Fast Encoding Algorithms for Vector Quantization Based on Orthogonal Transform

Jiann-Der Lee and Yaw-Hwang Chiou

***Abstract*—**For vector quantization (VQ), it is extremely time-consuming to extract the similar codeword with input vector during the encoding process. In this paper, three efficient algorithms are proposed to extract the features of input vector using orthogonal transform, i.e., PCA transform, Hadamard transform, Haar wavelet transform, respectively. These features are then used to early remove impossible codeword in the distortion computations stage. From the experimental results, it is shown that the proposed approaches can largely decrease the computation time for achieving VQ coding with the same quality with full search algorithm. More specifically, compared with the DHSS algorithm, the proposed algorithm reduces the computational time by 31% to 61%. Compared with the Pan's algorithm, the proposed algorithm reduces the computational time by 62% to 75%. Compared with the Lai's algorithm, the proposed algorithm reduces the computational time by 48% to 58%. Compared with the HTPDE algorithm, the proposed algorithm reduces the computational time by 27% to 44%. Compared with the WTPDE algorithm, the proposed algorithm reduces the computational time by 21% to 45%. Moreover, the computation time of the HWT-based approach is less than all other previous algorithms.

***Keywords*—**Vector quantization, PCA transform, Hadamard transform, Haar wavelet transform, Image coding.

I. Introduction

Vector Quantization (VQ)[1] is one of high performance and popular methods for data compression. In the past, it has been wildly used in various applications, e.g., image compression [2]-[3], watermarking [4], and image filtering [5], etc. However, traditional VQ methods require huge computing time to accomplish the encoding process and this factor limit its use for practical applications. To reduce the computing time in encoding stage, a lot of fast algorithms have been proposed [6]-[18] during the last decade. Basically, these approaches can obtain the almost same quality as full-search methods but much less time required. For example, the partial distance elimination (PDE) algorithm [6] can early stop the distortion computations if the partial distance between the input block and codeword excess the existed distance. Chung et al. [12] improved the PDE to increase the search efficiency. The equal-average nearest-neighbor search (ENNS) algorithm [7] removed the non-similar codeword by using the distance between the sum of input vector and the sum of codeword. The dynamical hyperplanes shrinking search (DHSS) algorithm [8] employed three projections of the input vector to reject unlike codewords and then speed up the search process. PAN *et al.* algorithm [9] also uses the sum and the variance of input vector and a segment of the codeword or whole codeword to get rid of impossible codeword. The method proposed by Lai and Liaw [14] utilizes the characteristic of the input vector, e.g., mean value, edge strength, and texture strength, to filter un-similar codewords. Lu *et al*. [19] employed the PDE technique to achieve an efficient VQ in Hadamard transform domain, namely (HTPDE). Additionally, another approaches [10]-[11] used the mean (sum) of image block and codeword to build a pyramid structure, respectively. With coarse-to-fine strategy, this method can gradually remove the false codewords.

As illustrated above, for VQ, high dimension of the input vectors and large codebooks are key factors of why huge computation time is unavoidable. Therefore, most of the previous approaches for fast VQ algorithms are to remove unlike codewords as early as possible during encoding stage. Hence, to achieve the goal of speed up the searching process, this work proposes three novel VQ encoding algorithms by using orthogonal transform to obtain the reduced features, and then increase the system performance. These algorithms not only reduce the search range for codebook, but also the dimension of the input vector and the complexity. Experimental results show that the performance of the proposed methods is superior to other previous methods in computation time.

The remainder of this paper is organized as following. Section II presents the details of the proposed algorithms which first introduce the brief concept of VQ, the theory of orthogonal transform and then illustrate the steps used. The experimental results and performances comparison of our methods and previous approaches are included in section III. Section IV is conclusions.

II. The proposed methods

A. Brief concept of VQ

The first step for VQ is to build a codebook from a set of training image and the elements of a codebook are named codewords. Generally, LBG algorithm [20] is commonly used to produce the desired codebook. Once the codebook is ready, the non-overlapped image blocks of an image can be encoded with the nearest codeword such that the total storage for the

The work was partially supported by National Science Council, Republic of China, Taiwan, under Grant NSC98-2220-E-182-002.

Jiann-Der Lee and Yaw-Hwang Chiou are with the Department of Electrical Engineering, Chang Gung University, Tao-Yuan, Taiwan, Republic of China. (e-mail: jdlee@mail.cgu.edu.tw)

image is minimum. In brief, given one codeword $C_j = (c_{j1}, c_{j2}, \ldots, c_{jk})$ and the image block $X = (x_1, x_2, \ldots, x_k)$, the squared Euclidean distance can be expressed as (1).

$$e^2 = D(X, C_j) = \sum_{i=1}^{k} (x_i - c_{ji})^2 \quad (1)$$

Refer to (1), it is clear that calculation of the distance between an image block and a codeword needs k multiplications, *k* subtractions, and *k-1* additions. Thus, it is extremely time-consuming for Full Search (FS) algorithm to encode an image block due to perform *N* iterations of (1) and *N-1* comparisons for a codebook with *N* codewords. In other words, the whole computation for encoding an image block requires $N * k$ multiplication, $N * k$ subtraction, $N * (k-1)$ additions, and $N-1$ comparisons. However, when the encoding process is accomplished, each image block is only represented with an index of the nearest codeword and this representation can significantly reduce the total memory storage.

