ELSEVIER

Signal Processing 43 (1995) 323–331

SIGNAL PROCESSING

# High-speed closest codeword search algorithms for vector quantization[☆]

Chang-Hsing Lee, Ling-Hwei Chen*

*Department of Computer and Information Science, National Chiao Tung University, 1001 Ta Hsueh Rd., Hsinchu, Taiwan 30050, Republic of China*

Received 2 October 1992; revised 13 September 1993, 14 July 1994 and 15 December 1994

**Abstract**

One of the most serious problems for vector quantization is the high computational complexity involved in searching for the closest codeword through a codebook in both codebook design and encoding phases. In this paper, based on the assumption that the distortion is measured by the squared Euclidean distance, two high-speed search methods will be proposed to speed up the search process. The first one uses the difference between the mean values of two vectors to reduce the search space. The second is to find the Karhunen–Loeve transform (KLT) for the distribution of the set of training vectors and then applies the partial distortion elimination method to the transformed vectors. Experimental results show that the proposed methods can reduce lots of mathematical operations.

**Zusammenfassung**

Eines der schwerstwiegenden Probleme bei der Vektorquantisierung ist die hohe rechnerische Komplexität, die mit der Suche des nächstliegenden Kodewortes innerhalb eines Kodebuches verbunden ist, sowohl beim Entwurf des Kodebuches, als auch bei der Kodierphase. In dieser Arbeit werden unter der Voraussetzung des quadriatischen Euklidischen Abstandes als Verzerrungsmaß zwei sehr schnelle Suchmethoden vorgeschlagen, um den Suchprozeß zu beschleunigen. Die erste Methode verwendet die Differenz zwischen den Mittelwerten zweier Vektoren, um den Suchraum zu verkleinern. Die zweite Methode besteht darin, zunächst die Karhunen–Loeve Transformation (KLT) für die Verteilung der Menge der Trainingsvektoren zu finden. Danach wird die partielle Verzerrungseliminierungsmethode auf die transformierten Vektoren angewendet. Experimentelle Ergebnisse zeigen, daß die vorgeschlagenen Methoden die Anzahl der mathematischen Operationen erheblich vermindern können.

**Résumé**

L'un des problèmes les plus sérieux dans la quantification vectorielle est la grande complexité de calcul nécessaire pour chercher le mot le plus proche dans un dictionnaire, ceci dans les deux phases de réalisation du dictionnaire et d'encodage. Dans cet article, deux méthodes de recherche rapide, basées sur la supposition que la distortion est mesurée par la distance euclidienne quadratique, sont proposées pour accélerer le processus de recherche. La première utilise la

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

* Corresponding author.

différence entre les valeurs moyennes de deux vecteurs pour réduire l'espace de recherche. La seconde consiste à trouver la transformée de Karhunen–Loeve (TKL) pour la distribution de l'ensemble des vecteurs d'apprentissage et ensuite applique la méthode d'élimination partielle de distorsion aux vecteurs transformés. Les résultats expérimentaux montrent que les méthodes proposées peuvent réduire un grand nombre d'opérations mathématiques.

*Keywords*: Codebook design; Karhunen–Loeve transform; Vector quantization

## 1. Introduction

Vector quantization (VQ) is a well-known technique for low-bit-rate image compression [6, 7, 13]. In VQ, the images are first decomposed into vectors (i.e., blocks) and then sequentially encoded vector by vector. The aim of VQ is to find the best matching codeword in the codebook, and the key aspect of VQ is to design a good codebook containing the most representative codewords. In the encoding process, the index of the codeword that is the closest one to the input vector is transmitted or stored. The decoder uses the index to reconstruct the representative codeword. Compression is achieved by transmitting or storing the index of a codeword rather than the codeword itself.

Linde et al. [11] proposed a clustering algorithm, which is referred to as the LBG algorithm, for VQ codebook design. The algorithm is an iterative process that minimizes the overall distortion of representing the training vectors by their corresponding closest codewords. However, the LBG algorithm needs a great deal of computation time to do exhaustive search for the closest codeword to a training vector. To avoid this kind of search, many fast algorithms have been proposed. Among them, the partial distortion elimination method [1], the partial search partial distortion method [8], the Voronoi cells method [2, 3], the *k–d* tree method [12, 15, 16], and the triangle inequality method [9, 14, 17–19] are more popular. However, many of these algorithms achieve the goal of decreasing the search time at the expense of the coding quality.

