Fast codeword search algorithm for real-time codebook generation in adaptive VQ

D. Ghosh
A. P. Shivaprasad

Indexing terms: Adaptive vector quantisation, Optimal codebook design, Mean value, Sum of the absolute differences, Euclidean distance

Abstract: Adaptive vector quantisation is used in image sequence coding where the code-book is updated continuously to keep track with the changing source statistics. Hence, for real-time video coding applications, both the processes of quantising the input vectors and updating the codebook are required to be fast. Since the nearest codeword search is involved in both these processes, a fast codeword search algorithm can make the coding process time efficient. The authors describe a proposed codeword search algorithm with reduced search space. The algorithm uses the mean value and the sum of the absolute differences as the two criteria to reject unlikely codewords, thereby saving a great deal of computational time, while introducing no more distortion than the conventional full search algorithm. Simulation results obtained confirm the effectiveness of the proposed algorithm in terms of computational complexity.

1 Introduction

1.1 Overview of vector quantisation

Vector quantisation (VQ) plays a critical role as an important building block of many lossy data compression systems [1]. In VQ for image compression, the input image is divided into several rectangular blocks which form vectors of fixed dimension. A codebook containing several codevectors (codewords) is used to encode the input vectors. The codeword closest to an input vector, i.e. one having the minimum Euclidean distance with respect to the input vector, is used to represent that input vector. Compression is achieved by transmitting the binary address of the selected codeword in the codebook. Thus, VQ can be defined as a mapping from the set of k-dimensional input vectors $\mathbf{X}$ to the set of k-dimensional codevectors $\mathbf{Y}$ (codebook), that is

$$Q : \mathbf{X} \rightarrow \mathbf{Y}$$

where $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\} \subset R^k$

$\mathbf{Y} = \{\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_C\} \subset R^k$

$\mathbf{x}_i = (x_{i1}, x_{i2}, ..., x_{ik})$ is the input vector for $i = 1, 2, ..., N$

$\mathbf{y}_j = (y_{j1}, y_{j2}, ..., y_{jk})$ is the codeword for $j = 1, 2, ..., C$

N = number of input vectors obtained from the input image

C = number of codewords in the codebook, i.e. the size of the codebook

If a fixed length coder is used to encode the addresses of the codewords, the average bit rate is given as

$$BR = \frac{1}{k}\lceil \log_2 C \rceil \text{ bits/pixel} \quad (1)$$

The distance between the input vector $\mathbf{x}_i$ and the codeword $\mathbf{y}_j$ is given by the second Euclidean norm as

$$d(\mathbf{x}_i, \mathbf{y}_j) = \|\mathbf{x}_i - \mathbf{y}_j\| = \left(\sum_{n=1}^{k} (x_{in} - y_{jn})^2 \right)^{1/2} \quad (2)$$

The image is reconstructed by replacing each input vector by the codeword used to represent it. This implies that in order to achieve a good quality of reconstructed image with minimum mean square error (MSE), $d^2(\mathbf{x}_i, \mathbf{y}_j)$, $i = 1, 2, ..., N$, should be as small as possible when $Q(\mathbf{x}_i) = \mathbf{y}_j$. Hence, the coding efficiency in terms of the quality of the reconstructed image depends upon the choice of codewords in the codebook. The process of choosing codewords in the codebook is referred to as the codebook design.

1.2 Optimal codebook design

When any input vector $\mathbf{v}$ is reproduced as $\hat{\mathbf{v}}$, the distortion is given by $d(\mathbf{v}, \hat{\mathbf{v}})$. A system will be good if it yields a small average distortion. For a quantiser to be optimal there are two necessary conditions [1] as follows:

(1) For a given decoder in a memoryless vector quantiser the best encoder is a nearest neighbour mapper, i.e. if $Q(\mathbf{x}_i) = \mathbf{y}_j$, then

$$\min_{\mathbf{y}_l \in \mathbf{Y}} d(\mathbf{x}_i, \mathbf{y}_l) = d(\mathbf{x}_i, \mathbf{y}_j) = d_{min}(\mathbf{x}_i) \quad (3)$$

(2) For a given encoder the best reproduction for all the input vectors contained in a given Voronoi region is the centroid of that region in the vector space.

Thus, a quantiser in VQ is optimal if it minimises the average distortion given as

$$D = \frac{1}{N} \sum_{i=1}^{N} d^2_{min}(\mathbf{x}_i) = \frac{1}{N} \sum_{i=1}^{N} \min_{\mathbf{y}_l \in \mathbf{Y}} d^2(\mathbf{x}_i, \mathbf{y}_l) \quad (4)$$

In designing a codebook we take a set of input vectors $\mathbf{X}$ as the training vectors and the design is based on

minimisation of the average distortion measure between the training vectors and the codebook vectors used to represent them. One possible approach for the minimisation of average distortion is the C-means algorithm (CM) or the generalised Lloyd algorithm where the criterion function **D** given in eqn. 4 is iteratively minimised. An initial codebook consisting of C random codewords is taken and the training vectors are mapped to these codewords on the basis of nearest neighbour rule. Thus the training vectors are partitioned into C clusters and the centroids of every cluster form the new set of codewords. If the average distortion function is small enough, the process is stopped. Else, the process of clustering and subsequent updating of the codewords is performed iteratively until a minimum for **D** is obtained. A variation of the C-means algorithm, known as the LBG algorithm [2], is widely used in VQ. Each step of the LBG algorithm must either reduce average distortion or leave it unchanged. The algorithm is usually stopped when the relative distortion decrease falls below some threshold.