B. *Fast Encoding Algorithm I based on PCA*

PCA is a well-known technique used for data reduction, especially in image compression. For example, PCA is commonly used to transform a random vector $X(x_1, x_2 \cdots x_d)$ to a simplified vector $Y(y_1, y_2 \cdots y_m)$, where $m \leq d$. That is, through PCA, a number of related variables are transformed to a smaller set of uncorrelated variables. At the same time, vector X has many projection axes, in which the axis with the largest variance is called as first axis, and the axis perpendicular to the first axis with the second largest variance is named as second axis, and so on [21]-[22].

To derive the transformation of PCA, vector X was first centralized. The covariance matrix C_x of X is then expressed as

$$C_x = E(XX^t) \quad (2)$$

where C_x is a positive symmetric matrix. Using C_x, the eigenvalues, denoted as ($\lambda_1, \lambda_2, \cdots, \lambda_d$) in descending order and their corresponding eigenvectors, denoted as { $w_1, w_2, \cdots, w_d$ }, can be obtained. Let *W* is an orthonormal matrix constructed with $w_1, w_2, \cdots, w_d$. The row vectors of W constitute an orthonormal basis. Since $W^{-1} = W^t$, we then express PCA as (3).

$$Y = WX \quad (3)$$

According to the linear algebra, y_k is written as

$$y_k = \sum_{i=1}^{d} w_{ki} x_i = w_k X \quad (4)$$

where y_k is the *k*-th principal component of vector X. Note that these principal components are uncorrelated and vector X can be reconstructed using (5) completely.

$$X = W^t Y \quad (5)$$

For data reduction, we use m $(m \leq d)$ principal components to reconstruct vector X as indicated by (6), and the mean square error (MSE) between vector X and the reconstructed vector $\tilde{X}$ is expressed as (7).

$$\tilde{X}_k = \sum_{i=1}^{m} W_k^t y_i \quad (6)$$

$$J_{MSE}^{PCA} = \sum_{i=m+1}^{d} \lambda_i \quad (7)$$

The projection values of the codewords in the codebook on first axis, 2nd axis, and 3rd axis are calculated. Let the projection values in the orthogonal coordinate be z1, z2, and z3, respectively. If the projection value on the first axis of the codeword C_c is the closest to z1 for all codewords, then it is chosen as the initial codeword. Next, we calculated the Euclidean distance *r* between the input vector X and the codeword C_c. If projection value of any codeword on 1st axis, 2nd axis, or 3rd axis outsides the interval [z1-r, z1+r], [z2-r, z2+r], or [z3-r, z3+r], this codeword is rejected directly in the encoding process. Whereas, the Euclidean distance between the codeword and the input vector X will be calculated and the value *r* will be updated by the smaller distance so that the search range is dynamically reduced [8]. Based on this strategy, a PCA-based method is proposed as below.

Let X_i is an input vector，X_c is the nearest codeword compared to input vector X_i so far, and X_{cz} is another codeword to be compared in the codebook. The dimensions of these three vectors are all the same, i.e., *k*. By applying PCA to these three vectors, we obtain Y_i, Y_c and Y_{cz}, respectively. It is noted that a vector after PCA transform, its orientation in feature space is changed, but the property of the vector is remained as the same. That is, we can use (8) to calculate the distance of X_i and X_c.

$$\begin{aligned} e^2 &= \|X_c - X_i\|^2 = \|Y_c - Y_i\|^2 \\ &= \sum_{j=1}^{k} (y_{cj} - y_{ij})^2 \end{aligned} \quad (8)$$

Additionally, if one codeword X_{cz} in the codebook satisfies (9), then it is considered as a possible nearest codeword. Otherwise, X_{cz} is rejected in encoding process.

$$\begin{aligned} e^2 &= \sum_{j=1}^{k} (y_{cj} - y_{ij})^2 > \sum_{j=1}^{k} (y_{czj} - y_{ij})^2 \\ &\geq \sum_{j=n}^{m} (y_{czj} - y_{ij})^2 \mid (m-n) < (k-1) \end{aligned} \quad (9)$$

For (9), let *m* = *n* = 1, we have

$$y_{i1} - e \leq y_{cz1} \leq y_{i1} + e \quad (10)$$

$$y_{i2} - e \leq y_{cz2} \leq y_{i2} + e \quad (11)$$

$$y_{i3} - e \leq y_{cz3} \leq y_{i3} + e \quad (12)$$

In other words, during the encoding process, if the values of three principal components of a codeword do not fall into the range indicated as (10)-(12), this codeword should be discarded. This criterion is similar to [8], in which the authors declaim three principal component is enough to reject most of unlike codewords. But, we found more principal components are usually required while deal with high-details images.

Next, substitute ($n = 1, m < k$) to (9), then we get

$$e^2 \geq \sum_{j=1}^{m} (y_{czj} - y_{ij})^2 \tag{13}$$

Here, (13) can be used to remove the impossible codewords that cannot be rejected by (10)-(12). As illustrated before, one can use a few principal components to represent the original vector without loss its characteristics. So if we can utilize principal components as less as possible, most of unnecessary codewords that do not locate in these principal axes can be early removed and then significantly reduce the time for distortion computations. The MSE shown as (7) is used to calculate the difference of origin vector and reconstructed vector with m principal components. Generally, the variances among these codewords located in first principal component are larger than others on 2nd principal component or 3rd principal component. Hence, if a codeword satisfies (10), then it may also fit the requirement of (11) and (12). Based on this concept, we use (10) to decide the search range for possible codewords, and utilize (13) to reject impossible codewords. This strategy is efficient to speed up the search process than [8].

In summary, this PCA-based algorithm is described as below.

1). Off-line preprocessing

Step 1.1: Derive the transform matrix *W* using the training images.

Step 1.2: Calculate the projections of the codewords in a codebook on first *m* axes. According to the projection value on the first axis, rearrange them in the ascending order.

