Journal of Information Hiding and Multimedia Signal Processing ©2010 ISSN 2073-4212
Ubiquitous International Volume **1**, Number **3**, July **2010**

A Survey of VQ Codebook Generation

Tzu-Chuen Lu
Department of Information Management
Chaoyang University of Technology
Taichung 41349, Taiwan, R.O.C.
tclu@cyut.edu.tw

Ching-Yun Chang
Computer Laboratory
University of Cambridge
Trinity Lane, Cambridge CB2 1TN,UK
Ching-Yun.Chang@cl.cam.ac.uk

Received March 2010; revised April 2010

Abstract. *One of the key roles of Vector Quantization (VQ) is how to generate a good codebook such that the distortion between the original image and the reconstructed image is the minimum. In the past years, many improved algorithms of VQ codebook generation approaches have been developed. In this paper, we present a snapshot of the recent developed schemes. The discussed schemes include mean-distance-ordered partial codebook search (MPS), enhance LBG (ELBG), neural network based techniques, genetic-based algorithms, principal component analysis (PCA) approaches, tabu search (TS) schemes, codeword displacement methods and so on.*
Keywords: Vector Quantization, LBG, ELBG, Principal Component Analysis, Tabu Search

1. Introduction.

Vector Quantization (VQ) is an efficient and simple approach for data compression. Since it is simple and easy to implement, VQ has been widely used in different applications, such as pattern recognition, image compression, speech recognition, face detection and so on [11].

For the purpose of image compression, the operations of VQ include dividing an image into several vectors (or blocks) and each vector is mapped to the codewords of a codebook to find its reproduction vector. In other words, the objective of VQ is the representation of vectors $X \subseteq R^k$ by a set of reference vectors $CB = \{C_1, C_2, \dots, C_N\}$ in R^k in which R^k is the k-dimension Euclidean space. CB is a codebook which has a set of reproduction codewords and $C_j = \{c_1, c_2, \dots, c_k\}$ is the j-th codeword. The total number of codewords in CB is N and the number of dimensions of each codeword is k.

There are three major procedures in VQ, namely codebook generation, encoding procedure and decoding procedure. In the codebook generation process, various images are divided into several k-dimension training vectors. The representative codebook is generated from these training vectors by the clustering techniques. In the encoding procedure, an original image is divided into several k-dimension vectors and each vector is encoded by the index of codeword by a table look-up method. The encoded results are called an index table. During the decoding procedure, the receiver uses the same codebook to translate the index back to its corresponding codeword for reconstructing the image.

Fig. 1 shows an example of encoding and decoding an image by VQ. In Fig. 1, an original image is divided into four blocks sized 2×2. Each block is then translated to a vector with four dimensions. The first vector $X_1 = (7, 10, 14, 6)$ is mapped to the 7th codeword of the codebook. The index of the 7th codeword replaces the vector to represent the block. When the receiver receives the index table, he uses the 7th codeword $c_7 = (9, 6, 9, 9)$ to reconstruct the first block. The final recovered image is called the reconstructed image.

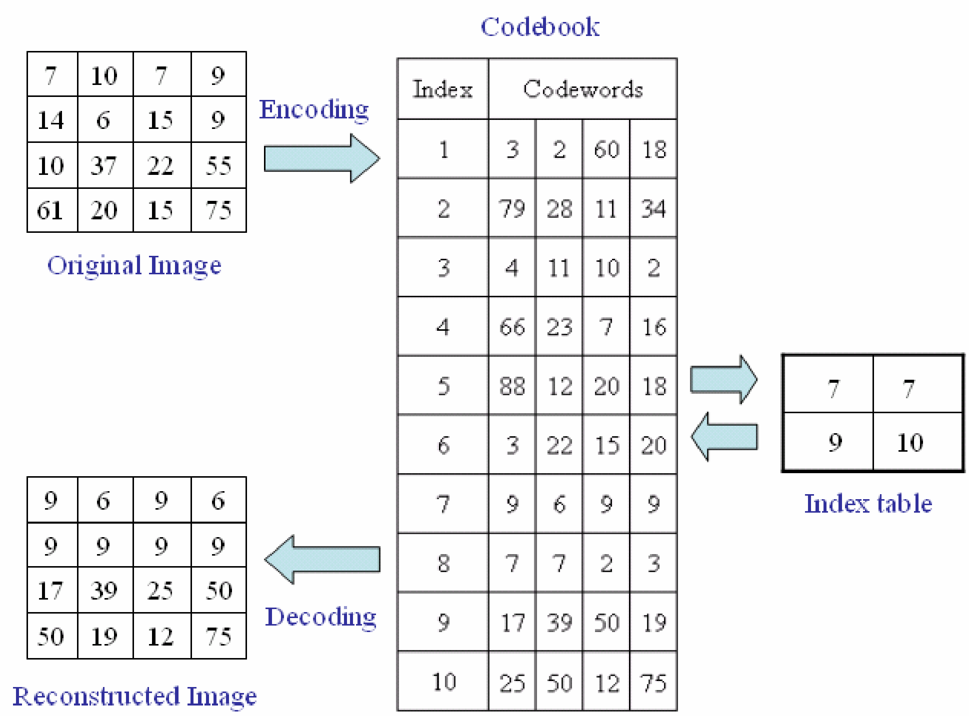

FIGURE 1. An Example of Encoding and Decoding by VQ

One of the key points of VQ is to generate a good codebook such that the distortion between the original image and the reconstructed image is the minimum. Moreover, since the codebook generation procedure is a time consuming process, how to reduce the computation time is another important issue for the VQ codebook generation. The most commonly used method in VQ is the Generalized Lloyd Algorithm (GLA) which is also called Linde-Buzo-Gary (LBG) algorithm. However, LBG has the local optimal problem, and the utility of each codeword in the codebook is low. The local optimal problem is that the codebook guarantees local minimum distortion but not global minimum distortion [4]. Therefore, many researchers have proposed different methods to solve the problems and speed up the process of finding the optimal solution. In this paper, we present a snapshot of the recent developed schemes. The discussed schemes include mean-distance-ordered partial codebook search (MPS), enhance LBG (ELBG), neural network based techniques, genetic-based algorithms, principal component analysis (PCA) approaches, tabu search (TS) schemes, codeword displacement methods and so on.