1.3 Adaptive vector quantisation

As follows from the discussion above, an optimal codebook should completely reflect the statistics of the input vectors. However, the codebook is locally optimised for the particular training set, chosen as the representative of the input images. In video applications, the sources are generally nonstationary. Thus, vector quantisation of an image sequence using a universal (fixed) codebook requires a large codebook which is *a priori* able to encode the entire sequence effectively. A different approach to this that uses a small size codebook is *adaptive vector quantisation* (AVQ) where the codebook is updated continuously to keep track of the local frame statistics [3–6]. The basic idea of AVQ is to generate a new set of codewords with the vectors obtained from the current input frame as the set of training vectors, and to replace some of the codewords in the old codebook with codewords from the set of new codewords. Thus, part of the codebook is updated with the changing source statistics. Therefore, the steps involved in coding the current input frame using AVQ are:

(1) Formation of the input vector set.

(2) Generation of a set of new codewords, i.e. a new codebook.

(3) Updating of the old codebook.

(4) Quantisation (encoding) of the input vectors.

The main task involved in the second and fourth steps (codebook generation and vector quantisation) is the search for the closest codeword in the codebook for each input vector. For a codebook of size C and vector dimension k, the conventional full-search algorithm [2] for closest codeword search requires C x k multiplications, $C \times (2k - 1)$ additions and $C - 1$ comparisons. For low distortion coding, the codebook should be large enough to accommodate codewords for various possible input vectors. High compression in VQ can be achieved if the dimensionality of the vector is increased. Generally, high quality or high compression can be attained only at the expense of the other. In order to achieve both quality and compression in VQ coding, both C and k should be large, thus making the VQ computationally inefficient. Hence, for real-time coding of video signals, the conventional full-search method for codeword search is inappropriate, necessitating the development of computationally efficient fast codeword search algorithms.

1.4 Fast codeword search algorithms

Various fast codeword search algorithms had been developed [7–15]. In [7], the partial distortion elimination (PDE) algorithm is introduced where a codeword $\mathbf{y}_j$ is rejected from consideration as a representative codeword for the input vector $\mathbf{x}_i$ if the accumulated distortion for the first k ($1 \leq k < k$) entities is larger than (or equal to) d^2_{min}, i.e.

$$\sum_{n=1}^{\hat{k}} (x_{in} - y_{jn})^2 \geq d^2_{min} \qquad (5)$$

where $d_{min} = d(\mathbf{x}_i, \mathbf{y}_l)$, $\mathbf{y}_l$ is the current best match for $\mathbf{x}_i$ and k is the dimensionality of the vectors.

An algorithm based on PDE search can be very effective if the first codeword chosen is the minimum distortion codeword or at least very close to it. If the current d_{min} value is very close to the minimum distortion, most of the remaining codewords will result in a premature exit and once the best match is obtained the rest of the codewords always result in a premature exit. Hence, it is necessary to find a probable nearest codeword and take it as the initial best match. This may be accomplished by prediction. Since spatial correlation exists in an image, it may be assumed that the current input vector and the preceding vectors in the neighbourhood are usually similar. Codewords that represent the neighbouring vectors are used for prediction; the one among them which gives the minimum distortion is taken. This is the predictive partial distance search (PPDS) algorithm [8].

Now it follows that, if the prediction is made appropriately, the initial best match codeword will be very close to the minimum distortion codeword. Moreover, as the search process continues, the current best match codeword $\mathbf{y}_l$ will be closer and closer to the minimum distortion codeword. When the current best match $\mathbf{y}_l$ is very close to the minimum distortion codeword, most of the remaining codewords will result in a premature exit. Thus, further computational efficiency can be achieved if the partial distortion calculations for these unlikely codewords is avoided. This means that it is required to identify the unlikely codewords and reject them beforehand. The triangle inequality elimination (TIE) algorithm [9] and the fast nearest neighbour search (FNNS) algorithm [10] reject unlikely codewords on the basis of a triangle inequality relation. **A** codevector $\mathbf{y}_j$ is eliminated from consideration as a candidate for the closest codeword if

$$d(\mathbf{y}_j, \mathbf{y}_l) > 2d_{min} \qquad (6)$$

In [11], Li and Salari have shown that a codevector $\mathbf{y}_j$ can be a candidate for the closest codeword only if it satisfies the relation

$$d(\mathbf{x}_i, \mathbf{z}) - d_{min} \leq d(\mathbf{y}_j, \mathbf{z}) \leq d(\mathbf{x}_i, \mathbf{z}) + d_{min} \qquad (7)$$

where $\mathbf{z}$ is a control vector. More than one control vector may be taken for further reduction of the range of $\mathbf{y}_j$.

In these algorithms [9–11], the search space is gradually confined to the minimum distortion codeword only at the cost of memory and excessive preprocessing. In the TIE and FNNS algorithms, after each iteration involving the processes of clustering of training vectors and subsequent updating of the codewords, the dis-

tances between all the pairs of codewords in the current codebook are calculated and stored in a table of size C x (C – 1). In the algorithm by Li and Salari, after each iteration, the distances between the codewords in the current codebook and the control vector(s) are calculated and stored in a table of size C x $\mathcal{N}_{cv}$, where $\mathcal{N}_{cv}$ is the number of control vectors.

