than the previously known methods.

Key words: Vector quantization, Fast search algorithm, Hyperplane decision rule.

1. Introduction

A standard vector quantization (VQ) [1] is an efficient compression technique for which many variant [2] are known. It is defined as a mapping Q from a k-dimensional Euclidean space R^k to a finite set $Y = \{y_1, y_2, ..., y_N\}$ of vectors in R^k called the codebook. Each representative vector y_i in the codebook is called a codeword. A complete description of vector quantization process includes three phases: codebook design, encoding and decoding. The objective of codebook design is to construct a codebook Y from a set of training vectors using clustering algorithms like the generalized Lloyd algorithm (GLA) [1]. This codebook is used in both the encoder and the decoder. The encoding phase is equivalent to finding the vector $Q(x) = y_i \in Y$ minimizing the distortion $D(x, y_i) = d^2(x, y_i)$ defined as the squared Euclidean distance, where $d(x, y_i)$ is the Euclidean distance between the vector x and y_i. The decoding phase is simply a table look-up procedure that uses the received index i to deduce the reproduction codeword y_i, and then uses y_i to represent the input vector x.

The computational cost of finding the closest codeword in the codebook design and encoding of VQ imposes practical limits on the codebook size N. When N becomes larger, the computational complexity problem for full codebook search occurs. To avoid such an exhaustive search through the codebook, many fast algorithms [3-7] have been proposed. These algorithms reduce the computational complexity by performing some simple tests before computing the distortion between the training vector and each codeword, and then rejecting those codewords that fail in the tests.

This article introduces a new algorithm to reduce the time complexity of the codebook search using a hyperplane decision rule. The algorithm separates the codebook into two parts and searches in one part according to the input vector feature. The efficiency of the algorithm is compared with Lee & Chen method [4].

The following section reviews one of the most interesting fast search algorithm [4]. Section 3 describes the hyperplane decision method in detail. Some experimental results are shown in section 4 and concluding remarks are given in section 5.

2. Equal-average hyperplane partitioning

The codeword assignment problem in vector quantization is a Nearest Neighbor Search (NNS) problem. Guan et al.[3] introduced an equal-average nearest neighbor search (ENNS) algorithm based on using hyperplanes orthogonal to the central line to partition the search space. This algorithm uses the mean value as a feature to reject unlikely codewords. Let l be the central line, which the vector $u = (1, 1, ..., 1)$ lies on. Any point $a = (a_1, a_2, ..., a_k)$ on l will have $a_i = a_j, i, j = 1, ..., k$. The hyperplane S normal to l, which intersects l at a point $L_s = (m_s, m_s, ..., m_s)$, is written as:

$$S(x): u x^T = \sum_{i=1}^{k} x_i = k\, m_s. \qquad (1)$$

Each point on S has the same mean value m_s. Such a hyperplane is called an equal-average hyperplane. For an input vector x with mean value m_x, the algorithm finds the codeword y_b that has the minimum mean difference to x and calculates the distance r_b between x and y_b. Any codeword that is closer to x than y_b has to be located inside the hypersphere centered at x with radius r_b. Two boundary points $L_{max} = (m_{max}, m_{max}, ..., m_{max})$ and $L_{min} = (m_{min}, m_{min}, ..., m_{min})$ can be obtained by projecting the hypersphere on l, where

$$\left.\begin{aligned} m_{max} &= m_x + \frac{r_b}{\sqrt{k}} \\ \text{and} \qquad m_{min} &= m_x - \frac{r_b}{\sqrt{k}} \end{aligned}\right\}. \qquad (2)$$

As shown in **Fig. 1** for 2-dimensional case the hypersphere can be bounded by two hyperplanes s_1 and s_2 with mean values m_{max} and m_{min}, respectively. Hence, just the codewords those have mean values between m_{min} and m_{max} will be searched. Lee and Chen [4] extended this work by introducing a new algorithm, which uses the variance of the vector as well as the mean value. The first part of this algorithm is the same as the ENNS algorithm, followed by calculating the squared root of variance, v_x, of the input vector x. From the geometrical interpretation in **Fig. 1**, v_x corresponds to the Euclidean distance $d(x,L_x)$ between x and its projection point L_x on l. If the mean value of each codeword y_i, m_{y_i}, is not between m_{min} and m_{max}, then y_i is rejected without calculating the $D = d^2(x,y_i)$. Next, following the next relations using triangle inequality,