2. **The LBG Algorithm.** In 1980, Linde et al. proposed a Generalized Lloyd Algorithm (GLA) which is also called Linde-Buzo-Gary (LBG) algorithm. They used a mapping function to partition training vectors into N clusters. The mapping function is defined as $R^k \to CB$. Let $X = (x_1, x_2, \ldots, x_k)$ be a training vector and $d(X, Y)$ be the Euclidean

distance between any two vectors. The iteration of GLA for a codebook generation is given as follows:

Step 1: Randomly generate an initial codebook CB_0.
Step 2: $i = 0$.
Step 3: Perform the following process for each training vector.

- Compute the Euclidean distances between the training vector and the codewords in CB_i. The Euclidean distance is defined as

$$d(X, C) = \sqrt{\sum_{t=1}^{k} (x_t - c_t)^2} \tag{1}$$

- Search the nearest codeword among CB_i.

Step 4: Partition the codebook into N cells.
Step 5: Compute the centroid of each cell to obtain the new codebook CB_{i+1}.
Step 6: Compute the average distortion for CB_{i+1}. If it is changed by a small enough amount since the last iteration, the codebook may converge and the procedure stops. Otherwise, $i = i + 1$ and go to Step 3.

Here, we use five training vectors as an example to demonstrate how to train a codebook. The training vectors are shown in Fig. 2. Suppose the total number of the codewords in a codebook is three, namely $N = 3$. First, we randomly generate an initial codebook CB_0 as shown in Table 1.

Next, the scheme computes the distances between the training vector and the codewords among CB_0. For example, the distance between X_1 and C_1 is $d(X_1, C_1) = \sqrt{(241-32)^2 + (192-177)^2 + \ldots + (156-210)^2} = 248.41$. From Table 1, we can see that the nearest code of X_1 is C_3 for $i = 0$.

The training vectors which have the same nearest codeword are partitioned into the same cell. The scheme computes the centroid of each cell to obtain the new codebook. In this example, X_1 and X_4 that have the same nearest codeword C_3 are partitioned into the same cell. The centroid of the two vectors is $(203, 150, 88, 98.5)$ that is the third codeword of the new generated codebook CB_1. The procedure is repeated until the codebook is converged. The final codebook is CB_3.

	x_1	x_2	x_3	x_4
X_1	241	192	21	156
X_2	212	76	123	36
X_3	10	220	108	233
X_4	165	108	155	41
X_5	109	52	19	247

Figure 2. Five Training Vectors

LBG is an easy and rapid algorithm. However, it has the local optimal problem which is that for a given initial solution, it always converges to the nearest local minimum. In other words, LBG is a local optimization procedure. Therefore, scholars proposed many approaches to solve this problem, such as directed-search binary-splitting (DSBS), mean-distance-ordered partial codebook search (MPS), double test of principal components (DTPC), enhance LBG (ELBG), centroid neural network adaptive resonance theory (CNN-ART), fast-searching algorithm using projection and inequality (FAUPI), GA-based algorithm, evolution-based tabu search approach (ETSA), PNM, codebook generation algorithm using codeword displacement (CGAUCD) and so on. The discussed schemes

TABLE 1. An Example of GLA Algorithm

i	X	Nearest Codeword	Euclidean Distance C_1	Euclidean Distance C_2	Euclidean Distance C_3
0	X_1	C_3	248.41	225.942	216.959
	X_2	C_2	270.7	99.6444	126.831
	X_3	C_1	63.93	351.763	255.421
	X_4	C_3	226.17	146.952	86.9368
	X_5	C_1	195.7	243.563	263.989
1	X_1	C_3	211.99	197.74	104.896
	X_2	C_2	268.35	0	103.384
	X_3	C_1	107.4	317.134	246.25
	X_4	C_2	244.72	65.437	104.896
	X_5	C_1	107.4	257.919	212.728
2	X_1	C_3	211.99	134.142	0
	X_2	C_2	268.35	68.9227	197.74
	X_3	C_1	107.4	267.536	260.083
	X_4	C_2	244.72	79.9229	209.793
	X_5	C_1	107.4	223.534	212.859
3	X_1	C_3			
	X_2	C_2			
	X_3	C_1			
	X_4	C_2			
	X_5	C_1			

CB_i	Codeword				
CB_0	C_1	32	177	143	210
	C_2	196	16	46	24
	C_3	180	130	212	101
CB_1	C_1	59.5	136	63.5	240
	C_2	212	76	123	36
	C_3	203	150	88	98.5
CB_2	C_1	59.5	136	63.5	240
	C_2	206	125.3	99.67	77.67
	C_3	241	192	21	156
CB_3	C_1	59.5	136	63.5	240
	C_2	206	125.3	99.67	77.67
	C_3	241	192	21	156

are shown in Fig. 3. Some of them are designed to improve the distortion of the reconstructed image and the others are designed to reduce the computation time of the training procedure.