The memory and preprocessing cost is reduced to some extent in the partial search partial distortion (PSPD) algorithm [12], where only the mean of each codeword in the codebook is calculated and stored after each iteration involving clustering and codeword updating. The mean value of a k-dimensional vector **p** is given as

$$m_p = \frac{1}{k} \sum_{n=1}^{k} p_n \tag{8}$$

The codeword with the minimum mean difference from the input vector x, is taken as the probable nearest codevector $\mathbf{y}_{l1}$ and PDE is applied only to codewords with mean differences from $\mathbf{y}_{l1}$ less than a predetermined threshold T. However, sometimes the closest codeword may not be included within this search range, thereby introducing more distortion compared to the full-search algorithm. This problem was resolved by Poggi [13], where the search range includes those codewords whose mean value ranges from m_{max} to m_{min}, where m_{max} and m_{min} are given as

$$m_{max} = m_{xi} + d_{min}/\sqrt{k} \tag{9}$$

$$m_{min} = m_{xi} - d_{min}/\sqrt{k} \tag{10}$$

The PDE algorithm is applied to the codewords included in the search range to select the best codeword.

A similar algorithm was developed by Guan and Kamel [14] called the equal average nearest neighbour search (ENNS) algorithm, where the search is restricted within the space between two equal-average hyperplanes. Further reduction in search space was proposed by Lee and Chen [15]. In this algorithm the ENNS algorithm is modified by rejecting those codewords, although included in the space between the two hyperplanes, whose variances differ from that of $\mathbf{x}_i$ by an amount greater than (or equal to) d_{min}. Here, after each iteration involving clustering and codeword updating, the variances of all the updated codewords are calculated and stored, in addition to their mean values. Hence, the search space is reduced only at the cost of additional memory and preprocessing costs. Therefore, the modified ENNS algorithm, although useful for quantising an input vector, is inappropriate for real-time codebook generationlupdating.

To overcome the problem posed by the modified ENNS algorithm introduced by Lee and Chen we proposed a different modification to the ENNS algorithm in [16]. This algorithm reduces the memory and preprocessing costs and at the same time reduces the search space compared to that of Lee and Chen. Hence, our proposed algorithm may be used for real-time codebook updating and subsequent vector quantisation of the input vectors in image sequence coding using AVQ.

2 ENNS algorithm

In [14], Guan and Kamel proposed an equal-average nearest neighbour search (ENNS) algorithm in which the search is restricted within the space between two equal-average hyperplanes, using the mean value criterion, making the search process faster than the full-search algorithm. Thus, the codebook contains a table for mean values of all the codewords. In an Euclidean space R^k, the coordinates of any point on the central line l have the same value. Each point on a fixed hyperplane orthogonal to 1 has the same mean value and such a hyperplane is called an equal-average hyperplane.

The search procedure is as follows:

(1) The ENNS algorithm first calculates the mean of input vector $\mathbf{x}_i$ as given by eqn. 8.

(2) Next the algorithm finds a codeword $\mathbf{y}_{l1}$ such that $|m_{xi} - m_{yl1}|$ is minimum and the current minimum distortion $d_{min} = d(\mathbf{x}_i, \mathbf{y}_{l1})$ is calculated. Any other codeword $\mathbf{y}_j$ closer to $\mathbf{x}_i$ must satisfy the condition $d(\sim\mathbf{y}_{lj}) < d_{min}$ and so should be located inside the hypersphere centred at $\mathbf{x}_i$ with radius d_{min}. The hypersphere can be bounded by two equal-average hyperplanes intersecting l at L_{max} and L_{min}, where $L_{max} = (m_{max}, m_{max}, ..., m_{max})$ and $L_{min} = (m_{min}, m_{min}, ..., m_{min})$, m_{max} and m_{min} are given by eqns. 9 and 10, respectively.

(3) Now, it is only necessary to search for the nearest codeword only among the codewords located between these two hyperplanes, i.e. codewords with mean values ranging from m_{min} to m_{max}. Fig. 1 illustrates this for k = 2.

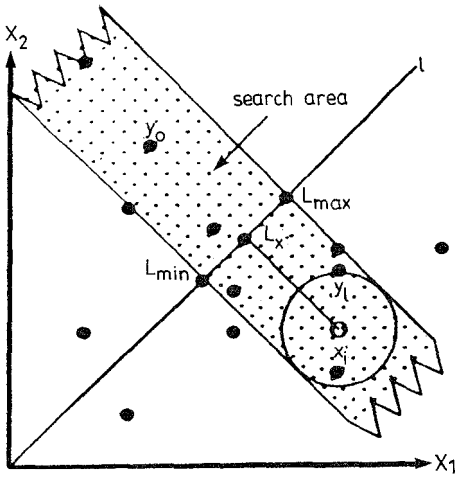


Fig. 1 *ENNS algorithm*

Two vectors with the same mean value may be far apart from each other. As in Fig. 1, $\mathbf{y}_o$ is a codeword that has a mean value well within the specified range but is quite unlikely to be a true representation of the vector $\mathbf{x}_i$. This problem is taken care of in the search algorithm given by Lee and Chen in [15] using both the mean and variance of a vector to sort out possible representative codewords, thereby reducing the search space further. The algorithm is described below.

