A Fast Search Algorithm for Vector Quantization Using Means and Variances of Codewords

Chang-Hsing Lee and Ling-Hwei Chen

Department of Computer and Information Science, National Chiao Tung University, Hsinchu, Taiwan 30050, R. O. C.

ABSTRACT

Vector Quantization has been applied to low-bit-rate speech and image compression. One of the most serious problems for vector quantization is the high computational complexity of searching for the closest codeword in the codebook design and encoding processes. To overcome this problem, a fast algorithm, under the assumption that the distortion is measured by the squared Euclidean distance, will be proposed to search for the closest codeword to an input vector. Using the means and variances of codewords, the algorithm can reject many codewords that are impossible to be candidates for the closest codeword to the input vector and hence save a great deal of computation time. Experimental results confirm the effectiveness of the proposed method.

1. INTRODUCTION

Vector quantization (VQ) is a very efficient approach to low-bit-rate image compression[1-2]. In VQ, the images to be encoded are first decomposed into vectors (i.e., blocks) and then sequentially encoded vector by vector. Each vector is compared with the codewords in the codebook to find the best matching codeword. The key aspect of VQ is to design a good codebook, $\mathbf{Y}=\{\mathbf{y}_i \mid i=1, 2, \ldots, N\}$, which contains the most representative codewords and will be used by the encoder and the decoder. In the encoding process, the encoder designs a mapping Q and assigns an index i to each k-dimensional input vector $\mathbf{x}=(x_1, x_2, \ldots, x_k)$, with $Q(\mathbf{x})=\mathbf{y}_i=(y_{i1}, y_{i2}, \ldots, y_{ik})$. In this paper, we will only consider the mapping Q, which is designed to map $\mathbf{x}$ to $\mathbf{y}_i$ with $\mathbf{y}_i$ satisfying the following condition:

$$d^2(\mathbf{x}, \mathbf{y}_i) = \min_{j} d^2(\mathbf{x}, \mathbf{y}_j), \quad \text{for } j = 1, 2, \ldots, N, \tag{1}$$

where $d^2(\mathbf{x}, \mathbf{y}_j)$ is the distortion of representing the input vector $\mathbf{x}$ by the codeword $\mathbf{y}_j$, and is measured by the squared Euclidean distance, i.e.,

$$d^2(\mathbf{x}, \mathbf{y}_j) = \sum_{n=1}^{k} (x_n - y_{jn})^2. \tag{2}$$

The decoder has the same codebook as the encoder. In the decoding process, for each index i, the decoder merely performs a simple table look-up operation to obtain $\mathbf{y}_i$ and then uses $\mathbf{y}_i$ to

0-8194-1858-7/95/$6.00

reconstruct the input vector **x**. Compression is achieved by transmitting or storing the index of a codeword rather than the codeword itself.

From the above description, we see that the compression ratio of VQ is determined by the codebook size and the dimension of the input vectors, and the distortion is dependent on the codebook size and the selection of codewords. Hence, a good codebook design is the main task of VQ. Many algorithms for optimal codebook design have been proposed[3-7]. Among these, the most popular was developed by Linde, Buzo and Gray[3] and is referred to as the LBG algorithm. This algorithm is basically an iterative process to minimize the overall distortion of representing the training vectors by their corresponding codewords. Since a full codebook search is needed to find the closest codeword for each training vector, the algorithm is time consuming. To reduce the computation time needed for such an exhaustive search through the codebook, many fast algorithms have been proposed[8-22]. Some of them achieve the goal of decreasing the search time at the expense of the coding quality. Some can reduce the search time without producing any extra error, but are still not fast enough. In the following section, we will introduce such an algorithm.