$$\begin{aligned} d(x,y_i) &\geq d(x,L_{y_i}) - d(y_i,L_{y_i}) \\ &\geq d(x,L_x) - d(y_i,L_{y_i}) \\ &= v_x - v_{y_i} \text{ for } v_x \geq v_{y_i}, \end{aligned} \quad (3)$$

$$\begin{aligned} d(x,y_i) &\geq d(y_i,L_x) - d(x,L_x) \\ &\geq d(y_i,L_{y_i}) - d(x,L_x) \\ &= v_{y_i} - v_x \text{ for } v_x < v_{y_i}, \end{aligned} \quad (4)$$

and

$$D(x,y_i) = d^2(x,y_i) \geq (v_x - v_{y_i})^2, \quad (5)$$

if $(v_x - v_{y_i})^2 \geq D_{min}$ for a codeword y_i of which the mean value m_{y_i} is between m_{min} and m_{max}, y_i is rejected, otherwise D is calculated, where D_{min} is the current minimum distortion. If $D < D_{min}$, the current minimum distortion D_{min} is replaced by D and m_{min} and m_{max} are updated. Finally, the search area bounded by two hyperplanes s_1 and s_2 has been reduced to two dotted squares in **Fig. 1**.

As mentioned above, the search area is reduced to the two dotted squares. However, if the two search areas are separated, the search area will be reduced to one dotted square only and the computation complexity may be reduced to around half. To accomplish this target, we introduce a new algorithm that uses a hyperplane decision technique for separating the codebook and searches in one side area according to the input vector feature.

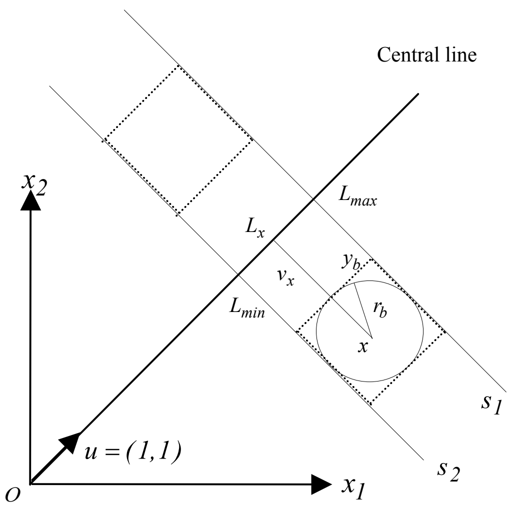

Fig. 1 Geometrical interpretation of the Lee & Chen method for 2-dimensional case.

3. Hyperplane decision method

The nearest codeword for an input vector belongs to one of two search areas shown in **Fig. 1**. If this relation is known before the codeword searching, the search area can be reduced. Although a perfect identification of the search area for all input vectors is difficult, a probable and reasonable separation of the codebook for image data may be possible when the codebook size is relatively large and its distribution in the signal space is smooth. In the following, one of such methods using the hyperplane decision rule is proposed.

3.1 Hyperplane equation

Considering the more reduction of search areas by Lee and Chen [4], the codebook is separated into two sub-groups by a hyperplane including the central line on it. Another condition for this hyperplane is that the centroid of input vectors is also on it, because the input vectors and their corresponding nearest codewords are likely in the same half-space separated by the hyperplane. This condition cannot satisfy the desired property strictly. However, there may be a small failure possibility in the case of large codebook size and smooth codebook distribution. As a result, a hyperplane including the origin o, the centroid of the input vectors $x_c = (x_{c1}, x_{c2}, \ldots, x_{ck})$ and the projection point of the centroid on the central line $x_p = (x_{p1}, x_{p2}, \ldots, x_{pk})$, where

$$x_{p1} = x_{p2} = \ldots = x_{pk} = \tfrac{1}{k}\textstyle\sum_{i=1}^{k} x_{c_i}$$

is chosen. There are many candidates for this hyperplane in k-dimensional space ($k > 3$) and a unique hyperplane cannot be defined. Among these hyperplanes, the simplest hyperplane for the inner product computation with input vectors, of which the only first three components are defined and the rest components are all zeros, is used. This hyperplane H is expressed as follows,

$$H(x) : hx^T = 0, \qquad (6)$$

where

$h = (x_{c3} - x_{c2}, x_{c1} - x_{c3}, x_{c2} - x_{c1}, 0, 0, \ldots, 0)$

is the normal vector to the hyperplane H.

