Joint Conference of 1996 International Computer Symposium
December 19~21, Kaohsiung, Taiwan, R.O.C.

A Fast Algorithm for Designing Better Codebooks in Image Vector Quantization

Mon-Ching Huang and Chang-Biau Yang
Department of Applied Mathematics
National Sun Yat-sen University
Kaohsiung, Taiwan 80424
cbyang@math.nsysu.edu.tw

Abstract

Vector quantization (VQ) is an effective method of data compression. Its decoding process is very simple and a high compression ratio can be obtained. So, VQ has been extensively used in image and speech compression. However, VQ has some difficulties such as the efficiency of codebook design process and the degradation of edges. In the codebook design and encoding phases, searching for the closest codeword is a highly computational process, especially for high dimensional vectors. So, the Linde-Buzo-Gray (LBG) algorithm, the earliest and the most famous algorithm for codebook generation, is a time consuming algorithm. In this paper, our efforts are to design an algorithm to generate a better codebook and to reduce the computation time compared with the previous algorithms in codebook generation.

Key words:. data compression, vector quantization, codebook generation, distortion measure.

1 Introduction

In the recent years, image compression is a major part of data compression because it is highly demanded in many applications such as video conferencing, television transmission, image database and archiving medical images. In the image compression, the insignificant information can be removed from images. Because the images are tested by the human visual system, the quality of images depend on the perceptual entries such as the contours of regions and edges. So, we can reduce much redundant and insignificant information to get a high compression ratio. The quantization method is an effective tool for image compression to achieve a high compression ratio. Several scalar quantization techniques have been developed, examples include differential pulse code modulation, transform coding and hybrid coding [10,13]. However, Shannon's rate-distortion theory states that using vectors instead of scalars always get a better performance [12]. Therefore, vector quantization (VQ) techniques are developed for image coding applications [2, 10, 11].

The Linde-Buzo-Gray (LBG) algorithm is one of the most famous algorithms for codebook generation [8]. It is an iterative method to modify the codebook to a better one, starting with an initial codebook. It usually requires very much time. The maximum descent (MD) algorithm [4] is another algorithm for generating codebooks. Initially, it treats the training set as a global cluster, and then splits it repeatedly until the number of clusters reaches the size of the codebook. Again, some steps are wasted in the MD algorithm. In this paper, we first propose a splitting algorithm to split a cluster into two clusters. This algorithm is based upon the simple longest distance partition heuristics. Then, using this splitting algorithm, we propose our codebook generation algorithm, called the longest distance first algorithm. The structure of our algorithm is similar to the MD algorithm, which splits the training set from one cluster into many clusters. By our experiments, the quality of codebook generated by our algorithm is superior to the LBG algorithm and the MD algorithm. Here, the distortion measure used to measure the quality of decoding images is the square error (SE). Besides, our algorithm is also faster than the two previous algorithms.

The rest of this paper is organized as follows. In section 2, we will introduce the vector quantization. In section 3, we will briefly review the previous algorithms for codebook generation. In section 4, we will propose our algorithm for generating good codebooks and present the philosophy of our algorithm. In section 5, we will give the experiment results and compare the performance of our algorithm with the previous algorithms. And finally, a conclusion will be given in section 6.

2 Definitions and Notations

In the image vector quantization [13], the first step is to decompose the input images into small blocks. We call these small blocks as *vectors*. Then, we choose the vectors of some representative images as the training set from which the codebook is generated. Moreover, each vector selects a best matching codeword from the codebook. The data compression is achieved by storing the codebook and the index of the best matching codeword of each vector. When we want to restore the images, we use the index to find the codeword in the codebook to reconstruct the images. The formal definition of VQ is as follows.

Definition 1 *An N-level k-dimensional vector quantizer is a mapping Q of k-dimensional Euclidean space R^k into a finite subset V of R^k. In other words, $Q : R^k \rightarrow V$ where $V = \{v_1, v_2, \ldots, v_N\}$ is the set of reproduction vectors(codewords). V is called the codebook.*

A vector quantizer is completely described by the codebook and the Voronoi partition, $C = \{C_1, C_2, \ldots, C_N\}$, where $C_i = \{x|Q(x) = v_i\}$ is the set of the input vectors whose codeword is v_i.

Because we use the codewords to reconstruct the image, there exists distortion between the original image and the reconstructed image. The distortion measure function $d(x, v_i)$ is used to measure the distortion between x and v_i where x is an input vector and $v_i = Q(x)$. Many kinds of distortion measure have been proposed in the literature [8]. The square error(SE) is the most commonly used distortion measure because of its mathematical convenience. The definition of square error distortion is $d(x, v_i) = \sum_{j=0}^{k-1} |x_j - v_{ij}|^2$, where $x = (x_0, x_1, \ldots, x_{k-1})$ and $v_i = (v_{i,0}, v_{i,1}, \ldots, v_{i,k-1})$.

In this paper, we use the square error as distortion measure.

3 Previous Algorithms for Codebook Generation

In this section, we will briefly review two previous algorithms for codebook generation: the LBG algorithm, developed by the Linde, Buzo and Gray [8] and the maximum descent (MD) method proposed by Chan [4].