2. PREVIOUS WORKS

Guan *et al.*[21] proposed an equal-average nearest neighbor search (ENNS) algorithm which uses hyperplanes orthogonal to the central line l to partition the search space. Each coordinate value of any point p=(p_1, p_2, ..., p_k) on l has the same value, i.e., $p_i = p_j$, $i, j = 1, 2, ..., k$. Each point on a fixed hyperplane H, which is orthogonal to the central line l and intersects l at point L_H=(m_H, m_H, ..., m_H), will have the same mean value m_H, such a hyperplane is called an equal-average hyperplane. For an input vector **x**=(x_1, x_2, ..., x_k), the algorithm first calculates its mean value m_x with

$$m_x = \frac{1}{k} \sum_{j=1}^{k} x_j.$$

It then finds the codeword $\mathbf{y}_p$ which has the minimum mean difference from **x** and calculates the distance r_p between **x** and $\mathbf{y}_p$. It is obvious that any other codeword which is closer to **x** than $\mathbf{y}_p$ has to be located inside the hypersphere centered at **x** with radius r_p. By projecting the hypersphere on l, two boundary projection points, L_{max}=(m_{max}, m_{max}, ..., m_{max}) and L_{min}=(m_{min}, m_{min}, ..., m_{min}) on l can be found, where

$$m_{max} = m_x + \frac{r_p}{\sqrt{k}}, \tag{3}$$

and

$$m_{min} = m_x - \frac{r_p}{\sqrt{k}}. \tag{4}$$

The hypersphere is bounded by two equal-average hyperplanes with mean values m_{max} and m_{min}. Hence, the algorithm only searches those codewords with mean values ranging from m_{min} to m_{max}. Fig. 1 shows the geometric interpretation of the method for a 2-dimensional case, the

search area is bounded by two lines l_1 and l_2, which are perpendicular to the central line l at L_{max} and L_{min}, respectively.

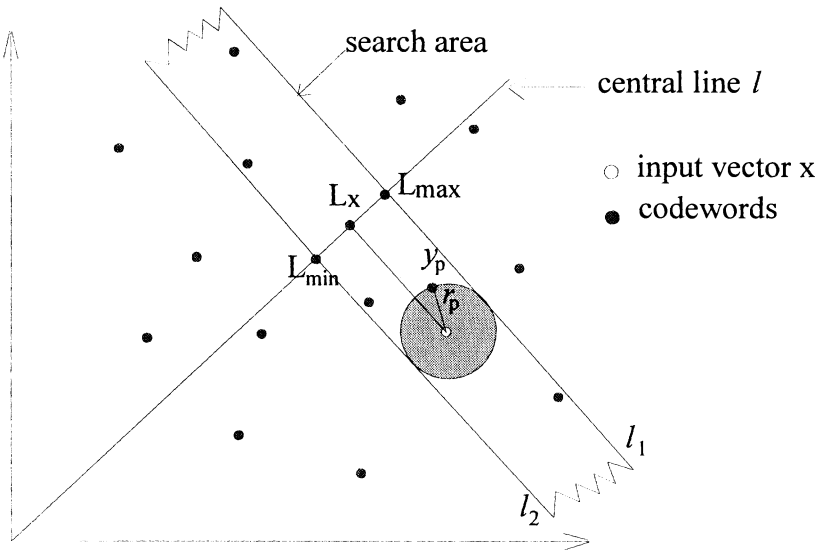

Fig. 1 An example of the ENNS algorithm for a 2-dimensional case.

As described above, the ENNS algorithm uses mean value as a feature to reject unlikely codewords and thus saves much of computation time. However, two vectors with similar mean values may have a large distance. For example, one vector represents an almost homogeneous block, i.e., entities in the vector are almost the same; the other represents an edge block, i.e., some entities will be larger than others. The distance between these two types of vectors will be large. To treat this phenomenon, we have proposed a new search method[22], called the mean-or-variance (MOV) method, to reduce the search area of the ENNS algorithm. Since the variance of a vector is a simple measure to detect whether a vector is homogeneous, this method uses both the mean value and the variance of a vector as two significant features to reject many unlikely codewords. Define the mean value and variance of a k-dimensional vector $\mathbf{x}$ as

$$m_x = \frac{1}{k} \sum_{j=1}^{k} x_j,$$

and

$$V_x^2 = \sum_{j=1}^{k} (x_j - m_x)^2.$$

We have proved[22] that for a codeword $\mathbf{y}_i$, the distortion between $\mathbf{x}$ and $\mathbf{y}_i$, $d^2(\mathbf{x}, \mathbf{y}_i)$, will satisfy