2). On-line processing

Step 2.1: Calculate the projections of the input vector X_i on first *m* axes, denote them as y_{i1}, y_{i2}, …, and y_{im}.

Step 2.2: Use the 1st principal component of X_i to search the nearest codeword in the codebook. Denote the nearest codeword as ($X_{c\,idx\_\min}$) and *idx_min* is the corresponding index of X_i.

Step 2.3: Calculate the squared Euclidean distance e^2 of X_i and $X_{c\,idx\_\min}$. Set *idx_L* = *idx_min* - 1, *idx_U* = *idx_min* + 1, *codebook_num* = codebook size, and $L = U = 1$.

Step 2.4: If *L* = 0, go to step 2.9.

Step 2.5: From (10), if $(y_{cidx\_L1} - y_{i1})^2 > e^2$ then *L* = 0 (i.e, discard the codewords range from *idx_L* to *1* in the codebook), go to step 2.9.

Step 2.6: From (13), if $\sum_{j=1}^{m}(y_{c\,idx\_L\,j} - y_{i\,j})^2 > e^2$, then discard the *idx_L*-th codeword in the codebook, go to step 2.8.

Step 2.7: Calculate the squared Euclidean distance en^2 of X_i and $X_{c\,idx\_L}$. If $en^2 < e^2$ then set the index of X_i as *idx_L*, and $e^2 = en^2$

Step 2.8: *idx_L* = *idx_L - 1*, if *idx_L < 1*, set *L*=0。

Step 2.9: if *U* = 0, go to step 2.14.

Step 2.10: From (10), if $(y_{c\,idx\_U1} - y_{i1})^2 > e^2$ then set *U* = 0 (i.e, discard the codewords range from *idx_U* to *codebook_num* in the codebook), go to step 2.14.

Step 2.11: For (13), if $\sum_{j=1}^{m}(y_{c\,idx\_U\,j} - y_{i\,j})^2 > e^2$ then discard the *idx_U*-th codeword in the codebook, go to step 2.13.

Step 2.12: Calculate the squared Euclidean distance en^2 of X_i and $X_{c\,idx\_U}$. If $en^2 < e^2$ then set the index of X_i as *idx_U*, and $e^2 = en^2$

Step 2.13: *idx_U* = *idx_U* + 1, If *idx_U* > *codebook_num*, then set *U*=0.

Step 2.14: If (*U* + *L*) > 0, go to step 2.4.

Step 2.15: Check if there is any input vector to be encoded, if true, go to step 2.1.

Step 2.16: End of the process.

C. Fast Encoding Algorithm II based on HT

Let W_n be the $2^n * 2^n$ Hadamard matrix with elements in the set {1,-1}, and

$$W_1 = \frac{1}{\sqrt{2}}\begin{Bmatrix} 1 & 1 \\ 1 & -1 \end{Bmatrix} \quad \text{and} \quad W_{n+1} = \frac{1}{\sqrt{2^{n+1}}}\begin{Bmatrix} W_n & W_n \\ W_n & -W_n \end{Bmatrix}$$

Similarly, a vector X with dimension *k* after HT can be expressed as $Y = WX$, where the row vectors of *W* constitute an orthonormal basis. For vectors X_1, X_2, their corresponding vectors after HT are denoted as Y_1, Y_2, respectively and the distance of vector X_1, X_2 can also be expressed as (8). If one codeword X_{cz} in the codebook satisfies (9), then it is considered as a possible nearest codeword. Otherwise, X_{cz} is rejected in encoding process. In this approach, refer to (9), let *m* = *n* = *1*, we have

$$y_{i1} - e \leq y_{cz1} \leq y_{i1} + e \tag{14}$$

Next, substitute (*n* = 1, *m* = *k/2*) and (*n* = *k/2+1*, *m* = *k*) to (9), respectively, then we get

$$e_1^2 = \sum_{j=1}^{k/2}(y_{czj} - y_{ij})^2 \tag{15}$$

$$e_2^2 = \sum_{j=k/2+1}^{k}(y_{czj} - y_{ij})^2 \tag{16}$$

During the encoding process, the search range in the codebook is firstly defined by (14), and each codeword located in this search range is evaluated by (15) and (16) to determine whether it will be removed or not. In this approach, the row vectors of a 16-dimensions Hadamard matrix used are shown as below.

W1=1/4*(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)
W2=1/4*(1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1)
W3=1/4*(1, 1, -1, -1, 1, 1, -1, -1, 1, 1, -1, -1, 1, 1, -1, -1)
W4=1/4*(1, -1, -1, 1, 1, -1, -1, 1, 1, -1, -1, 1, 1, -1, -1, 1)
W5=1/4*(1, 1, 1, 1, -1, -1, -1, -1, 1, 1, 1, 1, -1, -1, -1, -1)
W6=1/4*(1, -1, 1, -1, -1, 1, -1, 1, 1, -1, 1, -1, -1, 1, -1, 1)

W7=1/4*(1, 1, -1, -1, -1, -1, 1, 1, 1, 1, -1, -1, -1, -1, 1, 1)
W8=1/4*(1, -1, -1, 1, -1, 1, 1, -1, 1, -1, -1, 1, -1, 1, 1, -1)
W9=1/4*(1, 1, 1, 1, 1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1)
W10=1/4*(1, -1, 1, -1, 1, -1, 1, -1, -1, 1, -1, 1, -1, 1, -1, 1)
W11=1/4*(1, 1, -1, -1, 1, 1, -1, -1, -1, -1, 1, 1, -1, -1, 1, 1)
W12=1/4*(1, -1, -1, 1, 1, -1, -1, 1, -1, 1, 1, -1, -1, 1, 1, -1)
W13=1/4*(1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, 1)
W14=1/4*(1, -1, 1, -1, -1, 1, -1, 1, -1, 1, -1, 1, 1, -1, 1, -1)
W15=1/4*(1, 1, -1, -1, -1, -1, 1, 1, -1, -1, 1, 1, 1, 1, -1, -1)
W16=1/4*(1, -1, -1, 1, -1, 1, 1, -1, -1, 1, 1, -1, 1, -1, -1, 1)

In summary, the proposed HT-based algorithm is illustrated as below.