2.1 Modified ENNS algorithm [15]

The variance of a k-dimensional vector **p** is given as

$$V_p = \sqrt{\sum_{n=1}^{k} (p_n - m_p)^2} = d(\mathbf{p}, L_p) = \|(\mathbf{p} - L_p)\| \tag{11}$$

where $L_n = (in_n, m_p, ..., m_p)$ is the projection point of $\mathbf{p}$ on line l. It can be shown [15] that for any codeword $\mathbf{y}_t$

$$d(\mathbf{x}_i, \mathbf{y}_t) \geq |V_{xi} - V_{yt}| \quad (12)$$

We see that once the initial guess for $\mathbf{y}_{l1}$ is made, any other codeword $\mathbf{y}_j$, located between the two equal-average hyperplanes, closer to $\mathbf{x}_i$ must satisfy

$$d(\mathbf{x}_i, \mathbf{y}_j) < d_{min} \Rightarrow |V_{xi} - V_{yj}| < d_{min} \quad (13)$$

Therefore, the distance calculation is necessary only for those codewords which satisfy the inequality eqn. 13. The search space for a two-dimensional case is shown in Fig. 2.

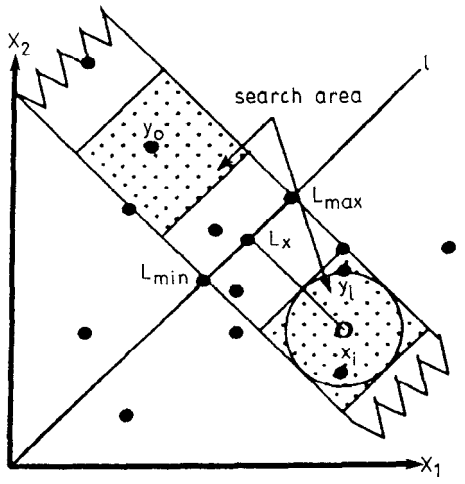


Fig. 2 *Modified ENNS algorithm [15]*

2.2 Drawbacks of the algorithm

Although here the search space is reduced to a great extent it suffers from the following drawbacks:

(1) The codebook design process is inefficient because of added preprocessing involving (2k – 1) additions, k multiplications and one syuare-root operation for variance calculation of each codeword after each iteration, hence, not useful for real-time codebook generation.

(2) The memory requirement is increased by an amount N to store N number of variances for the N codewords.

(3) The search space, although reduced to a great extent, still includes some unlikely codewords, e.g. $\mathbf{y}_o$ in Fig. 2.

A solution to the above problems was given in [16]. The fast equal average nearest neighbour search algorithm may bc used for real-timc codcbook generation/ updating in **AVQ.** This is described in detail in this paper. The ENNS algorithm is modified using the sum of absolute difference **(SAD)** criterion in addition to the mean value criterion. This means that some extra additions are required in our algorithm, while there is no need to compute the variances of the codewords. Since addition is simpler than multiplication, the proposed search method is computationally more efficient than that in [15]. Thus, the search algorithm can be used for rcal-time codebook design and encoding with a reduced search space in which the overall overhead computation and memory costs are minimised to a great extent only at the cost of some extra additions.

3 Proposed algorithm

Let us consider a set of orthonormal bases $\mathcal{V} = \{\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_k\}$ for the Euclidean vector spacc R^k. If $\mathbf{p}$ is any k-dimensional vector, then it can be written as a linear combination of the k orthonormal bases, i.e.

$$\mathbf{p} = (p_1, p_2, \ldots, p_k) = p_1\mathbf{v}_1 + p_2\mathbf{v}_2 + \cdots + p_k\mathbf{v}_k \quad (14)$$

Any vector contained in the vector space R^k can be represented as a point in an imaginary special coordinate system defined by the k orthonormal bases with the null vector $\mathbf{0}$ as the origin. The bases lie along the coordinate axes with the nth basis along the nth axis. Hence, the nth component of $\mathbf{p}$ (i.e. p_n) is the component of $\mathbf{p}$ along the nth coordinate axis.

Now, following the k-dimensional geometry [17] we have

Definition 1: A *hyperplane* in R^k is defined to be the set of points

$$\mathbf{P} = \{\mathbf{p} | \mathbf{c}\mathbf{p}^T = z\} \quad (15)$$

with $\mathbf{c} \neq \mathbf{0}$ being a given k-component row vector, $\mathbf{p}^T$ is the transpose of vector $\mathbf{p}$ and z is a given scalar.

Definition 2: Given the hyperplane $\mathbf{c}\mathbf{p}^T = z$ in R^k $(\mathbf{c} \neq \mathbf{0})$ then $\mathbf{c}$ is a vector *normal* to the hyperplane. Any vector $\lambda\mathbf{c}$ is also normal to the hyperplane (A ≠ 0). The two vectors of unit length $\mathbf{c}/\|\mathbf{c}\|$, $-\mathbf{c}/\|\mathbf{c}\|$ are the unit normals to the hyperplane.

The following can be deduced for a hyperplane given by eqn. 15:

(1) Two hyperplanes are parallel if they have the same unit normal.

(2) The distance of the hyperplane from the origin is given by $|z|/\|\mathbf{c}\|$.