$$d^2(\mathbf{x}, \mathbf{y}_i) \geq k(m_x - m_{yi})^2 \tag{5}$$

and

$$d^2(\mathbf{x}, \mathbf{y}_i) \geq (V_x - V_{yi})^2. \tag{6}$$

Therefore, if d^2_{min} is a known current minimum distortion of **x** represented by a certain codeword. For any codeword $\mathbf{y}_i$, if $k(m_x - m_{yi})^2 \geq d^2_{min}$ or $(V_x - V_{yi})^2 \geq d^2_{min}$, then $\mathbf{y}_i$ will not be the closest codeword to **x** and it is unnecessary to calculate $d^2(\mathbf{x}, \mathbf{y}_i)$. The MOV method was developed according to the above idea. Fig. 2 depicts the geometric interpretation of the MOV method for a 2-dimensional case. Comparing Fig. 1 and Fig. 2, we can see that the search area, which is originally an area bounded by two lines l_1 and l_2 perpendicular to the central line l, has been reduced to be the two shaded squares. In the next section, we will describe the proposed algorithm, which is faster than the MOV method.

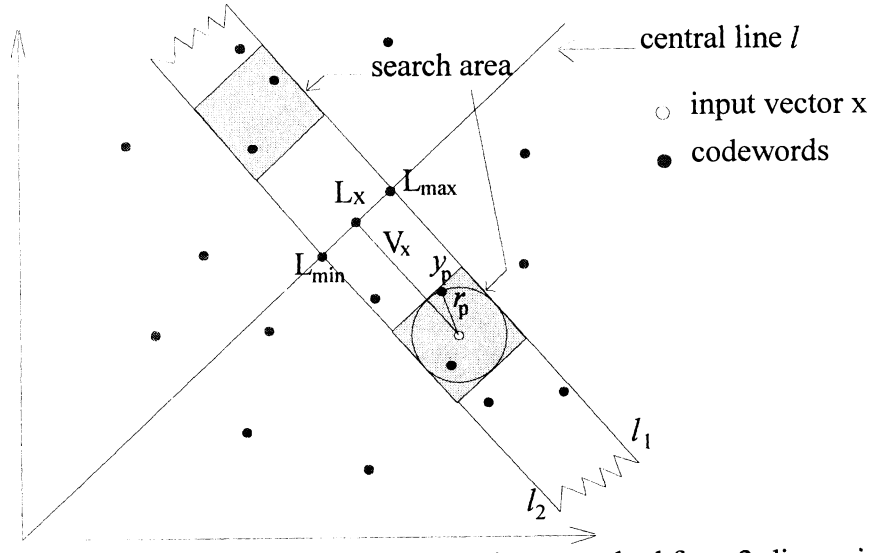

Fig. 2 An example to illustrate the mean-or-variance method for a 2-dimensional case.

3. THE MEAN-AND-VARIANCE METHOD

In this section, we will propose another inequality, which can reject more codewords than the MOV method, to speed up the closest codeword search process. This method, called the mean-and-variance (MAV) method, also uses the mean value and variance of a vector as two significant features. In the MOV method, if d^2_{min} is a known current minimum distortion of **x** represented by a certain codeword. For any codeword $\mathbf{y}_i$, if $k(m_x - m_{yi})^2 \geq d^2_{min}$ or $(V_x - V_{yi})^2 \geq d^2_{min}$, then $\mathbf{y}_i$ will not be the closest codeword to **x** and it is unnecessary to calculate $d^2(\mathbf{x}, \mathbf{y}_i)$. However, we have proved that the following inequality is true:

$$d^2(\mathbf{x}, \mathbf{y}_i) \geq (V_x - V_{yi})^2 + k(m_x - m_{yi})^2. \tag{7}$$

(The proof of Inequality (7) is given in Appendix A.) Therefore, those codewords with $(V_x - V_{yi})^2 + k(m_x - m_{yi})^2$ larger than the known current minimum distortion d^2_{min} will be rejected. Comparing Inequalities (5), (6) and (7), we can see that the new proposed method can reject more

codewords and thus without evaluating $d^2(\mathbf{x}, \mathbf{y}_i)$, this makes the algorithm to be more efficient. Fig. 3 depicts the geometric interpretation of the MAV method for a 2-dimensional case. Comparing Fig. 2 and Fig. 3, we can see that the search area, which is originally the two shaded squares, has been reduced to be the two shaded circles.