Year	Proposed Scheme
1986	LBG
1993	MPS, DSBS
1997	DTPC
2001	ELBG
2003	CNN-ART
2004	FAUPI
2005	GA-based algorithm
2007	ETSA, PNM
2008	CGAUCD

FIGURE 3. VQ Codebook Generation Schemes

3. **Mean-distance-ordered Partial Codebook Search (MPS).** Ra and Kim proposed a fast mean-distance-ordered partial codebook search algorithm in 1993 [10]. They used squared mean distance (SMD) to filter false candidate codewords. The definition of SMD is $d_{SMD}(X, C) = (\sum_{t=1}^{k} x_t - \sum_{t=1}^{k} c_t)^2$. In their scheme, if the codeword C whose $|\sum_{t=1}^{k} x_t - \sum_{t=1}^{k} c_t|$ is larger than $\sqrt{kd(X, C_{min})}$, it will not be the nearest neighbor to X. In other words, its SMD is larger than the Euclidean distance, $d_{SMD}(X, C) \geq d(X, C_{min})$ is the Euclidean distance between X and the tentative matching codeword which has minimum $|\sum_{t=1}^{k} x_t - \sum_{t=1}^{k} c_t|$ in the current stage.

Let us consider Fig. 4 as an example codebook to illustrate the MPS scheme. First, the scheme computes the sum of all dimensions for each codeword and sorts the sum values in increasing order. For a training vector $X = (109, 52, 19, 247)$, they calculate the mean distances $|\sum_{t=1}^{k} x_t - \sum_{t=1}^{k} c_t|$ and find the tentative matching codeword. In this case, the fifth codeword C_5 is the tentative matching codeword with the minimum mean distance. Next, the scheme computes the Euclidean distance $d(X, C_{min}) = \sqrt{(109-111)^2 + (52-137)^2 + \ldots + (247-51)^2} \approx 238.497$ between X and C_5. We can see that the codewords C_1, C_2, and C_8 for which $|\sum_{t=1}^{k} x_t - \sum_{t=1}^{k} c_t|$ is larger than $\sqrt{kd(X, C_{min})} = \sqrt{4 \times 238.497} \approx 30.887$ can be eliminated. Next, the scheme performs full searching algorithm for C_3, C_4, C_5, C_6 and C_7 to calculate the distances and update $d(X, C_{min})$.

	c_1	c_2	c_3	c_4	sum	$\left\lvert\sum_{t=1}^{k} x_t - \sum_{t=1}^{k} c_t\right\rvert$
C_1	103	2	134	94	333	94
C_2	87	11	16	224	338	89
C_3	11	236	2	166	415	12
C_4	136	26	161	99	422	5
C_5	111	137	125	51	424	3
C_6	173	1	88	178	440	13
C_7	98	187	132	26	443	16
C_8	82	53	247	98	480	53

FIGURE 4. An Example Codebook for MPS Algorithm

4. **Enhance LBG (ELBG).** Patane and Russo proposed a clustering algorithm called enhanced LBG (ELBG) in 2001. They used the concept of utility of a codeword to overcome the local optimal problem of LBG. The utility is defined as follow:

$$U_j = \frac{D_j}{D_{mean}}, \tag{2}$$

in which D_j is the total distortion of the j-th cluster and D_{mean} is the mean value of all the clusters that is computed by

$$U_{mean} = \frac{1}{N} \sum_{j=1}^{N} D_j. \tag{3}$$

They divide the codewords in the codebook into two cells, HC and LC. The codeword whose U_j is higher than 1 is classified into HC cell. Otherwise, the codeword is classified into LC cell. Next, the codeword C_L with the smallest distortion in LC cell is heuristically shifted to a nearby codeword C_H in HC cell.

Assume that the training vectors assigned to C_H are bounded in a hyper box $HT = [x_{1m}, x_{1M}] \times [x_{2m}, x_{2M}] \times \ldots \times [x_{km}, x_{kM}]$ in which x_{tm} and x_{tM} are the minimum and maximum values of t-th dimension of all training vectors belonging to HC. C_H is the centroid vector of HT. They recalculate the codewords C_L and C_H by
$C'_L = [x_{1m} + \frac{1}{4}(x_{1M} - x_{1m}), x_{2m} + \frac{1}{4}(x_{2M} - x_{2m}), \ldots, x_{km} + \frac{1}{4}(x_{kM} - x_{km})]$ and
$C'_H = [x_{1m} - \frac{1}{4}(x_{1M} - x_{1m}), x_{2m} - \frac{1}{4}(x_{2M} - x_{2m}), \ldots, x_{km} - \frac{1}{4}(x_{kM} - x_{km})]$, respectively, for shifting C_L to C_H. After codeword shifting, the scheme performs the traditional LBG to re-cluster all the training vectors by C'_L and C'_H.

An example is shown in Table 2. First, they apply the LBG algorithm to train the training vectors as shown in Fig. 2. X_3 and X_5 are classified into C_1, X_1 and X_4 are grouped into C_3, and X_2 belongs to C_2. The total distortions D_j of C_1, C_2, and C_3 are 259.63, 99.64, and 303.9, respectively. D_{mean} is 221.06 since $D_{mean} = \frac{1}{3}(259.63 + 99.64 + 303.9) \approx 221.06$. The utilities U_j of each cluster are 1.17, 0.45 and 1.37, respectively. Since U_2 is less than 1, C_2 is classified into LC cell. Meanwhile, C_1 and C_3 are classified into HC cell.