3.2 Codebook division by the hyperplane

The hyperplane H is used as a decision function that discriminates to which half-space a given vector x belongs using the following conditions:

- If $hx^T < 0$, (7)

then x belongs to the lower half-space separated by H.

- If $hx^T \geq 0$, (8)

then x belongs to the upper half-space.

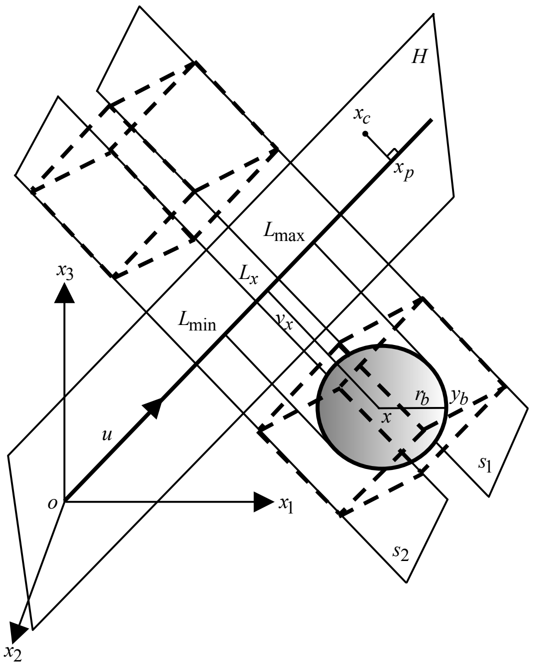

Fig. 2 Geometrical interpretation of the proposed method for 3-dimensional case.

Now we depict the proposed algorithm that uses this hyperplane H to separate the codebook. The algorithm divides the codebook Y into two sub-codebooks, Y_{lw} and Y_{up} (which contain the codewords that satisfy the equations (7) and (8), respectively). For a training vector x, the proposed algorithm determines the location of x from the equations (7) and (8) and then, decides which codebook (Y_{lw} or Y_{up}) will be searched for finding the closest codeword. Hence, the proposed algorithm can reduce the search area and speed up the search process than the Lee & Chen method [4]. **Fig. 2** depicts the geometric interpretation of the proposed algorithm for 3-dimenstional case. It is the extension of 2-dimenstional case in **Fig. 1** and includes the proposed hyperplane *H*. As we see in **Fig. 2**, the two search areas (which are represented by the two dotted cubes) are separated by the hyperplane *H* and reduced to one search area.

A detailed description of how to apply the proposed algorithm to design the codebook is given below:

Step 0: Initialization: Given N = codebook size, n = the number of training vectors, k = the vector dimension, Y_0 = initial codebook, ε = distortion threshold. Set iteration counter $e=0$, initial total distortion $D_{-1}=\infty$.

Step 1: Compute the centroid point, x_c, of the training vectors and the normal vector *h*. For all training vectors, compute their mean values and squared root of variances tables followed by computing the inner product table from the equation

$$h x_i^T, \quad i=1,\dots,n, \qquad (9)$$

Step 2: Separate the codebook Y_e into two sub-codebooks Y_{lw} and Y_{up} with sizes N_1 and N_2 $(N_1+N_2=N)$ respectively, according to the equations (7) and (8).

Step 3: Compute the mean value of each codeword in the codebooks Y_{lw} and Y_{up}. Sort each codebook according to increasing order of the codeword means, i.e. the sorted codebooks become Y_{slw} and Y_{sup}. Compute the squared root of the variance v_{y_i} of each codeword y_i in every codebook Y_{slw} and Y_{sup}.

Step 4: For each training vector *x*, check its location with respect to *H* from the inner product table that is computed in step 1: If $h x^T \geq 0$, go to step 6, otherwise go to the next step.