(3) If $z > 0$, then $\mathbf{c}$ points towards the hyperplane. If $z < 0$, then $\mathbf{c}$ points away from the hyperplane. If $z = 0$, the hyperplane passes through the origin. From this it follows that, for a given z $(z \neq 0)$, the two hyperplanes $\mathbf{c}\mathbf{p}^T = z$ and $\mathbf{c}\mathbf{p}^T = -z$ are parallel to each other and lie on the two opposite sides of the origin.

Definition 3: Given a boundary point $\mathbf{w}$ of a convex set $\mathcal{P}$; then $\mathbf{c}\mathbf{p}^T = z$ is called a *supporting hyperplane* at $\mathbf{w}$ if $\mathbf{c}\mathbf{w}^T = z$ and if all of $\mathbf{P}$ lies in one closed half-space produced by the hyperplane, that is, $\mathbf{c}\mathbf{u}^T \geq z$ for all $\mathbf{u} \in \mathcal{P}$ or $\mathbf{c}\mathbf{u}^T \leq z$ for all $\mathbf{u} \in \mathcal{P}$.

Definition 4: A *hypersphere* in R^k with centre at $\mathbf{a}$ and radius $\varepsilon > 0$ is defined as the set of points

$$\mathcal{P} = \{\mathbf{p} | \|\mathbf{p} - \mathbf{a}\| = \epsilon\} \quad (16)$$

Definition 5: The *inside* of a hypersphere with centre at $\mathbf{a}$ and radius $\varepsilon > 0$ is the set of points

$$\mathcal{P} = \{\mathbf{p} | \|\mathbf{p} - \mathbf{a}\| < \epsilon\} \quad (17)$$

Definition 6: An *ε neighbourhood* about the point $\mathbf{a}$ is defined as the set of points inside the hypersphere with centre at $\mathbf{a}$ and radius $\varepsilon > 0$.

Let us now consider a set

$$\mathcal{C} = \{\mathbf{c}_\mu | \mathbf{c}_\mu = (c_{\mu 1}, c_{\mu 2}, \ldots, c_{\mu k}); \; |c_{\mu n}| = 1, \forall n = 1, 2, \ldots, k\} \quad (18)$$

A total of 2^k distinct vectors can be obtained that satisfy the above set relation $\mathcal{C}$. This means that the cardinality of set $\mathcal{C}$ is 2^k, Therefore, as follows from the definition of normals to a hyperplane, we observe that for a given z, 2^k number of hyperplanes can be constructed with normals to each of them given by $\mathbf{c}_\mu$ (μ = 1, 2, ..., 2^k). The equation to the hyperplane with normal $\mathbf{c}_\mu$, thus, is given by

$$\mathbf{c}_\mu \mathbf{p}^T = z \quad (19)$$

If we take z such that $|z|/\|\mathbf{c}_\mu\| = d_{min}$ ($d_{min} > 0$) then the hyperplane given by eqn. 19 is at a distance d_{min} from the origin. Therefore, the hyperplane will be tangential to the hypersphere with centre at the origin and radius d_{min} and, hence, is the only supporting hyperplane for the hypersphere at the point where it touches the hypersphere. It thus follows that there can be 2^k supporting hyperplanes that are tangential to the hypersphere. Now,

$$|z| = d_{min}\|\mathbf{c}_\mu\| = d_{min}\sqrt{\sum_{n=1}^{k} c_{\mu n}^2} = \sqrt{k}d_{min} \quad (20)$$

Since z may be $\sqrt{k}d_{min}$, or $-\sqrt{k}d_{min}$, for every $\mathbf{c}_\mu$, we can have two hyperplanes parallel to each other and lying on the two opposite sides of of the origin that form a pair of supporting hyperplanes. But, for any vector $\mathbf{c}_u$ ($1 \le \mu \le 2^k$) we can find one and only one vector $\mathbf{c}_\gamma$ ($1 \le \gamma \le 2^k$, $\gamma \ne \mu$) such that $\mathbf{c}_\mu = -\mathbf{c}_\gamma$. Therefore,

$$\begin{aligned} & \mathbf{c}_\mu \mathbf{p}^T = -\sqrt{k}d_{min} \\ \Rightarrow \quad & -\mathbf{c}_\mu \mathbf{p}^T = \sqrt{k}d_{min} \\ \Rightarrow \quad & \mathbf{c}_\gamma \mathbf{p}^T = \sqrt{k}d_{min} \end{aligned} \quad (21)$$

Hence, we observe that the vectors in set $\mathcal{C}$ can form exactly 2" supporting hyperplanes that are tangential to the hypersphere at the origin with radius d_{min}. The equation to the family of these hyperplanes is given as

$$\mathbf{c}_\mu \mathbf{p}^T = \sqrt{k}d_{min}, \forall \mu = 1, 2, \ldots, 2^k \quad (22a)$$

or

$$c_{\mu 1}p_1 + c_{\mu 2}p_2 + \cdots + c_{\mu k}p_k = \sqrt{k}d_{min} \quad (22b)$$