C_2 is the codeword with the smallest distortion in LC cell, hence it is shifted to the codeword C_3 with the largest distortion in HC cell. The hyper boxes of C_2 and C_3 are shown in "Hyper Box" column of Table 2. The scheme recalculates the codewords C_2 and C_3 by
$C'_2 = [212 + \frac{1}{4}(212 - 212), 76 + \frac{1}{4}(76 - 76), \ldots, 212 + \frac{1}{4}(36 - 36)]$ and
$C'_3 = [165 - \frac{1}{4}(241 - 165), 108 - \frac{1}{4}(192 - 108), \ldots, 41 + \frac{1}{4}(156 - 41)]$, respectively,
for shifting C_2 to C_3. After codeword shifting, the scheme performs the traditional LBG to re-cluster all the training vectors by C'_2 and C'_3. The initial codebook is SCB_0. After shifting and retraining procedures, we can get the final C'_2 and C'_3 in SCB_1. The scheme resets the codewords to their original positions. The process is repeated until the codebook is converged.

5. **Principal Component Analysis (PCA).** Principal component analysis (PCA) is a statistical method that transforms a number of possibly correlated variables into a smaller number of uncorrelated variables that is therefore called principal components. PCA has been widely and successfully applied in many applications including pattern recognition, time series prediction, image processing, exploratory data analysis, data compression and so on.

Since it is a well-established technique for dimensionality reduction and multivariate analysis, PCA has been used in VQ. For example, Huang and Harris proposed a directed-search binary-splitting (DSBS) method in 1993 [4]. In their scheme, PCA is used to select the initial codebook to reduce the dimension of the training vectors. After that, Han et al. also used PCA to select seed [3].

In 1997, Chang et al. proposed an improved codebook search algorithm which is called double test of principal components (DTPC), by using PCA [1]. Let us use Fig. 4 as an example to demonstrate their algorithm. First of all, they generate the covariance matrix for the codewords. The covariance matrix is shown in Fig. 5. Next, the scheme calculates eigenvalues and its corresponding eigenvectors. Let $\lambda_1, \lambda_2, \ldots, \lambda_k$ be the eigenvalues of the covariance matrix in which $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_k$. $EV_1, EV_2, \ldots, EV_k$ denote the corresponding eigenvectors. The eigvalues and its corresponding eigenvectors of Fig. 4 are shown in Fig. 6. The first eigenvector EV_1 represents most of preserved information, since $50\% \approx (\frac{9439.9}{9439.9+7693.7+1226.5+377.3})$. Hence, the authors take EV_1 = (-0.3391, 0.8359, -0.4305, -0.0311) as a project vector. The codewords in the codebook are projected to

TABLE 2. An Example of ELBG Algorithm

i	X	Total Distortion C_1	C_2	C_3	LC	HC
	X_1			216.96	C_2	C_1 C_3
	X_2		99.64			
	X_3	63.93				
	X_4			86.94		
	X_5	195.7				
	D_j	259.63	99.64	303.9		
	U_j	1.17	0.45	1.37		
0	X_1	197.74				
	X_2	0.00				
	X_3		315.33			
	X_4	65.44				
	X_5		242.27			
	X_1	134.14				
	X_2	68.92				
	X_3		107.40			
	X_4	79.92				
	X_5		107.40			

Hyper Box

$HT\_LC\_C_2$

m	212	76	123	36
M	212	76	123	36

$HT\_HC\_C_3$

m	165	108	21	41
M	241	192	155	156

CB_i

CB_0

C_1	32	177	143	210
C_2	196	16	46	24
C_3	180	130	212	101

SCB_0

C_2'	212	76	123	36
C_3'	146	87	-12	12

SCB_1

C_2'	206	125	99	77
C_3'	59	136	63	240

EV_1 for computing the projected values. The scheme shorts the codewords according to the projected values. The shorted results are shown in Table 3.

Next, the scheme projects each training vector into EV_1 for computing the projected value. For example, the projected value of the training vector $X = (241, 192, 21, 156)$ is 64.88, since $156 = (241 \times -0.3391 + 192 \times 0.8359 + \ldots + 156 \times -0.0311)$.

The closest codeword is C_7' with projected value 65.4469. C_6' and C_8' are the neighboring codewords of C_7', so they have high possibility to be the candidate codewords too. Therefore, the scheme computes the Euclidean distances between X_1 and C_6', C_7', C_8', respectively. The distances are 204.37, 222.92 and 235.15. The distance between X_1 and C_6' is the smallest one. Hence, the encoded result of X_1 is the original index of C_6' and that is 5.

1896.609	-2504.33	1098.484	-277.25
-2504.33	7467.984	-1722.58	-1847.25
1098.484	-1722.58	5445.109	-2969.63
-277.25	-1847.25	-2969.63	3927.75

FIGURE 5. Covariance Matrix of the Codewords

6. **Neural Network.** Lin and Yu [8] proposed a centroid neural network adaptive resonance theory (CNN-ART) algorithm to generate a codebook in 2003. The CNN-ART network contains of an input layer and an output layer. The total number of neurons in the input layer is the same as the total number of dimension of the training vector. Each neuron in the input layer has connections to the neurons in the output layer. The total number of neurons in the output layer is that of the codewords in the codebook.