Step 5: Find the closest codeword y_j from the codebook Y_{slw} and assign *x* to class *j*. The procedure includes the following substeps:

Step 5.1: For the input vector *x*, find the closest codeword y_b that has the minimum mean difference to *x* (using binary search), i.e.

$$\left|m_x - m_{y_b}\right| \leq \left|m_x - m_{y_i}\right|, \text{ for all } i \neq b$$

Set $D_{min} = d_{min}^2 = d^2(x, y_b),\quad j=b,$

$$m_{max} = m_x + \frac{d_{min}}{\sqrt{k}}, \quad m_{min} = m_x - \frac{d_{min}}{\sqrt{k}}.$$

Step 5.2: Find the closest codeword y_j in Y_{slw} and assign *x* to class *j*. The search procedure is as follows:

```
Set $z = 1$
while ($m_{b+z} < m_{max}$ or $m_{b-z} > m_{min}$) begin
  if ($m_{b+z} < m_{max}$) begin
    if ($(v_x - v_{y_{b+z}})^2 < D_{min}$) begin
      if ($d^2(x, y_{b+z}) < D_{min}$) begin
        $D_{min} = d_{min}^2 = d^2(x, y_{b+z})$
        $m_{max} = m_x + \frac{d_{min}}{\sqrt{k}}$
        $m_{min} = m_x - \frac{d_{min}}{\sqrt{k}}$
        $j = b + z$
      end
    end
  end
  if ($m_{b-z} > m_{min}$) begin
    if ($(v_x - v_{y_{b-z}})^2 < D_{min}$) begin
      if ($d^2(x, y_{b-z}) < D_{min}$) begin
        $D_{min} = d_{min}^2 = d^2(x, y_{b-z})$
        $m_{max} = m_x + \frac{d_{min}}{\sqrt{k}}$
        $m_{min} = m_x - \frac{d_{min}}{\sqrt{k}}$
        $j = b - z$
      end
    end
  end
  $z = z + 1$
end {of while}
go to step 7.
```

Step 6: Find the closest codeword y_j from Y_{sup} and assign *x* to class *j* as the procedure in step 5.

Step 7: Compute the total distortion for *e*th iteration D_e.

	Lena		Baboon	
	Codebook design	Encoding	Codebook design	Encoding
Full search	322 (14.6)	27 (15.9)	215 (4.2)	27 (4.9)
Lee & Chen method	22 (1)	1.7 (1)	51 (1)	5.5 (1)
Proposed method	11.7 (0.53)	0.9 (0.53)	28 (0.55)	3 (0.54)

Table 1 Comparison of execution time (in seconds) for codebook design and image encoding at codebook size 256.

	Lena		Baboon	
	Codebook design	Encoding	Codebook design	Encoding
Full search	50331648	4194304	33554432	4194304
Lee & Chen method	3360652	257910	7881386	850562
Proposed method	1719409	125046	4260634	452592

Table 2 Comparison of the total number of distortion calculations for codebook design and image encoding at codebook size 256.

	Lena		Baboon	
	Lower half-space	Upper half-space	Lower half-space	Upper half-space
$N = 256$	132	124	134	122
n =16384	8719	7665	8396	7988

Table 3 The number of codewords(N) and training vectors(n) in each half-space separated by the hyperplane H at codebook size 256.

Step 8: If $(D_{e-1} - D_e)/D_e \leq \varepsilon$ halt with the final codebook Y_e, otherwise go to step 9.
Step 9: Compute the centroid of each class. Set $e = e + 1$ and go to step 2.

4. Experimental results

Experiments were carried on vectors taken from the USC grayscale image set. We used two images, Lena and Baboon with size 512 × 512 and 256 gray levels. Each image is divided into 4 × 4 blocks, so that the training set contains 16384 blocks, and the codebook size is 256. The tested methods are the full search, the Lee & Chen method[4)] and our proposed method. **Table 1** presents the time execution for these methods. The values in the parentheses denote the ratio of execution time of full search and the proposed method to that of Lee & Chen method. The timings were made on Pentium II (267.3MHZ). We can see that our new method significantly accelerates the codebook design and image encoding. Compared to the Lee & Chen method, our method reduces the time by factor of 1.88 for Lena and Baboon. **Table 2** displays the total number of distortion calculations in codebook design and image encoding for Lena and Baboon, which is a dominant figure of computational complexity. Compared to the Lee & Chen method, our proposed method reduces the total number of distortion calculations by 48.8% (for Lena) and 45.9% (for Baboon) in codebook design, while by 51.5% (for Lena) and 46.8% (for Baboon) in image encoding. **Table 3** shows that the number of codewords(N) and the number of