Since $c_{\mu n}$ is either +1 or –1 ($\forall n$ = 1, 2, ..., k), we can say that a point $\mathbf{p}_i$ that satisfies

$$\sum_{n=1}^{k} |p_{in}| = \sqrt{k}d_{min} \quad (23)$$

must lie on at least one of these supporting hyperplanes. For example, if

$$p_{in} \begin{cases} < 0 & \text{for } n = u, v, 1 \le u, v \le k \\ \ge 0 & \text{otherwise} \end{cases}$$

and $\mathbf{c}_\beta$ is a vector such that

$$c_{in} = \begin{cases} \mathbf{-1} & \text{for } n = u, v, 1 \le u, v \le k \\ +1 & \text{otherwise} \end{cases}$$

then, following the definition of set $\mathcal{C}$ (eqn. 18),

$$\boldsymbol{cp} \in \mathcal{C}$$

and we have

$$\mathbf{c}_\beta \mathbf{p}_i^T = \sum_{n=1}^{k} |p_{in}| = \sqrt{k}d_{min}$$

The above relation gives the equation for one of the supporting hyperplanes. Hence, the vector $\mathbf{p}_i$ is located on one of the hyperplanes defined by eqn. *22a* and *b*. Therefore, a point $\mathbf{p}_j$ that satisfies

$$\sum_{n=1}^{k} |p_{jn}| < \sqrt{k}d_{min} \quad (24)$$

must be located in the space enclosed by these supporting hyperplanes.

Hence, we conclude that the hypersphere at the origin with radius d_{min} is enclosed by 2" hyperplanes and any point $\mathbf{p}$ in the d_{min} neighbourhood about the origin must satisfy the inequality eqn. 24.

With the above results in hand, we now describe our proposed algorithm. For an input vector $\mathbf{x}_i$, a probable nearby codeword $\mathbf{y}_{l1}$ is found and is taken to be the current best match $\mathbf{y}_l$. **A** vector $\mathbf{y}_j$ will be nearer to **x,** than $\mathbf{y}_l$ if **y,** is in the d_{min} neighbourhood about $\mathbf{x}_i$, where $d_{min} = d(\mathbf{x}_i, \mathbf{y}_l)$. The first step is the same as in the ENNS algorithm where we select only those codewords whose mean values range from $\boldsymbol{m_{,,}}$ (eqn. 10) to m_{max} (eqn. 9). Following the discussion above we see that the hypersphere centred at **x,** with radius d_{min} can be enclosed by 2^k hyperplanes given as

$$\mathbf{c}_\mu(\mathbf{p} - \mathbf{x}_i)^T = \sqrt{k}d_{min}, \forall \mu = 1, 2, \ldots, 2^k \quad (25)$$

and any point $\mathbf{y}_j$ in the d_{min} neighbourhood about **x,** must satisfy

$$\sum_{n=1}^{k} |y_{jn} - x_{in}| < \sqrt{k}d_{min} \quad (26)$$

Thus, in the second step we can reject those codewords which do not satisfy the above inequality. If $\mathbf{y}_j$ is not rejected, $d(\mathbf{x}_i, \mathbf{y}_j)$ is calculated in the third step. If $d(\mathbf{x}_i, \mathbf{y}_j) \ge d_{min}$, $\mathbf{y}_j$ is rejected. Else $\mathbf{y}_j$ is taken as the current best match, and all the above steps are repeated. This guarantees the progressive confinement of the search space.

Now, the term $|y_{jn} - x_{in}|$ in inequality eqn. 26 is the absolute value of the difference between the nth entities of the vectors $\mathbf{y}_j$ and **x,.** Therefore, $\Sigma_{n=1}^{k} |y_{jn} - x_{in}|$ may be termed as the 'sum of the absolute differences' (SAD) between $\mathbf{y}_j$ and $\mathbf{x}_i$, and we can write

$$SAD(\mathbf{x}_i, \mathbf{y}_j) = \sum_{n=\mathbf{I}}^{k} |y_{jn} - x_{in}| \quad (27)$$

This means that, in the second step, codewords are selected on the basis of the sum of the absolute differences (SAD) criterion. **A** codeword $\mathbf{y}_j$ is selected if it satisfies the inequality

$$SAD(\mathbf{x}_i, \mathbf{y}_j) < SAD_{max} \quad (28)$$

where

$$SAD_{,,,} = \sqrt{k}d_{min} \quad (29)$$

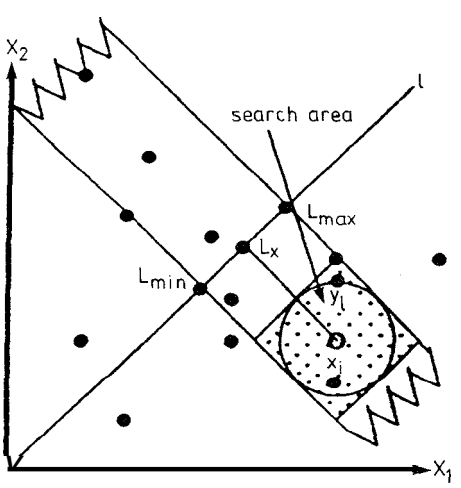


Fig.3 *Proposed search algorithm*

3.7 *The algorithm*

(1) In the first step of our algorithm a probable nearby codeword $\mathbf{y}_{l1}$ is guessed based on the minimum mean

difference criterion or by prediction; d_{min}, m_{min}, m_{max} and SAD_{min} are calculated.

(2) For each codeword $\mathbf{y}_j$ we check if m_{yj} is between m_{min} and m_{max}. If not then $\mathbf{y}_j$ is rejected, else we proceed to the next step.