1. Off-line preprocessing

Step 1.1: According to various codebook sizes, select a suitable Hadamard matrix *H*.

Step 1.2: Calculate the projections of the codewords in a codebook on each axis. According to the projection value on the first axis, rearrange them in the ascending order.

2. On-line processing

Step 2.1: Calculate the projections of the input vector X_i on each axis, denoted y_{i1}, y_{i2}, …and y_{ik}.

Step 2.2: Use the first element of X_i to search the nearest codeword in the codebook. Denote the nearest codeword as ($X_{c\,idx\_\min}$) and *idx_min* is the corresponding index of X_i.

Step 2.3: Calculate the squared Euclidean distance e^2 of X_i and $X_{c\,idx\_\min}$. Set *idx_L* = *idx_min* - 1, *idx_U* = *idx_min* + 1, *codebook_num* = codebook size and *L* = *U* = 1.

Step 2.4: If *L* = 0, go to step 2.9.

Step 2.5: From (14), if $(y_{c\,idx\_L1} - y_{i1})^2 > e^2$ then *L*=0 (i.e, discard the codewords range from *idx_L* to *1* in the codebook), go to step 2.9.

Step 2.6: From (15), if $e_1^2 = \sum_{j=1}^{k/2} (y_{c\,idx\_L\,j} - y_{i\,j})^2 > e^2$, then discard the *idx_L*-th codeword in the codebook, go to step 2.8.

Step 2.7: Calculate the squared Euclidean distance $en^2 = {e_1}^2 + \sum_{j=k/2+1}^{k} (y_{cidx\_L\,j} - y_{i\,j})^2$ of X_i and $X_{c\,idx\_L}$. If $en^2 < e^2$ then set the index of X_i as *idx_L*, and $e^2 = en^2$

Step 2.8: *idx_L* = *idx_L - 1*, if *idx_L* < *1*, set *L*=0。

Step 2.9: if *U* = 0, go to step 2.14.

Step 2.10: From (14), if $(y_{c\,idx\_U1} - y_{i1})^2 > e^2$ then set *U* = 0 (i.e, discard the codewords range from *idx_U* to *codebook_num* in the codebook), go to step 2.14.

Step 2.11: From (15), if $e_1^2 = \sum_{j=1}^{k/2} (y_{c\,idx\_U\,j} - y_{i\,j})^2 > e^2$ then discard the *idx_U*-th codeword in the codebook, go to step 2.13.

Step 2.12: Calculate the squared Euclidean distance $en^2 = {e_1}^2 + \sum_{j=k/2+1}^{k} (y_{cidx\_U\,j} - y_{i\,j})^2$ of X_i and $X_{c\,idx\_U}$. If $en^2 < e^2$ then set the index of X_i as *idx_U*, and $e^2 = en^2$

Step 2.13: *idx_U* = *idx_U* + 1, If *idx_U* > *codebook_num*, then set *U* = 0.

Step 2.14: If (*U* + *L*) > 0, go to step 2.4.

Step 2.15: Check if there is any input vector to be encoded, if true, go to step 2.1.

Step 2.16: End of the process.

D. *Fast Encoding Algorithm III based on HWT*

According to the algorithm of discrete orthogonal wavelet decomposition [23]-[24], an image $A_{2^{j+1}} f$ with resolution 2^{j+1} can be decomposed into four sub-images with resolution 2^j, i.e., $A_{2^j} f$, $D^1_{2^j} f$, $D^2_{2^j} f$, $D^3_{2^j} f$. Here, $A_{2^j} f$ denotes the low-frequency part of $A_{2^{j+1}} f$, $D^1_{2^j} f$ denotes the vertical high-frequency part of $A_{2^{j+1}} f$, $D^2_{2^j} f$ denotes the horizontal high-frequency part of $A_{2^{j+1}} f$, and $D^3_{2^j} f$ denotes the highest frequency part of $A_{2^{j+1}} f$, respectively. The block diagram of decomposition scheme is shown as Fig. 1. By repeatedly performing this decomposition scheme, we obtain the 2-D wavelet transform of an image. In other words, for any *J* > 0, an image can be decomposed into $3*J+1$ sub-images such as $(A_{2^{-J}} f, (D^1_{2^j} f, D^2_{2^j} f, D^3_{2^j} f))_{-J<=j<=-1}, {}_{J>0}$. For a vector X , its corresponding transformed vector Y with dimension *k* after HWT can be expressed as $Y = WX$, where W is an orthonormal Haar wavelet matrix and the row vectors of *W* constitute an orthonormal basis. The first element of *Y* is represented as $A_{2^{-J}} f$, the first segment of Y (K/4 elements) is denoted as $(A_{2^{-J}} f, (D^1_{2^j} f, D^2_{2^j} f, D^3_{2^j} f))_{-J<=j<-1}, {}_{J>0}$, the second segment of Y (K/4 elements) is denoted as $D^1_{2^{-1}} f$, the third segment of Y (K/4 elements) is denoted as $D^2_{2^{-1}} f$, and the forth segment of Y (K/4 elements) is denoted as $D^3_{2^{-1}} f$, respectively.