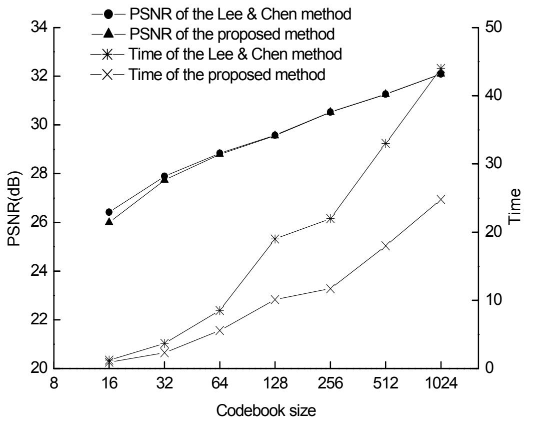

Fig. 3 Comparison between the proposed method and lee & Chen method for Lena image.

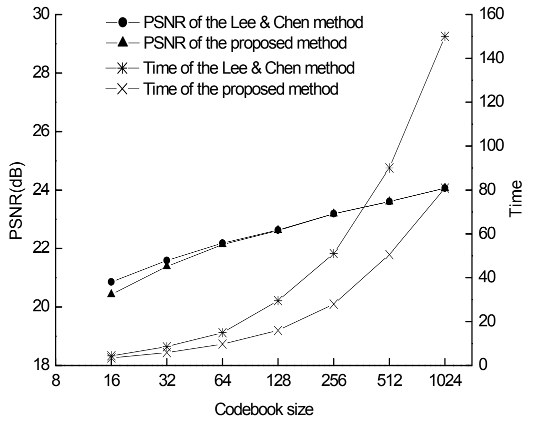

Fig. 4 Comparison between the proposed method and lee & Chen method for Baboon image.

training vectors(n) in each half-space separated by the hyperplane H are quite close. **Fig. 3** and **Fig. 4** show the PSNR and the execution time of the codebook design by the proposed method and the Lee & Chen method at different codebook sizes for both Lena and Baboon images, respectively. The proposed method has almost the same performance of the Lee & Chen method for codebook sizes from 64 to more than this size. There is small degradation in the proposed method for smaller codebook sizes, for example 32 or below. As mentioned before, this is because the proposed method is not equivalent to the Lee & Chen method completely, and the best codeword happens to be in the other search area and is missed to be searched out. However there may be a small failure possibility in the case of large codebook size and smooth codebook distribution. Our method has only 0.01 dB less than the Lee & Chen method for Lena and Baboon at the codebook size 256 and this value decreases by increasing the codebook size.

5. Conclusion

In this paper, we have presented a new method of accelerating the codebook design and image encoding for standard VQ. The proposed method uses a hyperplane decision technique for separating the codebook into two search areas, and then searches in one area according to the input vector feature. Compared with the Lee & Chen method, the obtained results show that the proposed method allowed acceleration ratios of at least 1.88 and reduced the total number of distortion calculations by 45.9%. Furthermore, the performance of the proposed method was quite closer to the performance of the Lee & Chen method in the case of the codebook size more than 64. The extension of our algorithm to Entropy-constrained vector quantization is a further study.

References

[1] Y. Linde, A. Buzo and R. M. Gray, “An algorithm for vector quantizer design”, IEEE Trans. Commun., **COM-28**, pp. 84-95, (Jan. 1980).

[2] Special issue on vector quantization, IEEE Trans. Image Processing, **5**, 2, (Feb. 1996).

[3] L. Guan and M. Kamel, “Equal-average hyperplane partitioning method for vector quantization of image data”, Pattern Recognition Lett., **13**, 10, pp. 693-699, (Oct. 1992).

[4] C.-H. Lee, L.-H. Chen, “Fast closest codeword search algorithm for vector quantization”, IEE Proc. Vis. Image Signal Process., **141**, 3, pp. 143-148, (June 1994).

[5] M. D. Orchard, “A fast nearest-neighbor search algorithm”, IEEE ICASSP, pp. 2297-2300, (1991).

[6] C.-H. Hsieh and Y.-J. Liu, “Fast search algorithms for vector quantization of images using multiple triangle inequalities and wavelet transform”, IEEE Trans. Image Processing, **9**, 3, pp. 321-328, (Mar. 2000).

[7] K.-S. Wu and J.-C. Lin, “Fast VQ encoding by an efficient kick-out condition”, IEEE Trans. Circuits Syst. Video Technol., **10**, 1, pp. 59-62, (Feb. 2000).