(3) Next we check if $SAD(\mathbf{x}_i, \mathbf{y}_j) < SAD_{min}$. If not then $\mathbf{y}_j$ is rejected, thus discarding those codevectors which are far away from $\mathbf{x}_i$, resulting in a reduced search space containing the hypersphere centred at $\mathbf{x}_i$ with radius d_{min}. For $k = 2$, the search space is a square enclosing the circle centred at $\mathbf{x}_i$ with radius d_{min} as shown in Fig. 3.

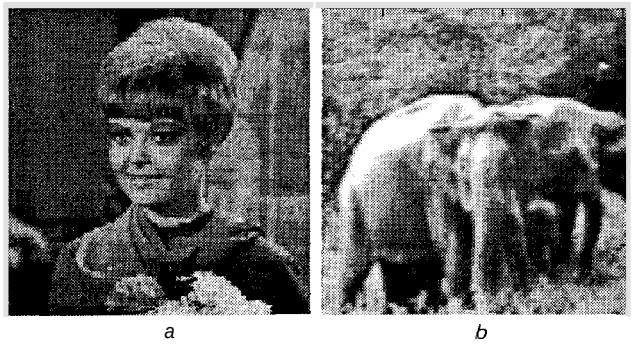
(4) If $\mathbf{y}_j$ is not rejected in the third step, then $d(\mathbf{x}_i, \mathbf{y}_j)$ is calculated. If $d(\mathbf{x}_i, \mathbf{y}_j) < d_{min}$, then the current closest codeword to $\mathbf{x}_i$ is taken as $\mathbf{y}_j$ with d_{min} set to $d(\mathbf{x}_i, \mathbf{y}_j)$, and m_{max}, m_{min} and SAD_{min} are updated accordingly. The procedure is repeated until we arrive at the best match for $\mathbf{x}_i$.

3.2 Merits of the algorithm

The performance of the algorithm is the same as that of the full-search LBG algorithm, although the search space is reduced to a great extent, since the search space encloses the hypersphere that contains the best match codeword. Hence, there is considerable saving in search time without any added degradation to the reconstructed picture quality.

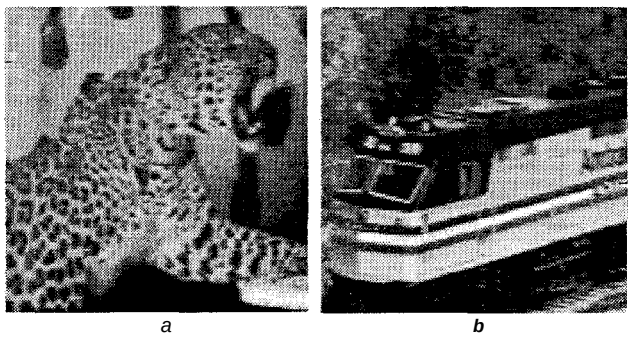
The merits of this algorithm over that described in [15] are:

(1) The search space is confined within the region enclosed by 2^k hyperplanes tangential to the hypersphere, thus reducing the search space to a great extent. From Figs. 2 and 3, we observe that, for a two-dimensional case, the search space in our method is exactly half of that in the Lee and Chen method.



a b

Fig.4 *Images used for simulation*
a Girl
h Elephant



a b

Fig.5 *Images used for simulation*
a Cheetah
h Train

(2) Only the mean values of the codewords are calculated and stored. This implies savings in computation in the codebook preprocessing stage, as no multiplication operation is involved, and also savings in the memory requirement.

3.3 Methods of further improvement

In order to achieve further efficiency in speed, the idea of early termination may be incorporated in the second and third steps as discussed below:

(1) While calculating $SAD(\mathbf{x}_i, \mathbf{y}_j)$ in the second step if

$$\sum_{n=1}^{\hat{k}} |x_{in} - y_{jn}| \geq \sqrt{k} d_{min} \qquad \text{for } 1 \leq \hat{k} < k$$

i.e. the accumulated difference for the first k entities is larger than (or equal to) $\sqrt{k} d_{min}$, reject $\mathbf{y}_j$.

(2) While calculating the Euclidean distance $d(\mathbf{x}_i, \mathbf{y}_j)$ if

$$\sum_{n=1}^{k} (x_{in} - y_{jn})^2 \geq d_{min}^2 \qquad \text{for } 1 \leq \hat{k} < k$$

i.e. the accumulated distortion for the first k entities is larger than (or equal to) d_{min}^2, reject $\mathbf{y}_j$.

Table 1: Computational efficiency of the four search algorithms

Codebook size	Search algorithm	DIST	Number of operations		
			Mult.	Add.	Comparisons
128	LBG	128	512	896	127
	ENNS	8.2	32.8	57.4	25.6
	Modified ENNS	5.3	21.2	44.2	31.5
	Proposed	3.0	12.0	71.0	27.6
256	LBG	256	1024	1792	255
	ENNS	9.6	38.4	67.2	29.8
	Modified ENNS	5.7	22.8	46.1	32.9
	Proposed	3.2	12.8	83.0	32.0
512	LBG	512	2048	3584	511
	ENNS	10.2	40.8	71.4	32.1
	Modified ENNS	6.3	25.2	44.6	33.2
	Prooosed	3.3	13.2	87.4	34.4