Equitz proposed another codebook design algorithm, called the pairwise nearest-neighbor (PNN) clustering algorithm [5], to accelerate the codebook design process. This algorithm begins with a separate cluster for each training vector and merges the two clusters that have the smallest distance at a time until the desired codebook size or cluster number is achieved. This algorithm significantly reduces computational complexity, but the resulting codebook is suboptimal.

Since the most effort of searching for the closest codeword to a vector is to evaluate the distance between two vectors, if those remote codewords can be rejected before evaluating their distances from the input vector, the search process will be sped up. Based on this idea, two fast search methods can be proposed under the assumption that the distortion is measured by the squared Euclidean distance. The first one uses some elimination rules, which are based on the difference between the means of two vectors, to avoid unnecessary distortion calculations for those codewords which are definitely not candidates of the closest codeword to the input vector. The second is to find the Karhunen–Loeve transform (KLT) for the distribution of the training vectors, and then the partial distortion elimination method [1] is applied to the transformed vectors.

In the next section, we will describe the proposed methods. Section 3 presents experimental results to show the effectiveness of the methods. Some conclusions are given in Section 4.

## 2. The proposed methods

In this section, we will present the proposed high-speed closest codeword search algorithms. The first method is based on the difference between two mean values to reduce the search space, it is also called the mean difference method (MDM). The second one, which is referred to as the eigenvector method (EVM), is to find the Karhunen–Loeve transform for the distribution of the training vectors. It then applies the partial distortion elimination method to the transformed vectors to reject those remote codewords.

## 2.1. The mean difference method

In VQ, the great effort of searching for the closest codeword to an input vector is to compute the distortion of representing the input vector by every candidate codeword. Thus, if we can use a simple test to reject those codewords which are definitely not candidates for the closest codeword to the input vector before the distortions are calculated, the search process can be sped up. In practice, two vectors with large mean difference will have large distance. Based on this idea, the first search method is developed. It uses the mean difference to determine the search bound of the codewords. Before describing the method, we will first give some definitions and theorems.

**Definition 1.** Let $\boldsymbol{x} = (x_1, x_2, \dots, x_k)$ be a $k$-dimensional vector and $\boldsymbol{y} = (y_1, y_2, \dots, y_k)$ be a $k$-dimensional codeword. Define the mean values of $\boldsymbol{x}$ and $\boldsymbol{y}$ as

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

$$m_y = \frac{1}{k}\sum_{j=1}^{k} y_j. \tag{2}$$

**Theorem 1.** *If the distortion measure of representing* $\boldsymbol{x}$ *and* $\boldsymbol{y}$ *is defined to be the squared Euclidean distance, i.e.,*

$$\mathscr{d}^2(\boldsymbol{x},\boldsymbol{y}) = \sum_{j=1}^{k} (x_j - y_j)^2,$$

*then* $\mathscr{d}^2(\boldsymbol{x},\boldsymbol{y}) \geqslant k(m_x - m_y)^2$. (3)

**Proof.** Let $X$ be a random variable and has occurrences $x_j - y_j$, $j = 1, 2, \dots, k$, with probability $p_j = 1/k$. Then from the following well-known inequality in probability [4],

$$E(X^2) \geqslant [E(X)]^2,$$

where $E(X)$ is the expected value of $X$, we can get

$$\frac{1}{k}\left[\sum_{j=1}^{k} (x_j - y_j)^2\right] \geqslant \left[\frac{1}{k}\sum_{j=1}^{k} (x_j - y_j)\right]^2. \tag{4}$$

By combining the above inequality and Eqs. (1)–(2), Inequality (3) can be easily derived. □

From the above theory, we can see that for a codeword $\boldsymbol{y}$, if $k(m_x - m_y)^2$ is larger than the current minimum distortion $\mathscr{d}^2_{\min}$, the codeword $\boldsymbol{y}$ will not be the closest codeword to the input vector $\boldsymbol{x}$ and thus can be rejected. Based on this reason, many codewords which are definitely not candidates for the closest codeword to $\boldsymbol{x}$ can be rejected without evaluating their distortions. However, two vectors with similar mean values will have large distance if their components are very different. To solve this problem, an extra stage will be introduced. Assume that the size of each block (vector) is $k$ with $k = n \times n$, i.e. any row or column in the block contains $n$ pixels. First, every block $\boldsymbol{x}$ is decomposed into $n$ subvectors, $X_1, X_2, \dots, X_n$, with each subvector $X_i$ representing the $i$th row or the $i$th column of the block. Let $m_{xi}$ and $m_{yi}$ be the mean values of the $i$th subvectors, $X_i$ and $Y_i$, of vectors $\boldsymbol{x}$ and $\boldsymbol{y}$, respectively, for $i = 1, 2, \dots, n$. Based on these notations, a definition and two corollaries will be given.