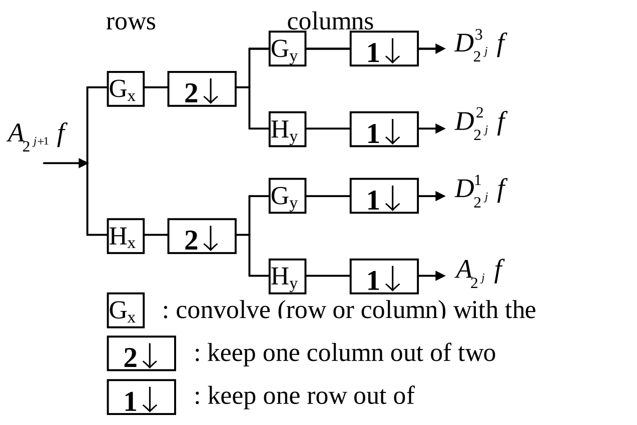

Fig. 1 Orthogonal wavelet decomposition.

For vectors X_1, X_2, their transformed vectors after HWT are denoted as Y_1 , Y_2 , respectively. And the distance of vector X_1, X_2 can be also expressed as (8). Similarly, for one codeword X_{cz} in the codebook, if it satisfies (9), then it is considered as a possible nearest codeword. Otherwise, X_{cz} is rejected in encoding process. Also from (9), let $m = n = 1$, the lowest-frequency part of the image is used, i.e., $A_{2^{-J}} f$, we have

$$y_{i1} - e \le y_{cz1} \le y_{i1} + e \tag{17}$$

In other words, during the encoding process, if the lowest-frequency part of a codeword did not fall into the range indicated as (17), this codeword should be discarded.

Similarly, substitute $n = 1$, $m = k/4$ to (9). The low-frequency portion of the image is used, i.e., $(A_{2^{-J}} f, (D^1_{2^j} f, D^2_{2^j} f, D^3_{2^j} f))_{-J<=j<-1}, {}_{J>0}$, then we get

$$e_1^{\,2} = \sum_{j=1}^{k/4} (y_{czj} - y_{ij})^2 \le e^2 \tag{18}$$

Substitute $n = 1$, $m = 2k/4$ to (9). The low-frequency portion and the vertical high- frequency portion $D^1_{2^{-1}} f$ are used, and then we have

$$e_2^{\,2} = e_1^{\,2} + \sum_{j=(k/4)+1}^{2(k/4)} (y_{czj} - y_{ij})^2 \le e^2 \tag{19}$$

Substitute $n = 1$, $m = 3k/4$ to (9). The low-frequency portion, the vertical high-frequency portion $D^1_{2^{-1}} f$ and the horizontal high-frequency portion $D^2_{2^{-1}} f$ are used, and we have

$$e_3^{\,2} = e_2^{\,2} + \sum_{j=(2k/4)+1}^{3k/4} (y_{czj} - y_{ij})^2 \le e^2 \tag{20}$$

During the encoding process, the search range in the codebook is firstly defined by (17), and each codeword located in this search range is evaluated by (18)-(20) to determine whether it will be removed or not.

In summary, the proposed algorithm based on HWT is described as below.

1. Off-line preprocessing

Calculate the projections of each codeword in a codebook on each axis, i.e., all the sub-images of each codeword, $(A_{2^{-J}} f, (D^1_{2^j} f, D^2_{2^j} f, D^3_{2^j} f)_{-J<=j<=-1}, {}_{J=2}$. According to the projection value on the first axis (the lowest frequency), rearrange them in the ascending order.

2. On-line processing

Step 2.1: Calculate the projections of the input vector X_i on each axis, denoted as y_{i1}, y_{i2}, …and y_{ik}.

Step 2.2: Use the first element of X_i to search the nearest codeword in the codebook. Denote the nearest codeword as ($X_{c\,idx\_\min}$) and *idx_min* is the corresponding index of X_i.

Step 2.3: Calculate the squared Euclidean distance e^2 of X_i and $X_{c\,idx\_\min}$. Set *idx_L = idx_min* - 1, *idx_U = idx_min* + 1, *codebook_num = codebook size* and *L* = *U* = 1.

Step 2.4: If *L* = 0, go to step 2.11.

Step 2.5: From (17), if $(y_{c\,idx\_L1} - y_{i1})^2 > e^2$ then *L*=0 (i.e, discard the codewords range from *idx_L* to *1* in the codebook), go to step 2.11.

Step 2.6: From (18), if $e_1^{\,2} > e^2$, then discard the *idx_L*-th codeword in the codebook, go to step 2.10.

Step 2.7: From (19), if $e_2^{\,2} > e^2$, then discard the *idx_L*-th codeword in the codebook, go to step 2.10.

Step 2.8: From (20), if $e_3^{\,2} > e^2$, then discard the *idx_L*-th codeword in the codebook, go to step 2.10.

Step 2.9: Calculate the squared Euclidean distance $en^2 = e_3^{\,2} + \sum_{j=(3k/4)+1}^{k} (y_{c\,idx\_L\,j} - y_{ij})^2$ of X_i and $X_{c\,idx\_L}$. If $en^2 < e^2$ then set the index of X_i as *idx_L*, and $e^2 = en^2$

Step 2.10: *idx_L = idx_L - 1*, if *idx_L < 1*,set *L*=0.

Step 2.11: if *U* = 0, go to step 2.18.