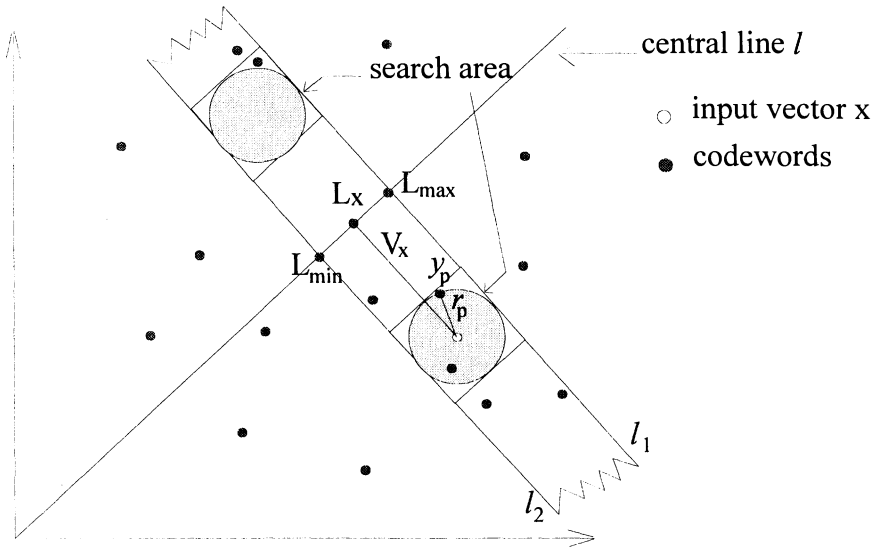

Fig. 3 An example to illustrate the mean-and-variance method for a 2-dimensional case.

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

Step 0 Initialization: Given N = codebook size, n = the number of training vectors, k = the vector dimension, $\mathbf{Y}_0$ = initial codebook, ε = distortion threshold. Set iteration counter c = 0, initial total distortion $D_{-1} = \infty$.

Step 1 Compute the mean value of each codeword in the codebook $\mathbf{Y}_c$, and sort $\mathbf{Y}_c$ according to increasing order of the codeword means, i.e., the sorted codebook $\mathbf{Y}_{sc}$ is

$$\mathbf{Y}_{sc} = \{\mathbf{y}_i \mid m_{yi} \le m_{y(i+1)},\ 1 \le i \le N\text{-}1\},$$

where m_{yi} is the mean value of the codeword $\mathbf{y}_i$.

Step 2 Compute the squared root of the variance, V_{yi}, of each codeword $\mathbf{y}_i$.

Step 3 For each training vector $\mathbf{x}_t$, find the closest codeword $\mathbf{y}_{i(t)}$ in the codebook $\mathbf{Y}_{sc}$ and assign $\mathbf{x}_t$ to class i(t). The procedure includes the following sub-steps:

Step 3.1 Input a training vector $\mathbf{x}_t=(x_{t1}, x_{t2}, ..., x_{tk})$, compute its mean value m_{xt} and its square root of variance V_{xt}.

Step 3.2 Find the codeword $\mathbf{y}_p$ which has the minimum mean difference from $\mathbf{x}_t$ (using binary search), i.e.,

$$|m_{xt} - m_{yp}| \le |m_{xt} - m_{yi}|, \text{ for all } i \ne p.$$

Set $i(t) = p$, $d^2_{min} = d^2(\mathbf{x}_t, \mathbf{y}_p)$.