**Definition 2.** Let $M_x = (m_{x1}, m_{x2}, \dots, m_{xn})$ and $M_y = (m_{y1}, m_{y2}, \dots, m_{yn})$, the squared distance between $M_x$ and $M_y$ is defined to be

$$\mathscr{d}_2^2(\boldsymbol{x},\boldsymbol{y}) = \mathscr{d}^2(M_x, M_y) = \sum_{i=1}^{n} (m_{xi} - m_{yi})^2.$$

From Definition 2 and Theorem 1, we have the following corollary.

**Corollary 1.** $\mathscr{d}^2(\boldsymbol{x},\boldsymbol{y}) \geqslant n\mathscr{d}_2^2(\boldsymbol{x},\boldsymbol{y}) \geqslant k(m_x - m_y)^2$. (5)

**Proof.** Since $m_x$ and $m_y$ are the mean values of $M_x$ and $M_y$, respectively, from Definition 2 and Theorem 1, we have

$$\mathscr{d}_2^2(\boldsymbol{x},\boldsymbol{y}) = \sum_{i=1}^{n} (m_{xi} - m_{yi})^2 \geqslant n(m_x - m_y)^2.$$

Since $k = n \times n$, we have

$$n\mathscr{d}_2^2(\boldsymbol{x},\boldsymbol{y}) \geqslant k(m_x - m_y)^2. \tag{6}$$

Since $m_{xi}$ and $m_{yi}$ are the mean values of $X_i$ and $Y_i$, from Theorem 1, we can get

$$d^2(x,y) = \sum_{i=1}^{n}\sum_{j=1}^{n}(X_{ij} - Y_{ij})^2 \geqslant \sum_{i=1}^{n} n(m_{xi} - m_{yi})^2 = nd_2^2(x,y), \qquad (7)$$

where $X_{ij}$ and $Y_{ij}$ represent the $j$th pixels of $X_i$ and $Y_i$, respectively. Combining Inequalities (6) and (7), we finally obtain Inequality (5). □

From Theorem 1 and Corollary 1, we can obtain the following corollary immediately.

**Corollary 2.** *Let $x$ be a training vector and $d^2_{\min}$ be the current candidate minimum distortion. For any codeword $y$, if $k(m_x - m_y)^2 \geqslant d^2_{\min}$ or $nd_2^2(x,y) \geqslant d^2_{\min}$, then $d^2(x,y) \geqslant d^2_{\min}$.*

From Corollary 2, we know that for a training vector $x$ and a codeword $y$, if $k(m_x - m_y)^2 \geqslant d^2_{\min}$ or $nd_2^2(x,y) \geqslant d^2_{\min}$, $y$ will not be the closest codeword to $x$ and can be rejected with calculating $d^2(x,y)$.

With the above theorem and corollaries in hand, we now turn to describe the MDM. For a training vector $x$, we first calculate the mean values of all subvectors of $x$ and the mean value of $x$. For every codeword $y$, if $k(m_x - m_y)^2$ is larger than the current minimum distortion $d^2_{\min}$, the codeword $y$ will be rejected. Otherwise, if $nd_2^2(x,y) \geqslant d^2_{\min}$, the codeword $y$ is rejected. If $y$ is not rejected, the distortion $d^2(x,y)$ is calculated, and if $d^2(x,y) < d^2_{\min}$, the current minimum distortion $d^2_{\min}$ is replaced by $d^2(x,y)$.

A detailed description of how to employ the MDM to the codebook design of the LBG algorithm is given below.

*Step 0.* Initialization: Given $N$ = codebook size, $M$ = the number of training vectors, $k = n \times n$ = the dimension of a training vector, $C_0$ = initial codebook, $\varepsilon$ = distortion threshold. Set iteration counter $r = 0$, initial total distortion $D_{-1} = \infty$.

*Step 1.* Compute the mean values of all subvectors and the mean value, $m_i$, of each codeword $y^{(i)}$ in the codebook $C_r$, for $i = 1, 2, \ldots, N$. Sort $C_r$ according to the increasing order of the mean values of codewords, then the sorted codebook $C_{sr}$ is

$$C_{sr} = \{y^{(i)} \mid m_i \leqslant m_{i+1},\ 1 \leqslant i \leqslant N-1\}.$$

*Step 2.* For each training vector $x_t$, find the closest codeword $y^{i(t)}$ in the codebook $C_{sr}$ and assign $x_t$ to class $i(t)$. This procedure includes the following substeps:

*Step 2.1.* Input a training vector $x_t = (x_{t1}, x_{t2}, \ldots, x_{tk})$, compute the mean values of all subvectors of $x_t$ and the mean value $m_{xt}$ of $x_t$.