Step 2.12: From (17), if $(y_{c\,idx\_U1} - y_{i1})^2 > e^2$ then set *U* = 0 (i.e, discard the codewords range from *idx_U* to *codebook_num* in the codebook), go to step 2.18.

Step 2.13: From (18), if $e_1^{\,2} > e^2$ then discard the *idx_U*-th codeword in the codebook, go to step 2.17.

Step 2.14: From (19), if $e_2^{\,2} > e^2$ then discard the *idx_U*-th codeword in the codebook, go to step 2.17.

Step 2.15: From (20), if $e_3^{\,2} > e^2$ then discard the *idx_U*-th codeword in the codebook, go to step 2.17.

Step 2.16: Calculate the squared Euclidean distance $en^2 = e_3^{\,2} + \sum_{j=(3k/4)+1}^{k} (y_{c\,idx\_U\,j} - y_{ij})^2$ of X_i and $X_{c\,idx\_U}$. If $en^2 < e^2$ then set the index of X_i as *idx_U*, and $e^2 = en^2$

Step 2.17: *idx_U* = *idx_U* + 1, If *idx_U* > *codebook_num*, then set *U* = 0.

Step 2.18: If (*U* + *L*) > 0, go to step 2.4.

Step 2.19: Check if there is any input vector to be encoded, if true, go to step 2.1.

Step 2.20: End of the process.

III. Experimental results

To evaluate the performance of the proposed methods, three gray images (Fig. 2, Lena, Pepper, Baboon)[25] with the size of 512*512 are used in the experiment. The platform for computation we used is a general–purpose PC (Pentium D 3.00/3.00Ghz CPU, 512MB RAM). The vector dimension is 16 (image blocks of 4*4). The codebook is produced by LBG algorithm with "Lena" as the training image. We also compare this algorithm with previous remarkable algorithms, e.g., FS algorithm, ENNS algorithm, DHSS algorithm, Pan's algorithm, Lu's algorithm, Hwang's algorithms, and Lai's algorithms in terms of the average number of distance calculations and the computational time.

With various codebook sizes, the average number of distance calculations and the computational time are shown in Table I and Table II, respectively. The performance comparison is described as following:

(A) Fast Encoding Algorithm (based on PCA): From Tables I and II, it is clear that even the average number of distance calculations is reduced, but its computational time is not always smaller than others. The reason is that extra computation time is often needed for removing the unlike codewords such that total time consuming is increased. In the experiment, when the test images is “Lena” or ”Pepper”, and three principal components are adopted, the proposed algorithm have the best efficiency compared with previous algorithms in various codebook size. Otherwise, more principal components are required for “baboon” image which has much more details than the other images. More specifically, comparing with the DHSS algorithm, this proposed algorithm reduces the computational time by 7.14% to 19.64%. Comparing with the PAN’s algorithm, the proposed algorithm reduces the computational time by 47.67% to 64.86%. Comparing with the LAI’s algorithm the proposed algorithm reduces the computational time by 28.57% to 36.36%. When seven principal components are used, our algorithm has the best efficiency compared with previous algorithms in various codebook sizes for three test images. That is, comparing with the DHSS algorithm, when codebook size is 256, 512, and 1024, the proposed algorithm reduces the computational time by 0% to 39.46%. Comparing with the Pan’s algorithm the proposed algorithm reduces the computational time by 38.91% to 56.76%. Comparing with the Lai’s algorithm the proposed algorithm reduces the computational time by 15.79% to 36.36%.

(B) Fast Encoding Algorithm (based on HT): As illustrated in III(A), even the average number of distance calculations is less, but its computational time is not always smaller than others. In the experiment, the proposed algorithm has the best efficiency compared with previous algorithms in various codebook sizes for three test images. More specifically, comparing with the DHSS algorithm, the proposed algorithm reduces the computational time by 25% to 61%. Comparing with the Pan’s algorithm, the proposed algorithm reduces the computational time by 60% to 70%. Comparing with the Lu’s algorithm the proposed algorithm reduces the computational time by 20% to 40%. Comparing with the Lai’s algorithm the proposed algorithm reduces the computational time by 42% to 55%.

(C) Fast Encoding Algorithm (based on HWT): From the Table I, this algorithm has the best performance among the previous approaches in removing non-similar codewords. From the Table II, this algorithm in computation time has the best efficiency compared with previous algorithms in various codebook sizes for three test images. More specifically, comparing with the DHSS algorithm, the proposed algorithm reduces the computational time by 31% to 61%. Comparing with the Pan’s algorithm, the proposed algorithm reduces the computational time by 62% to 75%. Comparing with the Lu’s algorithm the proposed algorithm reduces the computational time by 27% to 44%. Comparing with the Hwang’s algorithm the proposed algorithm reduces the computational time by 21% to 45%. Comparing with the Lai’s algorithm the proposed algorithm reduces the computational time by 48% to 58%.

From the above comparison with the proposed three algorithms, it is clear that the computation time of the HWT-based approach is less than all other algorithms in all cases. It not only reduces the search range for codebook, but also the dimension of the input vector and the complexity.

(a)Lena (b)Pepper (c)Baboon

Fig. 2 Test images.

Table I. Comparison of the average numbers of distance computations.