Step 3.3 Find the closest codeword $\mathbf{y}_{i(t)}$ in $\mathbf{Y}_{sc}$ and assign $\mathbf{x}_t$ to class i(t). The search procedure is as follows:

Set d = 1;

```
while((p+d ≤ N and k(m_xt - m_y(p+d))^2 < d^2_min ) or
      (p-d ≥ 1 and k(m_xt - m_y(p-d))^2 < d^2_min )) begin
   if(p+d ≤ N and k(m_xt - m_y(p+d))^2 < d^2_min ) begin
      if(k(m_xt - m_y(p+d))^2 + (V_xt - V_y(p+d))^2 < d^2_min ) begin
         if( d^2(x_t, y_p+d) < d^2_min ) begin
            d^2_min = d^2(x_t, y_p+d);
            i(t) = p+d;
         end;
      end;
   end;
   if(p-d ≥ 1 and k(m_xt - m_y(p-d))^2 < d^2_min ) begin
      if(k(m_xt - m_y(p-d))^2 + (V_xt - V_y(p-d))^2 < d^2_min ) begin
         if( d^2(x_t, y_p-d) < d^2_min ) begin
            d^2_min = d^2(x_t, y_p-d);
            i(t) = p-d;
         end;
      end;
   end;
end; {of while}
```

Step 4 Compute the overall distortion for the *c-th* iteration, D_c. Here D_c is defined to be

$$D_c = \sum_{t=1}^{n} d^2(\mathbf{x}_t, \mathbf{y}_{i(t)}).$$

Step 5 If $(D_{c-1} - D_c)/D_c \leq \varepsilon$, halt with final codebook being $\mathbf{Y}_{sc}$. Otherwise, go to *Step 6.*

Step 6 Compute the centroid of each class. The centroids are regarded as the codewords of a new codebook. Set c = c + 1 and go to *Step 1* for next iteration.

The encoder has to find the closest codeword in a predesigned codebook for each input vector and then uses the codeword as the reproduction one of the corresponding input vector. Therefore, it can use the MAV to find the closest codeword to each input vector. The details of this procedure are similar to those in *Step 3* of the codebook design algorithm described above.

4. EXPERIMENTAL RESULTS

To examine the efficiency of the proposed algorithm, we performed some experiments on a Sun SPARC-station-IPC using several 512×512 monochrome images with 256 gray levels. Each image is divided into 4×4 blocks, so that the training sequence contains 16384 16-dimensional vectors. The proposed algorithm was compared with the LBG algorithm and the MOV

algorithm[22] in terms of the execution time required in codebook design and image encoding. Table 1 shows the execution time required to design a codebook using the well-known image Lena. Table 2 shows the time needed to encode an image given the previously designed codebook. The codebook was used to encode the four images (Lena, Peppers, Jet, and Baboon). From these two tables, we see the effectiveness of the proposed algorithm in both codebook design and image encoding.

Table 1
Comparison of execution time (in seconds) for codebook design. Values in parentheses denote the ratio of execution time of the current algorithm to that of the LBG algorithm.

Codebook size	LBG	MOV	MAV
128	2815	266(0.094)	236(0.084)
256	5608	383(0.068)	337(0.060)
512	11263	576(0.051)	506(0.045)
1024	22732	925(0.041)	820(0.036)

Table 2
Comparison of execution time (in seconds) for image encoding.

Codebook size	Method	Encoded image			
		Lena	Peppers	Jet	Baboon
128	Full search	140.9	140.1	125.1	137.5
	MOV	11.5(0.082)	11.9(0.085)	11.4(0.091)	32.2(0.233)
	MAV	10.9(0.077)	11.0(0.079)	10.5(0.084)	24.8(0.180)
256	Full search	278.6	279.5	249.7	275.0
	MOV	17.2(0.062)	18.8(0.067)	17.4(0.070)	54.1(0.197)
	MAV	15.6(0.056)	16.4(0.059)	15.0(0.060)	43.0(0.156)
512	Full search	557.6	558.1	499.3	547.8
	MOV	26.7(0.048)	30.2(0.054)	26.7(0.053)	96.2(0.176)
	MAV	23.6(0.042)	26.2(0.047)	23.2(0.046)	76.3(0.139)
1024	Full search	1108.5	1108.5	992.7	1087.9
	MOV	40.7(0.037)	51.1(0.046)	43.6(0.044)	174.6(0.160)
	MAV	37.3(0.034)	44.6(0.040)	38.5(0.039)	139.3(0.128)