*Step 2.2.* Find the codeword $y^{(p)}$ that has the minimum mean difference from $x_t$ (using binary search), i.e.,

$$|m_{xt} - m_p| \leqslant |m_{xt} - m_i| \quad \text{for all } i \neq p,$$

where $m_p$ is the mean value of $y^{(p)}$. Set $i(t) = p$ and the current minimum distortion $d^2_{\min} = d^2(x_t, y^{(p)})$.

*Step 2.3.* Find the closest codeword $y^{i(t)}$ in $C_{sr}$ and assign $x_t$ to class $i(t)$. The procedure is as follows:

```
Set $d = 1$;
while (($p + d \leqslant N$ and $k(m_{xt} - m_{p+d})^2$
      $< d^2_{\min}$) or
  ($p - d \geqslant 1$ and $k(m_{xt} - m_{p-d})^2 < d^2_{\min}$) begin
  if ($p + d \leqslant N$ and $k(m_{xt} - m_{p+d})^2 < d^2_{\min}$)
    begin /* first stage */
    if ($nd_2^2(x_t, y^{p+d}) < d^2_{\min}$) begin /*
      second stage */
      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 ($p - d \geqslant 1$ and $k(m_{xt} - m_{p-d})^2 < d^2_{\min}$)
      begin /* first state */
      if($nd_2^2(x_t, y^{p-d}) < d^2_{\min}$)
        begin /* second stage */
        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;
    $d = d + 1$;
  end; {of while}
```

*Step 3.* Compute the overall distortion for the $r$th iteration, $D_r$. Here $D_r$ is defined to be

$$D_r = \sum_{t=1}^{M} d^2(x_t, y^{i(t)}).$$

*Step 4.* If $(D_{r-1} - D_r)/D_r \leqslant \varepsilon$, halt with final codebook being $C_{sr}$. Otherwise go to *Step 5.*

*Step 5.* Compute the centroid of every class. The centroids are regarded as the codewords of the new codebook. Set $r = r + 1$ and go to *Step 1* for next iteration.

The MDM contains two stages. The first one compares the mean value of a codeword with that of the training vector; the second compares the mean values of the $n$ subvectors of the codeword with those of the training vector. Note that the second stage will be very effective when every vector is normalized so that it has zero mean or when two compared vectors with similar means have variant block types (for example, if one vector represents a homogeneous block and the other is a block with edge pixels).

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 MDM to find the closest codeword to each input vector. The details of this procedure are similar to those in *Step 2* of the codeword design algorithm described above.

## 2.2. *The eigenvector method*

As mentioned previously, the most effort in searching for the closest codeword to a training vector is to evaluate the distance between two vectors. The lower the vector dimension is, the less the effort of the distance evaluation is. If we can find a lower-dimensional subspace $S$ and for every vector $\boldsymbol{v}$, its projection vector on $S$ can approximate $\boldsymbol{v}$ very well, the distance between two projected vectors will approach to that between two original vectors. Thus, those remote codewords can be rejected only through testing the distance between their projected vectors and the projected vector of the input vector. The first stage of the MDM is such an example, the mean value of a vector represents the point of projecting the vector onto the subspace $S_1$ spanned by the unit vector $(1/\sqrt{k}, 1/\sqrt{k}, \dots, 1/\sqrt{k})$. This means that in order to lower the complexity of the distance evaluation, we have projected every vector onto the one-dimensional subspace $S_1$ to approximate the vector. Since the rate of rejecting codewords before evaluating their distortions depends on the approximation accuracy, how to find a projection subspace with proper dimension becomes an important topic. In fact, the Karhunen–Loeve transform (KLT) for the distribution of the training vectors can be used to find the best subspace [10]. Based on this transform and the above idea, the eigenvector method (EVM) is developed. It will first use KLT to find the best lower-dimensional subspace and projects every training vector and codeword onto the subspace to get the projected vectors. Then the partial distortion elimination method [1] is applied to quickly reject those codewords which are definitely not candidates for the closest codeword match. The detail is described as follows.