Table 2: Comparison when vector quantising the 'girl' image

Codebook size	Search algorithm	DIST	Number of operations		
			Mult.	Add.	Comparisons
128	LBG	128	512	896	127
	ENNS	7.4	29.6	51.8	23.9
	Modified ENNS	4.5	18.0	38.4	28.8
	Proposed	2.7	10.8	64.1	25.7
256	LBG	256	1024	1792	255
	ENNS	8.6	34.4	60.2	27.8
	Modified ENNS	5.0	20.0	42.5	31.6
	Proposed	2.9	11.6	73.7	29.8
512	LBG	512	2048	3584	511
	ENNS	7.6	30.4	53.2	26.7
	Modified ENNS	4.9	19.6	42.3	32.8
	Proposed	3.0	12.0	67.4	28.7

4 Simulation results

The computational efficiency of the proposed algorithm in codebook design, in comparison to LBG, ENNS and modified ENNS (by Lee and Chen) algorithms, was evaluated. The set of training vectors were extracted from four monochrome images ('girl', 'cheetah', 'elephant' and 'train') with size 128 pixels x 128 lines and 8 bits per pixel (Figs. 4 and 5). The vector dimension was taken as 4 (image blocks of size 2 pixels x 2 lines) and the distortion threshold as 0.0001. The comparison was done in terms of the average number of distance calculations (DIST) and various operations involved in the search process per input vector, for codebooks of different sizes, as shown in Table 1. Table 2 shows the comparison when vector quantising the 'girl' image using the generated codebooks. It is seen that the number of distance calculations in our case is significantly reduced at the cost of some extra additions only. The coding quality remains the same for all the algorithms since they are full-search equivalent, as mentioned earlier.

5 Conclusions

The fast equal-average nearest neighbour search algorithm for vector quantisation presented here uses the Euclidean distance, the sum norm (sum of the absolute differences) and the mean as the three measures for searching for the best representation of an input vector in a small search region in the codebook. Once an initial guess for the probable nearest codeword is made and codewords which are closer to the input vector are obtained in the subsequent stages, the search is reduced significantly in comparison to other existing fast codeword search algorithms, without the introduction of any extra distortion compared to the full-search algorithm. Simulation results show that the algorithm performs better than LBG, ENNS and the modified ENNS algorithms in terms of speed, and preprocessing cost. Also, as pointed out earlier, since only the mean values of the codewords in the codebook are stored, there is a considerable saving in the memory requirement in our proposed algorithm.

6 References

1 GRAY, R.M.: 'Vector quantization', *IEEE ASSP Mag.*, 1984, **1,** pp. 4–29
2 LINDE, Y., BUZO, A., and GRAY, R.M.: 'An algorithm for vector quantizer design', *IEEE Trans.*, 1980, **COM–28,** pp. 84–95
3 GERSHO, A., and YANO, M.: 'Adaptive vector quantisation by progressive codevector replacement'. Proceedings of the IEEE ICASSP, 1985, Vol. 1, pp. 133–136
4 GOLDBERG, M., and SUN, H.: 'Image sequence coding using vector quantization', *IEEE Trans.*, 1986, **COM–34,** pp. 703–710
5 GOLDBERG, M., and SUN, H.: 'Frame adaptive vector quantization for image sequence coding', *ZEEE Trans.*, 1988, **COM–36,** pp. 629–635
6 CHEN, O.T.-C., SHEU, B.J., and ZHANG, Z.: 'An adaptive vector quantizer based on the gold-washing method for image compression', *IEEE Trans. Circuits Syst. Video Technol.*, 1994, **4,** pp. 143–157
7 BEI, C.D., and GRAY, R.M.: 'An improvement of the minimum distortion encoding algorithm for vector quantisation', *IEEE Trans.*, 1985, **COM–33,** pp. 1132–1133
8 NGWA-NDIFOR, J., and ELLIS, T.: 'Predictive partial search algorithm for vector quantization', *Electron. Lett.*, 1991, **27,** pp. 1722–1723
9 HUANG, S.H., and CHEN, S.H.: 'Fast encoding algorithm for VQ-based image coding'. *Electron. Lett.*, 1990, ***26,*** pp. 1618–1619
10 ORCHARD, M.T.: 'A fast nearest-neighbour search algorithm'. Proceedings of the IEEE ICASSP, 1991, pp. 2297–2300
11 LI, W., and SALARI, E.: 'A fast vector quantization encoding method for image compression', *ZEEE Trans. Circuits Syst. Video Technol.*, 1995, ***5,*** pp. 119–123
12 HSIEH, C.H., LU, P.C., and CHANG, J.C.: 'Fast codebook generation algorithm for vector quantization of images', *Pattern Recognit. Lett.*, 1991, **12,** pp. 605–609
13 POGGI, G.: 'Fast algorithm for full-search VQ encoding', *Electron. Lett.*, 1993, **29,** pp. 1141–1142
14 GUAN, L., and KAMEL, M.: 'Equal-average hyperplane partitioning method for vector quantization of image data', *Pattern Recognit. Lett.*, 1992, **13,** pp. 693–699
15 LEE, C.-H., and CHEN, L.-H.: 'Fast closest codeword search algorithm for vector quantization', *IEE Proc. Vis. Image Signal Process.*, 1994, **141,** pp. 143–148
16 GHOSH, D., and SHIVAPRASAD, A.P.: 'Fast equal average nearest neighbour codeword search algorithm for vector quantization'. Proceedings of the National Conference on Communication (NCC'96), 1996. Bombay, India, pp. 154–157
17 HADLEY, G.: 'Linear algebra' (Narosa Publishing House, New Delhi, India, 1987)