EV_1	(-0.3391,0.8359,-0.4305,-0.0311)	λ_1	9439.90
EV_2	(0.0184,0.3218,0.6594,-0.6792)	λ_2	7693.70
EV_3	(-0.7478,-0.0328,0.4933,0.4431)	λ_3	1226.50
EV_4	(-0.5705,-0.4435,-0.3695,-0.5843)	λ_4	377.30

FIGURE 6. The Eigvalues and Its Corresponding Eigenvectors of Fig. 2

TABLE 3. The Sorted Projected Values and Its Codeword

Old Index	New Index	Projected values	Codeword			
C_6	C'_1	-101.248	173	1	88	178
C_4	C'_2	-96.7736	136	26	161	99
C_1	C'_3	-93.8659	103	2	134	94
C_8	C'_4	-92.8848	82	53	247	98
C_2	C'_5	-34.1612	87	11	16	224
C_5	C'_6	21.4796	111	137	125	51
C_7	C'_7	65.4469	98	187	132	26
C_3	C'_8	187.5187	11	236	2	166

In their scheme, the first input training vector X_1 is selected as the centroid of the first neuron, and then the next input vector is compared to the neuron. If the Euclidean distance between the neuron and the next vector is higher than a predefined threshold, the input vector forms the centroid of a new neuron. The procedure is repeated for all the training vectors. The algorithm is shown in the following.

Step 1: The initial weight of the network is equal to the values of the first training vector. The number of neuron in the network is 1.

Step 2: Input the training vector to the network and compute Euclidean distance between the vector and existing weights.

Step 3: If the smallest Euclidean distance is greater than the predefined threshold and the number of neuron is less than the total number of codewords, then generate a new neuron. The weight of the neuron is equal to the values of the input vector. Go to Step 2.

Step 4: Reward the weights of the winner neuron and punish the loser neurons.

Step 5: Go to Step 2 until the network is converged. The weights in the neurons are considered as the codewords.

In 2007, Han et al. proposed a hybrid scheme called PNM based on Lin and Yu's CNN-ART algorithm, PCA and mean shift (MS) operation to improve traditional LBG approach [3]. The CNN-ART algorithm is used to generate the initial cluster results, the MS operation is perform on each cluster to refine the codeword and PCA technique is applied to resetting the seed of the codebook for avoiding the local optimal problem. They calculate the sample distribution by using PCA and replace the codeword with low utility by a new seed.

In Han et al.'s scheme, they compute the covariance matrix for the training vectors first. Next, the major k eigenvectors corresponding with the largest k eigenvalues are selected. The training vectors are projected to the k eigenvectors to obtain the temporary points with scalar values. They cluster the temporary points by k-mean clustering method and compute the center for each cluster to get the candidate codeword.

The PNM algorithm is shown as follow:

Step 1: Apply CNN-ART algorithm to generate the cluster result.
Step 2: Compute the mean shift vector for each cluster and shift it to the next position.
Step 3: Resetting the seed by PCA scheme.

- Split or merge cluster based on the codeword utility.
- For each split cluster
  - **a:** Compute the eigenvectors and eigenvalues.
  - **b:** Project the vectors in the cluster to the first three eigenvectors.
  - **c:** Cluster the vectors by 1D projected value for each eigenvector.
  - **d:** Select some candidate centers with the smallest variances to be the new seeds.
- Delete codewords with low utility.

Step 4: Go to Step 1 until the codebook is converged.

The comparison results among LBG, CNN-ART, ELBG and PNM are shown in Table 4.

Table 4. The Comparison Results among LBG, CNN-ART, ELBG and PNM (unit: dB)

N	LBG	CNN-ART	ELBG	PNM
32	25.77	26.47	27.85	28.86
64	27.65	28.05	28.93	30.54
128	28.74	28.94	29.80	31.66
256	30.19	30.69	32.10	32.76

7. **Genetic-based Approach.** Genetic algorithm (GA) is another approach to avoid local optimal problem from finding the globally optimal solution [2, 12]. For example, Zhang et al. used genetic algorithm to design codebook and applied it to speaker identification [12]. In their scheme, the codewords are regarded as the chromosomes. The set of the codewords is the population. The fitness value of each chromosome is the total number of the training vectors in the chromosome.

First, they randomly generate a population of chromosomes. Next, the scheme performs crossover and mutation operations to evolve the offspring population. Then, the training vectors are assigned to the nearest chromosome in the population. The scheme computes the fitness values for the chromosomes and sorts each chromosome in terms of the fitness values. The best N chromosomes are selected to be the new population in the next generation. The diagram of GA is shown in Fig. 7.

Pan and Cheng applied tabu search to develop an evolution-based codebook generation approach in 2007 which is called evolution-based tabu search approach (ETSA) [9]. Their scheme is similar to the GA-based algorithm. In the training procedure, a parent population P with N codewords is randomly generated firstly. Then, they use sexual and asexual reproduction operators to generate the sexual offspring population P_s and the asexual offspring P_a, respectively. The procedures are similar to the crossover and mutation operators in GA. Next, two offspring populations P_s and P_a are combined in order to form a new population $P_o = \{P_s \cup P_a\}$. Hence, the total number of codewords in P_o is $2 \times N$.

The scheme computes the fitness value which combines with the distortion and tabu-distance for each codeword in P_o. The best N offspring from P_o are selected as the next parent population to evolve. The process is terminated at a predefined number of generations. The flowchart of their scheme is shown in Fig. 8.