Let $C_x$ be the covariance matrix of the training vectors $\boldsymbol{x}$'s and be represented by

$$C_x = E((\boldsymbol{x} - \boldsymbol{m}_x)(\boldsymbol{x} - \boldsymbol{m}_x)^{\mathrm{T}}),$$

where $\boldsymbol{m}_x = E(\boldsymbol{x})$, and $T$ indicates vector transposition. For all training vectors, under the mean square error sense, the best $p$-dimensional projection subspace, $S_p$, is spanned by the $p$ orthonormal eigenvectors associated with the $p$ largest eigenvalues of $C_x$, the reason can be found in [10]. Let $A$ be the matrix whose rows are formed by the orthonormal eigenvectors of $C_x$ and are ordered such that the first row of $A$ is the eigenvector associated with the largest eigenvalue, and the last row is the eigenvector associated with the smallest eigenvalue. If we use the eigenvectors as the coordinate axes, then the new coordinates $\boldsymbol{x}' = (x'_1, x'_2, \dots, x'_k)$ of each vector $\boldsymbol{x}$ can be obtained by the following equation:

$$\boldsymbol{x}' = A\boldsymbol{x}.$$

This is the well-known KLT. Let $\boldsymbol{x}$ and $\boldsymbol{y}$ be two $k$-dimensional vectors, and $\boldsymbol{x}'$ and $\boldsymbol{y}'$ be the new coordinates of $\boldsymbol{x}$ and $\boldsymbol{y}$ projected on the eigenvector coordinate system, respectively, then $\boldsymbol{x}' = A\boldsymbol{x}$ and

$y' = Ay$. Since $A$ is an orthonormal transform matrix, the distance between $x$ and $y$ is equal to the distance between $x'$ and $y'$, i.e., $d^2(x',y') = d^2(x,y)$. Let $A_p$ be the $p \times k$ matrix whose rows are formed by the first $p$ rows of $A$. Let $x'_p = (x'_1, x'_2, \ldots, x'_p)$ and $y'_p = (y'_1, y'_2, \ldots, y'_p)$ be two $p$-dimensional vectors of $x$ and $y$ projected on the subspace, $S_p$, spanned by $A_p$, i.e., $x'_p = A_p x$ and $y'_p = A_p y$. Therefore, we can get

$$d^2(x,y) = d^2(x',y') = \sum_{j=1}^{k} (x'_j - y'_j)^2 \geqslant \sum_{j=1}^{p} (x'_j - y'_j)^2 = d^2(x'_p, y'_p).$$

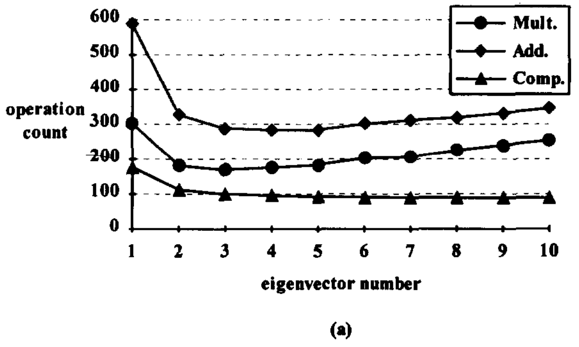
Having the above result, we will begin describing the EVM.

Let $x$ be a training vector and the current candidate minimum distortion be $d^2_{\min}$. For any codeword $y$, if $d^2(x'_p, y'_p)$ is larger than $d^2_{\min}$, $y$ will not be the closest codeword to $x$ and can be rejected without calculating its distortion. To test whether $d^2(x'_p, y'_p)$ is larger than $d^2_{\min}$, the partial distortion elimination method [1] is applied to speed up the testing process.

Note that the EVM needs some overhead to obtain the covariance matrix of the training vectors, the eigenvectors and the projected vectors. As contrasted with the MDM, the EVM has a higher rejection rate but needs more overhead. Therefore, for codebooks with smaller codebook sizes, the MDM may outperform the EVM, and the EVM is proper for those codebooks with larger codebook sizes.

## 3. Experimental results

To examine the efficiency of the proposed two methods, experiments were performed 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 algorithms were compared with the LBG algorithm and the fast nearest-neighbor search (FNNS) algorithm [14], which is based on the triangle inequality, in terms of the numbers of multiplication, addition, and comparison operations required in codebook design as well as image encoding.