5. CONCLUSIONS

In this paper, we have proposed a fast closest codeword search algorithm, called the MAV algorithm, for vector quantization. This algorithm uses two significant features of a vector, mean value and variance, to reject a lot of codewords which are definitely not candidates for the closest codeword to the input vector. It can speed up the search process in conventional VQ codebook

design and encoding. The performance of the proposed algorithm has been evaluated in both codebook design and image encoding. The results obtained show that the proposed algorithm outperforms the ENNS and the MOV algorithms and reduces a great deal of computation time required by the LBG algorithm. Furthermore, it is worth mentioning that the proposed algorithm does not introduce any extra error than the LBG algorithm.

6. ACKNOWLEDGMENT

This research was supported in part by the National Science Council of R. O. C. under contract NSC-83-0404-E009-117.

7. REFERENCES

[1] R. M. Gray, "Vector quantization," *IEEE ASSP Magazine*, vol. 1, pp. 4-29, Apr. 1984.

[2] N. M. Nasrabadi and R. A. King, "Image coding using vector quantization: A review," *IEEE Trans. Commun.*, vol. COM-36, no. 8, pp. 957-971, Aug. 1988.

[3] Y. Linde, A. Buzo, and R. M. Gray "An algorithm for vector quantizer design," *IEEE Trans. Commun.*, vol. COM-28, no. 1, pp. 84-95, Jan. 1980.

[4] W. H. Equitz, "A new vector quantization clustering algorithm," *IEEE Trans. Acoust., Speech, and Signal Processing,* vol. 37, no. 10, pp. 1568-1575, Oct. 1989.

[5] B. Marangelli, "A vector quantizer with minimum visible distortion," *IEEE Trans. Signal Processing*, vol. 39, no. 12, pp. 2718-2721, Dec. 1991.

[6] K. Zeger, J. Vaisey, and A. Gersho, "Globally optimal vector quantizer design by stochastic relaxation," *IEEE Trans. Signal Processing*, vol. 40, no. 2, pp. 310-322, Feb. 1992.

[7] K. Rose, E. Gurewitz, and G. C. Fox, "Vector quantization by deterministic annealing," *IEEE Trans. Inform. Theory*, vol. IT-38, no. 4, pp. 1249-1257, Jul. 1992.

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

[9] K. K. Paliwal and V. Ramasubramanian, "Effect of ordering the codebook on the efficiency of the partial distance search algorithm for vector quantization," *IEEE Trans. Commun.*, vol. COM-37, no. 5, pp. 538-540, May. 1989.

[10] C. H. Hsieh, P. C. Lu, and J. C. Chang "Fast codebook generation algorithm for vector quantization of images," *Pattern Recognition Letters*, vol. 12, pp. 605-609, 1991.

[11] D. Y. Cheng, A. Gersho, B. Ramamurthi, and Y. Shoham, "Fast search algorithms for vector quantization and pattern matching," in *Proc. IEEE ICASSP*, 1984, pp. 9.11.1-9.11.4.

[12] D. Y. Cheng and A. Gersho "A fast codebook search algorithm for nearest-neighbor pattern matching," in *Proc. IEEE ICASSP*, 1986, pp. 265-268.

[13] A. Lowry, S. Hossain, and W. Millar, "Binary search trees for vector quantization," in *Proc. IEEE ICASSP*, 1987, pp. 2205-2208.

[14] V. Ramasubramanian and K. K. Paliwal, "An optimized k-d tree algorithm for fast vector quantization of speech," in *Proc. European Signal Processing Conf.*, 1988, pp. 875-878.