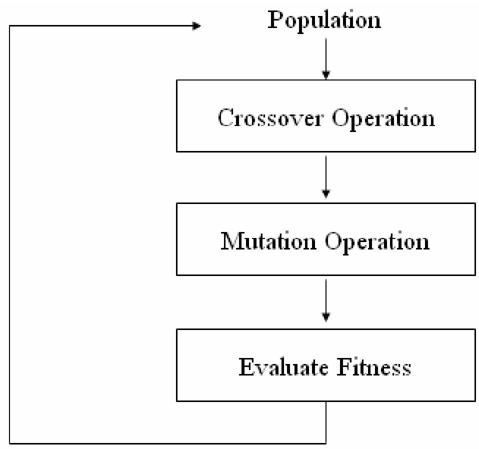

FIGURE 7. The Diagram of GA

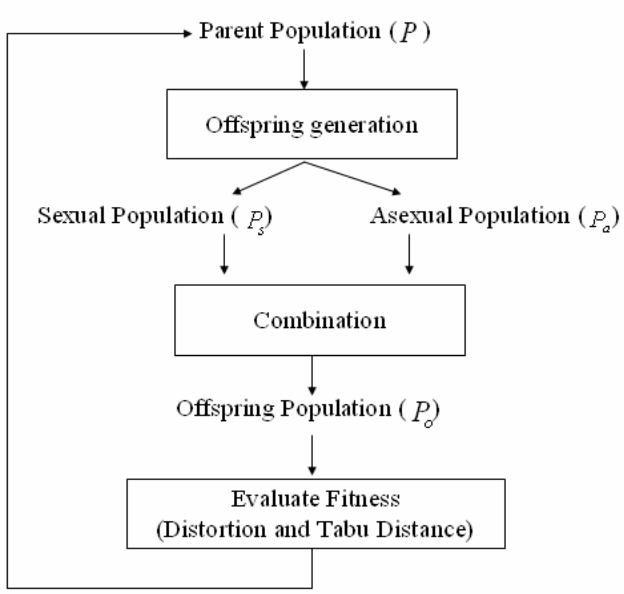

FIGURE 8. The Flowchart of ETSA Scheme

8. **Codeword Displacement.** Lai and Liaw proposed a fast-searching algorithm using projection and inequality (FAUPI) in 2004 [5]. They used two inequalities to reduce the distortion computation and reject the unlikely codewords. The first equation is used to terminate the searching process and the second equation is to delete the impossible codewords.

After that, Lai et al. proposed a codebook generation algorithm using codeword displacement which is called codebook generation algorithm using codeword displacement (CGAUCD) [7]. In their scheme, the codewords are grouped into two clusters which are static cluster and active cluster. The codeword categorized in the static cluster means the value of the codeword is the same as that of the codeword in the last iteration. Otherwise, the codeword is categorized into the active cluster. If a training vector is in a static cluster, then they only need to calculate the distances between the vector and the codewords in the active cluster to find the nearest codeword. In addition, they also applied other fast search algorithms, such as MPS, FAUPI and so on to reduce the computation time. The algorithm of their scheme is shown as below.

Step 1: Randomly generate an initial codebook CB_0 and preprocess all training vectors for fast search.

Step 2: $i = 1$.

Step 3: Calculate the closest distance r'_1 and the second closest distance r'_2 for each training vector and generate a new codebook CB_i.

Step 4: Group the codewords of CB_i into two clusters which are static cluster and active cluster.

Step 5: Perform the following process for each training vector.

- If the training vector is in a static cluster, then search the nearest codeword from the active cluster.
- If the training vector is in an active cluster and the current distance r between the vector and the center codeword is less than r'_2, then search the nearest codeword from the active cluster.
- Otherwise, search all codewords from CB_i to find the nearest codeword.

Step 6: $i = i + 1$.

Step 7: Go to Step 3 until the codebook is converged.

Let us use the same example to describe Lai et al.'s algorithm. Fig. 2 shows the training vectors. The initial generated codebook CB_0 is shown in Table 5. In the first step, the scheme performs full searching to compute the distance for each training vector. The column "Distance" records whether the training vector needs to compute the distance with the center codeword C_j or not. The column recorded 1 means that the scheme needs to compute the distance between the vector and the center codeword.

In the next step, the scheme calculates the closest distance r'_1 and the second closest distance r'_2 for each training vector. For example, the distances among the first training vector X_1 and three codewords C_1, C_2 and C_3 are 248.40, 225.94, and 216.96, respectively. Hence, the closest distance r'_1 is 216.96 and the second closest distance r'_2 is 225.94. Next, the scheme partitions the training vectors into three cells and computes the centroid of each cell to generate the new codebook CB_1. For example, because both of X_1 and X_4 have the same nearest codeword C_3, they are partitioned into the same cell. Hence, the centroid of the two vectors is (203, 150, 88, 98.5) that is the third codeword of the new generated codebook CB_1. Because the values of the codewords in CB_1 are not the same as that of the codewords in CB_0, no codeword is categorized into the static cluster. All of the codewords are in the active cluster.

In the fourth step, the scheme compares each training vector with the codewords in the proper cluster for searching the nearest codeword. For example, in the last stage X_1 is close to C_3 which is in an active cluster. Therefore, the scheme calculates the distance r between X_1 and C_3. Since $r = 104.90 < r'_2 = 225.94$, the scheme searches the nearest codeword from the active cluster. In this example, all codewords are in the active cluster. Therefore, the scheme still needs to compute the distances between X_1 and all the codewords.

Next, the scheme goes back to the step 3 to calculate r'_1 and r'_2, and generate the new codebook CB_2. Since the value of C_1 in CB_2 is the same as that of C_1 in CB_1, C_1 is categorized into static cluster.

In this stage, the schemes does not need to refer to the codewords in a static cluster except the training vector is in an active cluster and $r \geq r'_2$. For example, for $i = 2$, X_1 is in an active cluster and $r = 0 < r'_2 = 197.74$, the scheme only needs to search the nearest codeword from the active cluster. C_1 is in the static cluster, so the scheme skip the codeword. The procedure is repeated until the codebook is converged. The final codebook is CB_3.