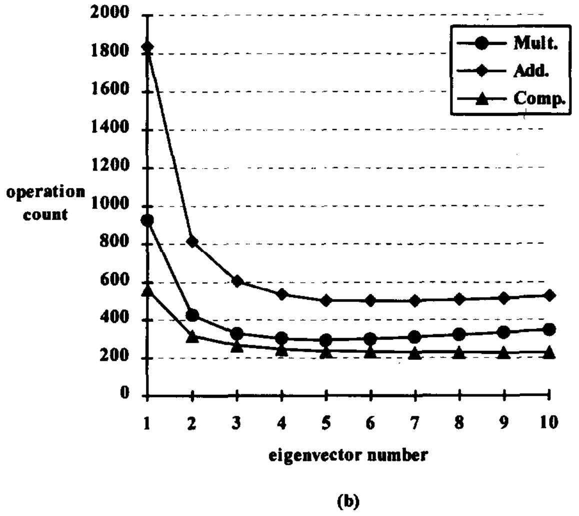


Fig. 1. The average numbers of mathematical operations (overhead included) required per vector in codebook design using the EVM with variant codebook sizes versus selected eigenvector number $p$. (a) Codebook size 128, (b) codebook size 512.

Fig. 1 shows the average numbers of multiplication, addition, and comparison operations required per vector in codebook design using the EVM algorithm versus the number of selected eigenvectors. From this figure, we can see that selecting more eigenvectors does not guarantee to result in better performance. For example, when codebook sizes of 128 and 512 are chosen, selecting 3 and 5 eigenvectors, respectively, will perform best. In practice, the more eigenvectors are selected, the more codewords can be rejected. However, the

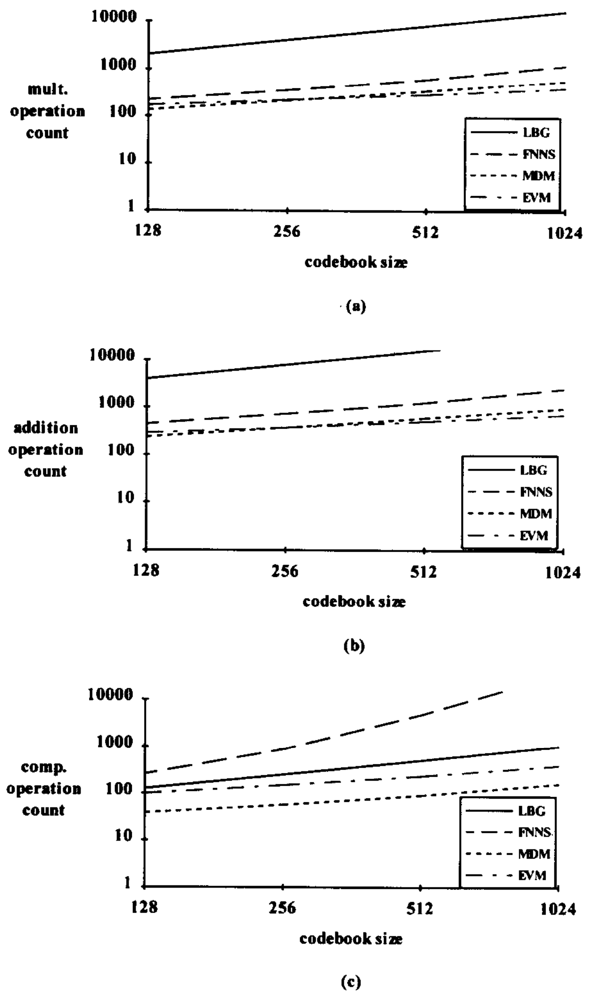


Fig. 2. The average numbers of mathematical operations (overhead included) required per vector in codebook design versus codebook sizes. (a) Multiplication, (b) addition, (c) comparison.