[15] V. Ramasubramanian and K. K. Paliwal, "Fast k-dimensional tree algorithms for nearest neighbor search with application to vector quantization encoding," *IEEE Trans. Signal Processing*, vol. 40, no.3, pp. 518-531, Mar. 1992.
[16] M. R. Soleymani and S. D. Morgera, "A high-speed search algorithm for vector quantization," in *Proc. IEEE ICASSP*, 1987, pp. 45.6.1-3.
[17] V. Ramasubramanian and K. K. Paliwal, "An efficient approximation-elimination algorithm for fast nearest-neighbor search based on a spherical distance coordinate formulation," *Pattern Recognition Letters*, vol. 13, pp. 471-480, 1992.
[18] E. Vidal, "An algorithm for finding nearest neighbors in (approximately) constant average time complexity," *Pattern Recognition Letters*, vol. 4, pp. 145-157, 1986.
[19] M. T. Orchard, "A fast nearest-neighbor search algorithm," in *Proc. IEEE ICASSP*, 1991, pp. 2297-2300.
[20] C. M. Huang, Q. Bi, G. S. Stiles, and R. W. Harris, "Fast full search equivalent encoding algorithms for image compression using vector quantization," *IEEE. Trans. Image Processing*, vol.1, no. 3, pp. 413-416, Jul. 1992.
[21] L. Guan and M. Kamel, "Equal-average hyperplane partitioning method for vector quantization of image data," *Pattern Recognition Letters*, vol. 13, pp. 693-699, 1992.
[22] C. H. Lee and L. H. Chen, "Fast closest codeword search algorithm for vector quantization," *IEE Proc.- Vision, Image and Signal Processing*, vol. 141, no. 3, June 1994, pp. 143-148.

APPENDIX A

Proof of Inequality (7): To prove Inequality (7), we have to prove the following inequality first:

$$V_x V_{yi} \geq \sum_{j=1}^{k} (x_j - m_x)(y_{ij} - m_{yi}). \tag{A.1}$$

From the well-known Cauchy-Schwarz Inequality, we can easily get

$$(V_x V_{yi})^2 = \sum_{j=1}^{k} (x_j - m_x)^2 \times \sum_{j=1}^{k} (y_{ij} - m_{yi})^2 \geq [\sum_{j=1}^{k} (x_j - m_x)(y_{ij} - m_{yi})]^2.$$

From the above inequality, we can derive Inequality (A.1).

Since $\sum_{j=1}^{k} (x_j - y_{ij})^2 = \sum_{j=1}^{k} (x_j - m_x + m_x - m_{yi} + m_{yi} - y_{ij})^2$

$$= \sum_{j=1}^{k} (x_j - m_x)^2 + \sum_{j=1}^{k} (m_x - m_{yi})^2 + \sum_{j=1}^{k} (m_{yi} - y_{ij})^2 +$$

$$2 \sum_{j=1}^{k} (x_j - m_x)(m_x - m_{yi}) + 2 \sum_{j=1}^{k} (m_x - m_{yi})(m_{yi} - y_{ij}) +$$

$$2\sum_{j=1}^{k}(m_{yi} - y_{ij})(x_j - m_x)$$

$$= (V_x)^2 + k(m_x - m_{yi})^2 + (V_{yi})^2 + 2(m_x - m_{yi})\sum_{j=1}^{k}(x_j - m_x) +$$

$$2(m_x - m_{yi})\sum_{j=1}^{k}(m_{yi} - y_{ij}) - 2\sum_{j=1}^{k}(x_j - m_x)(y_{ij} - m_{yi})$$

$$= (V_x)^2 + (V_{yi})^2 + k(m_x - m_{yi})^2 - 2\sum_{j=1}^{k}(x_j - m_x)(y_{ij} - m_{yi}). \quad (A.2)$$

And since $(V_x - V_{yi})^2 + k(m_x - m_{yi})^2 = (V_x)^2 + (V_{yi})^2 + k(m_x - m_{yi})^2 - 2V_x V_{yi}$. (A.3)
Comparing Eqs. (A.2) and (A.3) and from Inequality (A.1), we can prove Inequality (7).