Code-book size	Algorithm	image		
		Lena	Pepper	Baboon
128	FS	128	128	128
	ENNS	9.63	8.51	17.03
	PAN	2.58	2.34	4.80
	DHSS	3.2	2.88	7.80
	LAI	1.85	1.71	2.77
	HTPDE	2.42	2.28	5.07
	WTPDE	2.43	2.31	5.21
	Proposed(PCA 3pc)	2.25	1.86	9.51
	Proposed(PCA 7pc)	1.67	1.46	5.52
	Proposed(HT)	5.71	5.14	10.72
	Proposed(HWT)	1.47	1.41	2.03
256	FS	256	256	256
	ENNS	17.94	15.91	46.2
	PAN	3.97	3.94	12.07
	DHSS	4.72	4.50	20.03
	LAI	2.72	2.63	6.90
	HTPDE	3.88	3.77	13.12
	WTPDE	3.89	3.82	13.67
	Proposed(PCA 3pc)	3.73	3.00	19.47
	Proposed(PCA 7pc)	2.51	2.13	10.54
	Proposed(HT)	10.27	9.33	28.67
	Proposed(HWT)	2.07	2.09	4.04
512	FS	512	512	512
	ENNS	33.84	30.27	92.68
	PAN	6.32	5.97	21.35
	DHSS	7.78	7.72	39.49
	LAI	4.2	3.93	12.13
	HTPDE	6.34	6.24	25.18
	WTPDE	6.45	6.36	26.80
	Proposed(PCA 3pc)	6.04	5.51	35.54
	Proposed(PCA 7pc)	3.72	3.65	18.25
	Proposed(HT)	18.80	17.17	57.05
	Proposed(HWT)	2.65	2.69	6.10
1024	FS	1024	1024	1024
	ENNS	61.32	60.55	170.92
	PAN	10.41	11.05	35.1
	DHSS	12.80	14.43	72.18
	LAI	6.80	7.05	19.96
	HTPDE	10.63	11.58	45.05
	WTPDE	10.83	11.85	48.84

	Proposed(PCA 3pc)	10.12	10.61	66.19
	Proposed(PCA 7pc)	5.35	5.94	27.32
	Proposed(HT)	33.47	33.51	98.82
	Proposed(HWT)	3.58	4.04	9.10

Table II. Comparison of the computation time in milliseconds .

Code-book size	Algorithm	Image		
		Lena	Pepper	Baboon
128	FS	248	248	248
	ENNS	20	17	30
	PAN	40	36	47
	DHSS	16	15	27
	LAI	21	19	28
	HTPDE	15	14	23
	WTPDE	14	13	21
	Proposed(PCA 3pc)	14	13	35
	Proposed(PCA 7pc)	17	16	31
	Proposed(HT)	12	11	16
	Proposed(HWT)	11	9	14
256	FS	456	456	456
	ENNS	34	30	78
	PAN	49	46	81
	DHSS	23	22	61
	LAI	29	27	55
	HTPDE	20	19	48
	WTPDE	20	18	47
	Proposed(PCA 3pc)	21	17	64
	Proposed(PCA 7pc)	22	20	51
	Proposed(HT)	16	15	30
	Proposed(HWT)	13	13	27
512	FS	886	886	886
	ENNS	58	51	164
	PAN	64	59	129
	DHSS	35	33	110
	LAI	43	39	95
	HTPDE	30	27	78
	WTPDE	31	28	84
	Proposed(PCA 3pc)	29	27	107
	Proposed(PCA 7pc)	29	28	80
	Proposed(HT)	21	20	51
	Proposed(HWT)	19	18	46
1024	FS	1731	1731	1731
	ENNS	98	100	263
	PAN	91	89	202
	DHSS	54	56	195
	LAI	66	65	158
	HTPDE	47	48	129
	WTPDE	45	47	135
	Proposed(PCA 3pc)	46	45	185
	Proposed(PCA 7pc)	42	43	135
	Proposed(HT)	30	31	77
	Proposed(HWT)	28	28	76

IV. Conclusions

In this paper, three fast encoding algorithms based on orthogonal transform have been proposed to early reject unlikely codeword and increase the performance of VQ. That is, using the transformed vectors obtained by orthogonal transform, those non-similar codewords can be early removed and the computation time is then reduced. From the experimental results, the PSNR of this approach is the same as the full search method. Among these proposed algorithms, the one based on HWT has the best efficiency compared with previous algorithms in various codebook sizes for three test images. More specifically, comparing with the DHSS algorithm, this algorithm reduces the computational time by 31% to 61%. Comparing with the Pan's algorithm, this algorithm reduces the computational time by 62% to 75%. Comparing with the Lu's algorithm, this algorithm reduces the computational time by 27% to 44%. Comparing with the Hwang's algorithm, this algorithm reduces the computational time by 21% to 45%. Comparing with the Lai's algorithm, this algorithm reduces the computational time by 48% to 58%. In other words, the proposed approaches can significantly reduce the computation time and then speed up the search process in VQ.

References

[1] R. M. Gray, "Vector quantization," IEEE ASSP Mag., vol. 1, pp. 4–29, Apr.1984.

[2] N. M. Nasrabadi and R. A. King; "Image coding using vector quantization: a review," IEEE Trans. Communication, vol. 36, pp. 957 – 971, Aug. 1988.

[3] F. Rizzo, J. A. Storer and B. Carpentieri, "Overlap and channel errors in adaptive vector quantization for image coding," Information Sciences, vol. 171, pp. 125–143 2005.

[4] Z. M. Lu, D. G. Xu, and S. H. Sun, "Multipurpose image watermarking algorithm based on multistage vector quantization," IEEE Trans. Image Process., vol. 14, pp. 822–831, 2005.

[5] A. I. Gonzalez, M. Grana, J. R. Cabello, A. DAnjou, and F.X. Albizuri, "Experimental results of an evolution-based adaptation strategy for VQ image filtering," Information Sciences, vol. 133, pp. 249–266, 2001

[6] C. D. Bei and R. M. Gray, "An improvement of the minimum distortion encoding algorithm for vector quantization," IEEE Trans. Communication, vol. 33, pp. 1132–1133, Oct. 1985.