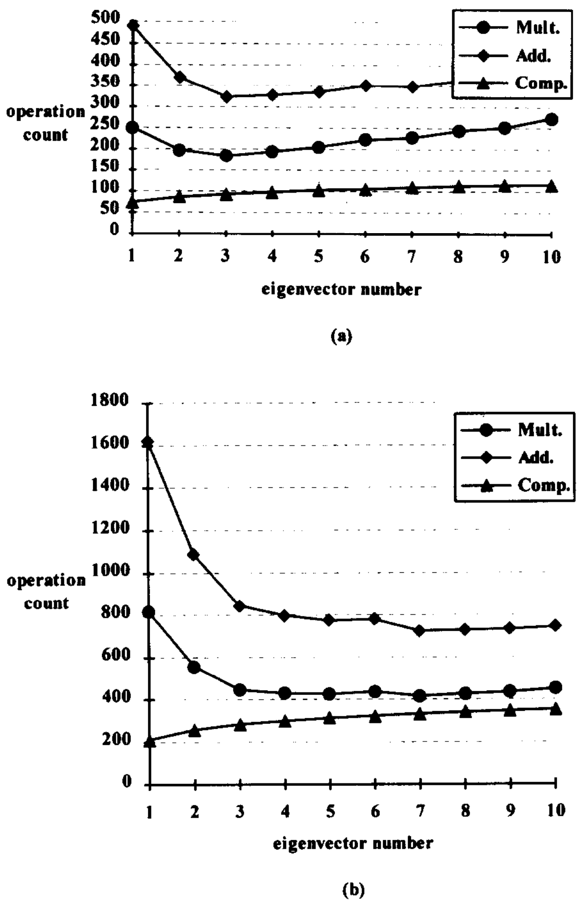


Fig. 3. The average numbers of mathematical operations required per vector in image encoding using the EVM with variant codebook sizes versus selected eigenvector number $p$. (a) Codebook size 128, (b) codebook size 512.

overhead will increase in proportion to the number of selected eigenvectors. From this figure, we can see that the more the codewords are, the more the eigenvectors we need.

Fig. 2 shows the average numbers of multiplication, addition, and comparison operations required per vector in codebook design using the LBG, the FNNS, the MDM and the EVM algorithms. The MDM uses a column of a block to represent a subvector. The image Lena was used to design the codebook. From this figure, we can see that the MDM and EVM algorithms require much fewer multiplication and addition operations needed by the LGB algorithm and outperform the FNNS algorithm. The overhead required in the MDM and the EVM algorithms and that in the FNNS algorithm are included in this figure.

Fig. 3 shows the average numbers of multiplication, addition, and comparison operations required per vector in image encoding using the EVM algorithm versus the number of selected eigenvectors. In the simulation, the image Lena was used to design a codebook, and the resulting codebook was then used to encode the four images (Lena, Peppers, Jet

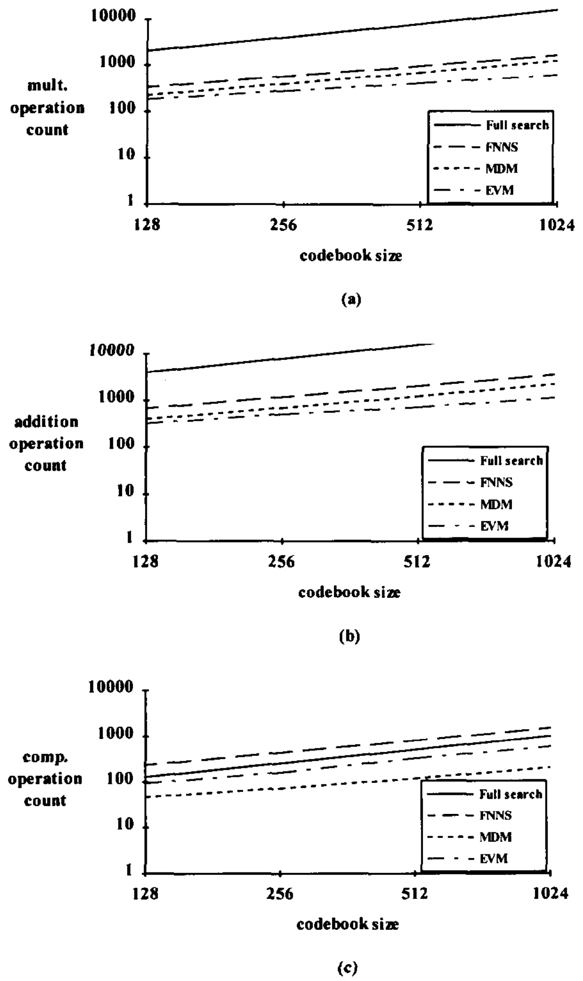


Fig. 4. The average numbers of mathematical operations required per vector in image encoding of four images (Lena, Peppers, Jet and Baboon) versus codebook sizes: (a) multiplication, (b) addition, (c) comparison.

and Baboon). The overhead is excluded in this figure, since the eigenvectors and the projection of each codeword onto the subspace spanned by the selected eigenvectors can be calculated in advance and stored for image encoding.

Fig. 4 shows the average numbers of mathematical operations required per vector for image encoding by giving a predesigned codebook using the full search, the FNNS, the MDM and EVM algorithms. From this figure, we can see that the EVM results in the best performance, since much of its overhead has been calculated in advance and is not included in this figure.