In CGAUCD scheme, the initial codebook is randomly generated. In order to speed up the clustering process, Lai et al. applied kd-tree to generate the cluster center as the initial codebook [6]. In addition, they also used the kd-tree to determine the nearest neighbors. We use the example as shown in Fig. 2 to illustrate their improvement algorithm.

First, they find the median value of the first dimension x_1 to split the five training vectors. In this example, the median value of x_1 is 165. Because the values of x_1 are lesser than the median value, X_3 and X_5 are split into the left side. On the contrary, X_1, X_2 and X_4 are split into the right side. The tree structure is shown in Fig. 9. Next, they use the median values of the second dimension x_2 in the different sides to split the vectors. The splitting procedure repeats until every vector becomes a leaf node. The final kd-tree is shown in Fig. 10.

The total number of codewords in the initial codebook is three, so they choose three significant nodes from the tree structure to be the codewords. The selected nodes are X_3, X_5 and the centroid ofX_1, X_2 and X_4. The initial codebook CB_0 is shown in Table 6. After the initial codebook is generated, they apply CGAUCD scheme to train the final codebook. The training process is shown in Table 6. Because CB_1 is the same as CB_0, all codewords are moved into the static cluster.

The codebook is converged in the first round by the improvement algorithm. That means the improvement algorithm is faster than CGAUCD scheme. However, the algorithm still has local optimal problem.

TABLE 5. An Example of Lai et al.'s Algorithm

i	X	r	r_1'	r_2'	Nearest Codeword	Distance C_1	Distance C_2	Distance C_3	static cluster	active cluster	CB_i
0	X_1		216.96	225.94	C_3	1	1	1			CB_0
	X_2		99.64	126.83	C_2	1	1	1			C_1: 32, 177, 143, 210
	X_3		63.93	255.42	C_1	1	1	1			C_2: 196, 16, 46, 24
	X_4		86.94	164.95	C_3	1	1	1			C_3: 180, 130, 212, 101
	X_5		195.70	243.56	C_1	1	1	1			
1	X_1	104.90	104.90	197.74	C_3	1	1	1		C_1	CB_1
	X_2	0.00	0.00	103.38	C_2	1	1	1		C_2	C_1: 59.5, 136, 63.5, 240
	X_3	107.40	107.40	246.25	C_1	1	1	1		C_3	C_2: 212, 76, 123, 36
	X_4	104.90	65.44	104.90	C_2	1	1	1			C_3: 203, 150, 88, 98.5
	X_5	107.40	107.40	212.73	C_1	1	1	1			
2	X_1	0.00	0.00	134.14	C_3		1	1	C_1	C_2	CB_2
	X_2	68.92	68.92	197.74	C_2		1	1		C_3	C_1: 59.5, 136, 63.5, 240
	X_3	107.40	107.40	260.08	C_1		1	1			C_2: 206, 125.3, 99.67, 77.67
	X_4	79.92	79.92	209.79	C_2		1	1			C_3: 241, 192, 21, 156
	X_5	107.40	107.40	212.86	C_1		1	1			
3	X_1	0.00	0.00	134.14	C_3				C_1		CB_3
	X_2	68.92	68.92	197.74	C_2				C_2		C_1: 59.5, 136, 63.5, 240
	X_3	107.40	260.08	267.54	C_1				C_3		C_2: 206, 125.3, 99.67, 77.67
	X_4	79.93	79.93	209.79	C_2						C_3: 241, 192, 21, 156
	X_5	107.40	212.86	225.53	C_1						

9. **Conclusions.** In this paper, we present a snapshot of the recent developed VQ codebook generation schemes. The discussed schemes include MPS, DSBS, DTPC, ELBG, CNN-ART, FAUPI, GA-based algorithms, ETSA, PNM and CGAUCD.