[7] S. W. Ra and J. K. Kim, "A fast mean-distance-ordered partial codebook search algorithm for image vector quantization," IEEE Trans. Circuits Syst. II: Analog and Digital Signal Process., vol. 40, pp. 576–579, Sept. 1993.

[8] S. C. Tai, C. C. Lai, and Y. C. Lin, "Two fast nearest neighbor searching algorithms for image vector quantization," IEEE Trans. Communication, vol. 40, pp. 1623–1628, Dec. 1996.

[9] J. S. Pan, Z. M. Lu, and S. H. Sun, "An efficient encoding algorithm for vector quantization based on subvector technique," IEEE Trans. Image Processing, vol. 12, pp. 265–270, Mar. 2003.

[10] C. H. Lee and L. H. Chen, "A fast search algorithm for vector quantization using mean pyramids of codewords," IEEE Trans. Communication, vol. 43, pp. 1697–1702, Feb. 1995.

[11] Z. Pan, K. Kotani, and T. Ohmi, "An improved fast encoding algorithm for vector quantization using 2-pixel-merging sum pyramid data structure," Pattern Recognition Letters, vol. 25, pp. 459–468, 2004.

[12] M. C. Chung, S.C. Chen; C.T. Yu, and P.Y. Chen, "An improvement of fast search algorithm for vector quantization," Proceedings of 2005 International Symposium on Intelligent Signal Processing and Communication Systems, pp 97-100, Dec. 2005.

[13] S. E. Qian, "Fast vector quantization algorithms based on nearest partition set search," IEEE Trans. Image Processing, vol. 15, pp. 2422–2430, Aug. 2006.

[14] Jim Z. C. Lai, and Yi-Ching Liaw, "Fast-searching algorithm for vector quantization using projection and triangular inequality," IEEE Trans. Image Processing, vol. 13, pp. 1554–1558, Dec. 2004.

[15] C. C. Wang and C. W. Tung, "Fast search algorithm for vector quantization using dynamic subvectors," Optical Engineering, vol. 45, no. 9, pp.1-10, Sep. 2006.

[16] Jim Z. C. Lai and Yi-Ching Liaw, "A Fast Search Algorithm for Mean-Removed Vector Quantization Using Edge and Texture Strengths of A Vector," Image and Vision Computing, vol. 23, pp. 739-746, August 2005.

[17] Shu-Chuan Chu, Zhe-Ming Lu and Jeng-Shyang Pan, "Hadamard transform based fast codeword search algorithm for high-dimensional VQ encoding," Information Sciences, vol. 177, pp. 734-746. 2007.

[18] Wen-Jyi Hwang, Sen-Shiang Jeng, and Biing-Yau Chen, “Fast codeword search algorithm using wavelet transform and partial distance search techniques,” Electronics Letters, vol. 33, pp. 365–366, 1997.
[19] Z. M. Lu, J. S. Pan, S. H. Sun, “Efficient codeword search algorithm based on Hardmard transform,” Electronics Letters, vol. 36, pp. 1364–1365, 2000.
[20] Y. Linde, A. Buzo, and R. M. Gray, “An algorithm for vector quantizer design,” IEEE Trans. Commun., vol. 28, no. 1, pp. 84-95, Jan. 1980.
[21] H. Hotelling, “Analysis of a complex of statistical variable into principal components,” J. Educ. Psysch., vol. 24, pp. 417-520, 1933
[22] Hyvarinen Aapo, Juha Karhunen, and Erkki Oja, Independent Component Analysis, Wiley, 2001.
[23] S. G. Mallat, “A theory for multiresolution signal decomposition: The wavelet representstion”, IEEE Trans. Pattern Analysis Machine Intelligence, vol. 11, pp. 674-693, 1989.
[24] P. Scheunders, “An orthogonal wavelet representation of multivalued images,” IEEE Trans. Image Processing, vol. 12, pp. 718–725, June 2003.
[25] Chang Yun Fah, Omar Mohd Rijal and Syed Abdul Rahman Abu Bakar, “Functional quality and performance metric for some image processing applications”, International Journal of Mathematical Models and Methods in Applied Science, vol. 2, issue 4, pp. 543-552, 2008.

Jiann-Der Lee was born in Tainan, Taiwan, in 1961. He received B.S., M.S., and Ph.D. degrees from the department of Electrical Engineering, National Cheng Kung University, Taiwan, in 1984, 1988, and 1992, respectively. He is currently a full professor and the head of the department of Electrical Engineering, Chang Gung University, Tao-Yuan. His current research interests include image processing, pattern recognition, computer vision, consumer electronics and VLSI CAD design.

Dr. Lee is a member of IEEE and IAPR, and is listed in the Who’s Who in the World, Who's Who in Finance and Industry. He received a number of investigator awards (e.g. from National Science Council, Taiwan, and Acer Foundations, Taiwan), the Excellent Teaching Award, 2002, and the Excellent Research Award, 2003, from the Chang Gung University.

Yaw-Hwang Chiou was born in Taichung, Taiwan, in 1958. He received the B.S. degree from Department of Electronic Engineering, National Taiwan University of Science and Technology, Taipei, Taiwan, in 1984 and the M.S. degree from Department of Electronic Engineering, Chung Yuan Christian University, Taoyuan, Taiwan, in 1992. During the years from 1992 to 1998, he jointed the Department of Electronic Engineering, Vanung University, Taiwan, as a lecturer. He is currently pursuing the Ph.D. degree at the Department of Electrical Engineering, Chang Gung University, Taiwan, where his research interests concern image coding and digital watermarking.