From these figures, we can see that in the codebook design process, the MDM performs best when a codebook has a smaller size (e.g., 128 and 256), while the EVM performs best for codebooks with larger size (eg., 512 and 1024). Both methods outperform the FNNS algorithm. In image encoding, the EVM results in the best performance because the overhead required in evaluating the eigenvectors and the projection vectors of all codewords can be done in advance.

## 4. Conclusions

In this paper, based on the assumption that the distortion is measured by the squared Euclidean distance, two high-speed closest codeword search algorithms: the MDM and the EVM, for vector quantization have been proposed. The proposed algorithms can speed up the search process in VQ codebook design as well as image encoding. The MDM uses the difference between the mean values of two vectors to reduce the search space. The EVM projects each vector onto the subspace spanned by the Karhunen–Loeve transform. It then calculates the distance between a training vector and every codeword in the projected subspace to reject those codewords which are definitely not candidates for the closest codeword to the training vector. The performance of the proposed algorithms has been evaluated in both codebook design as well as image encoding. Simulation results show that the proposed algorithms can save a great number of mathematical operations required in the LBG algorithm and outperform the FNNS algorithm.

## Acknowledgements

The authors would like to thank the anonymous reviewers for their many valuable suggestions which have greatly improved the presentation of the paper.

## References

[1] 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, October 1985, pp. 1132–1133.

[2] D.Y. Cheng, A. Gersho, B. Ramamurthi and Y. Shoham, "Fast search algorithms for vector quantization and pattern matching", *Proc. IEEE Internat. Conf. Acoust. Speech Signal Process.*, 1984, pp. 9.11.1–9.11.4.

[3] D.Y. Cheng and A. Gersho, "A fast codebook search algorithm for nearest neighbour pattern matching", *Proc. IEEE Internat. Conf. Acoust. Speech Signal Process.*, 1986, pp. 265–268.

[4] E.R. Dougherty, *Probability and Statistics for the Engineering, Computing and Physical Sciences*, Prentice-Hall, Englewood Cliffs, NJ, 1990.

[5] W.H. Equitz, "A new vector quantization clustering algorithm", *IEEE Trans. Acoust. Speech Signal Process.*, Vol. 37, No. 10, October 1989, pp. 1568–1575.

[6] A. Gersho and R.M. Gray, *Vector Quantization and Signal Compression*, Kluwer, Boston, 1992.

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

[8] C.H. Hsieh, P.C. Lu and J.C. Chang, "Fast codebook generatioin algorithm for vector quantization of images", *Pattern Recognition Lett.*, Vol. 12, 1991, pp. 605–609.

[9] C.M. Huang, Q. Bi, G.S. Stiles and R.W. Harris, "Fash full search equivalent encoding algorithms for image compression using vector quantization", *IEEE Trans. Image Process.*, Vol. 1, No. 3, July 1992, pp. 413–416.

[10] A.K. Jain, *Fundamentals of Digital Image Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1988.

[11] Y. Linde, A. Buzo and R.M. Gray, "An algorithm for vector quantizer design", *IEEE Trans. Commun.*, Vol. COM-28, No. 1, January 1980, pp. 84–95.

[12] A. Lowry, S. Hossain and W. Millar, "Binary search trees for vector quantization", *Proc. IEEE Internat. Conf. Acoust. Speech Signal Process.*, 1987, pp. 2205–2208.

[13] N.M. Nasrabadi and R.A. King, "Image coding using vector quantization: A review", *IEEE Trans. Commun.*, Vol. COM-36, No. 8, August 1988, pp. 957–971.

[14] M.T. Orchard "A fast nearest-neighbor search algorithm", *Proc. IEEE Internat. Conf. Acoust. Speech Signal Process.*, 1991, pp. 2297–2300.

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

[16] V. Ramasubramanian and K.K. Paliwal, "Fast *k*-dimensional tree algorithms for nearest neighbor search with application to vector quantization encoding". *IEEE Trans. Signal Process.*, Vol. 40, No. 3, March 1992, pp. 518–531.

[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 Lett.*, Vol. 13, 1992, pp. 471–480.

[18] M.R. Soleymani and S.D. Morgera, "An efficient nearest neighbor search method", *IEEE Trans. Commun.*, Vol. COM-35, No. 6, June 1987, pp. 677–679.

[19] E. Vidal "An algorithm for finding nearest neighbors in (approximately) constant average time complexity", *Pattern Recognition Lett.*, Vol. 4, 1986, pp. 145–157.