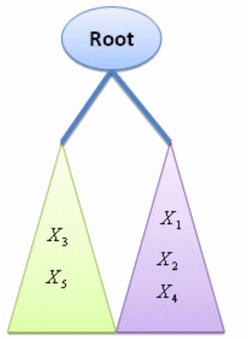

FIGURE 9. The Tree Structure of kd-tree in the First Step

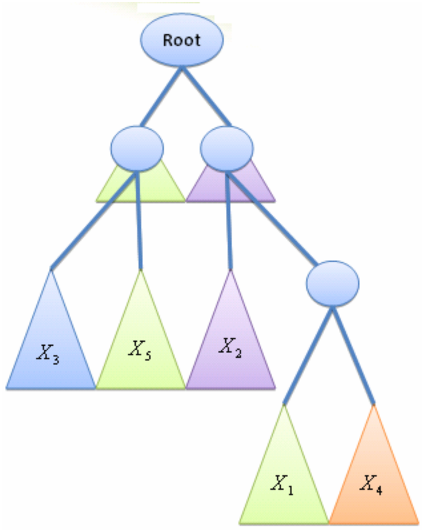

FIGURE 10. The Final Tree Structure of kd-tree

TABLE 6. An Example of Lai et al.'s Improvement Algorithm

<table>
<tr><th rowspan="2">i</th><th rowspan="2">X</th><th rowspan="2">r</th><th rowspan="2">r_1'</th><th rowspan="2">r_2'</th><th>Nearest</th><th colspan="3">Distance</th><th>static</th><th>active</th><th rowspan="2">CB_i</th></tr>
<tr><th>Codeword</th><th>C_1</th><th>C_2</th><th>C_3</th><th>cluster</th><th>cluster</th></tr>
<tr><td rowspan="5">0</td><td>X_1</td><td></td><td>134.14</td><td>212.86</td><td>C_3</td><td>1</td><td>1</td><td>1</td><td rowspan="5"></td><td rowspan="5">C_1 C_2 C_3</td><td rowspan="5">CB_0<table><tr><td>C_1</td><td>10</td><td>220</td><td>108</td><td>233</td></tr><tr><td>C_2</td><td>109</td><td>52</td><td>19</td><td>247</td></tr><tr><td>C_3</td><td>206</td><td>125.33</td><td>99.67</td><td>77.67</td></tr></table></td></tr>
<tr><td>X_2</td><td></td><td>68.92</td><td>257.92</td><td>C_3</td><td>1</td><td>1</td><td>1</td></tr>
<tr><td>X_3</td><td></td><td>0</td><td>214.81</td><td>C_1</td><td>1</td><td>1</td><td>1</td></tr>
<tr><td>X_4</td><td></td><td>79.92</td><td>259.24</td><td>C_3</td><td>1</td><td>1</td><td>1</td></tr>
<tr><td>X_5</td><td></td><td>0</td><td>214.81</td><td>C_2</td><td>1</td><td>1</td><td>1</td></tr>
<tr><td rowspan="5">1</td><td>X_1</td><td>134.14</td><td></td><td></td><td>C_3</td><td></td><td></td><td></td><td rowspan="5">C_1 C_2 C_3</td><td rowspan="5"></td><td rowspan="5">CB_1<table><tr><td>C_1</td><td>10</td><td>220</td><td>108</td><td>233</td></tr><tr><td>C_2</td><td>109</td><td>52</td><td>19</td><td>247</td></tr><tr><td>C_3</td><td>206</td><td>125.33</td><td>99.67</td><td>77.67</td></tr></table></td></tr>
<tr><td>X_2</td><td>68.92</td><td></td><td></td><td>C_3</td><td></td><td></td><td></td></tr>
<tr><td>X_3</td><td>0</td><td></td><td></td><td>C_1</td><td></td><td></td><td></td></tr>
<tr><td>X_4</td><td>79.92</td><td></td><td></td><td>C_3</td><td></td><td></td><td></td></tr>
<tr><td>X_5</td><td>0</td><td></td><td></td><td>C_2</td><td></td><td></td><td></td></tr>
</table>

MPS, DSBS, DTPC, FAUPI and CGAUCD are designed to reduce the computation time of LBG. Most of them can indeed speed up the training process. However, the qualities of the reconstructed images of these schemes are worse than that of LBG. In addition, some of them even have the block effects.

ELBG, CNN-ART, ETSA, PNM and genetic-based algorithms are designed to overcome the local optimal problem and prevent the premature convergence. However, most of them need long runtime because candidate solutions must be fine tuned by LBG.

Acknowledgment. This work is partially supported by NSC 97-2221-E-324-008. The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation.

REFERENCES

[1] C. C. Chang, D. C. Lin, and T. S. Chen, An Improved VQ Codebook Search Algorithm Using Principal Component Analysis, *Journal of Visual Communication and Image Representation*, vol. 8, no. 1, pp. 27–37, 1997.

[2] P. Franti, J. Kivijarvi, T. Kaukoranta and O. Nevalainen, Genetic Algorithms for Codebook Generation in Vector Quantization, *Proc. of the 3rd Nordic Workshop on Genetic Algorithms (3NWGA)*, Helsinki, Finland, pp. 207–222, 1997.

[3] C. C. Han, Y. N. Chen, C. C. Lo, and C. T. Wang, A Novel Approach for Vector Quantization Using a Neural Network, Mean Shift, and Principal Component Analysis-based Seed Re-initialization, *Signal Processing*, vol. 87, pp. 799–810, 2007.

[4] C. M. Huang and R. W. Harris, A Comparison of Several Vector Quantization Codebook Generation Approaches, *IEEE Trans. Image Processing*, vol. 2, no. 1, pp. 108-112, 1993.

[5] J. Z. C. Lai and Y. Liaw, Fast-searching Algorithm for Vector Quantization Using Projection and Triangular Inequality, *IEEE Trans. Image Processing*, vol. 13, no. 12, pp. 1554–1558, 2004.

[6] J. Z. C. Lai and Y. C. Liaw, Improvement of the K-means Clustering Filtering Algorithm, *Pattern Recognition*, vol. 41, pp. 3667-3681, 2008.

[7] J. Z. C. Lai, Y. C. Liaw, and J. Liu, A Fast VQ Codebook Generation Algorithm Using Codeword Displacement, *Pattern Recognition*, vol. 41, pp. 315–319, 2008.

[8] T. C. Lin and P. T. Yu, Centroid Neural Network Adaptive Resonance Theory for Vector Quantization, *Signal Processing*, vol. 83, pp. 649–654, 2003.

[9] S. M. Pan and K. S. Cheng, An Evolution-based Tabu Search Approach to Codebook Design, *Pattern Recognition*, vol. 40, pp. 476–491, 2007.

[10] S. W. Ra and J. K. Kim, A Fast Mean-Distance-Ordered Partial Codebook Search Algorithm for Image Vector Quantization, *IEEE Trans. Circuits and Systems - II: Analog and Digital Signal Processing*, vol. 40, no. 9, pp. 576–579, 1993.

[11] C. W. Tsai, C. Y. Lee, M. C. Chiang, and C. S. Yang, A Fast VQ Codebook Generation Algorithm via Pattern Reduction, *Pattern Recognition Letters*, vol. 30, pp. 653–660, 2009.

[12] L. Zhang, B. Zheng and Z. Yang, Codebook Design using Genetic Algorithm and Its Application to Speaker Identification, *Electronics Letters*, vol. 41, no. 10, pp. 619–620, 2005.