Linde, Buzo and Gray (LBG) algorithm [8] is an iterative, non-variational technique extended from the design of scalar quantizer proposed by Lloyd [7,9]. This algorithm is as follows: At step 1, N initial codewords are chosen. There are many methods to choose the initial codewords. The simplest initialization is choosing codewords from the training set randomly. We can also choose evenly spaced points in the vector space as the initial codewords. In LBG algorithm, a splitting technique is applied to generate the initial codebook. The splitting technique is more reliable but it spends more time than the method of random choice does. At step 2, each vector is assigned to its closest codeword, based on some distortion measure and an exhaustive search. At step 3, the vectors assigned to the same codeword form a set. We find the centroid of each of these sets as a new codeword. The *centroid* of a set is the vector which minimizes the overall distortion error of the set. These new codewords form the new codebook. The second and third steps are performed repeatedly until the overall distortion error changes by a small enough fraction in two successive iterations. The execution time of this algorithm is uncertain, because we do not know how many iterations would be executed. With the LBG algorithm, we can achieve the local minimum solutions by iterating the two steps until the overall distortion error does not change.

The maximum descent (MD) algorithm [4], proposed by Chok-Ki Chan and Chi-Kit Ma, generates the codebooks with lower overall distortion errors and with much less time computation time as compared with the LBG algorithm. At the beginning of the MD algorithm, it treats the training set as a global cluster. This global cluster is partitioned into two new clusters by a partition method. One of these two clusters is further selected to be partitioned into two new clusters according to a maximum distortion reduction criterion. Then, we have three clusters. The process continues to select one of the current clusters to split into two new clusters such that the distortion reduction is maximum. This process is terminated when the number of clusters is increased to the size of the desired codebook. And each codeword of this codebook is the centroid of one of the clusters.

Suppose that cluster C_i is partitioned into two nonempty clusters C_{ia} and C_{ib}. Let $D(C_i)$ be the total distortion errors in cluster C_i, and $D(C_i) = \sum_{x \in C_i} d(x, v_i)$, where v_i is the centroid of C_i. Then, the reduction of distortion error $\triangle D_i$ due to the partition of cluster C_i into C_{ia} and C_{ib} is $D(C_i) - [D(C_{ia}) + D(C_{ib})]$, i.e. $\triangle D_i = D(C_i) - [D(C_{ia}) + D(C_{ib})]$. If both clusters C_{ia} and C_{ib} are not empty, $\triangle D_i$ is always positive.

Suppose, now, there are M clusters. The distortion reduction functions of all clusters are computed. Cluster C_j is selected to split into two new clusters C_{ja} and C_{jb} if $\triangle D_j \geq \triangle D_i$ for $1 \leq i, j \leq M$. Then we have $M + 1$ clusters. When we continue to split the clus-

ters, we need only to calculate the $\triangle D_{ja}$ and $\triangle D_{jb}$ of the newly formed clusters C_{ja} and C_{jb}. Since $\triangle D_i$'s of the other clusters have been computed in the previous steps. So, to form N clusters, $N-1$ splitting operations and $2N-3$ partition searches (to find the value of $\triangle D$) are required.

4 An Algorithm for Designing a Better Codebook

The structure of our new algorithm is very similar to that of the maximum descent (MD) algorithm [4]. We use the longest distance first method to choose which cluster should be split instead of the maximum descent criterion in the MD algorithm and we use the longest distance partition technique to partition one cluster into two new clusters instead of the 2-level LBG partition technique [4] or the hyperplane partition technique [4].

At the beginning of our new algorithm, we view the whole training set as a global cluster. We use our new splitting technique, the longest distance partition, to partition the global cluster into two new clusters. Moreover, the centroid of each new cluster is calculated and then, for each vector in each new cluster, we calculate the distance between it and its centroid as its new distance. Then, every new cluster finds the longest distance, which is the distance between the centroid and the furthest vector in the cluster. We choose the cluster whose longest distance is the maximum in all clusters to be split into two new clusters. This is why we call our method as the *longest distance first* method. The above process continues until the number of clusters is increased to the desired one.

In the MD algorithm with 2-level LBG partition technique, the 2-level LBG algorithm is used first to calculate the reduction of distortion of each cluster. Then it chooses the cluster with maximum reduction of distortion to split into two clusters. If some clusters are not split before the algorithm terminates, then the 2-level LBG partition in these clusters are wasted. If an N-level vector quantizer is designed, $(N-2)$ 2-level LBG partitions are performed, but not really used.

The most benefit of the longest distance first method is time saving. We only find the longest distance in every cluster, not calculate the reduction of distortion. Therefore, the longest distance first method requires much less time than the maximum descent method does. Although the longest distance first method saves much time, one may think that the quality of the codebooks generated by this method is not very good. We find that, in general, the clusters with large longest distances usually have large distortions. If we can use more representative codewords in such clusters, we can usually get more reduction of distortion. Therefore, usually, our longest distance first method has good performance.

Before we describe our longest distance partition technique, let us see the Huygen's theorem [1, 2]. We need some notations as follows:

N: number of levels.
X: training set $\{x_1, x_2, \ldots, x_n\}$.
C_i: ith cluster of training vectors.
n_i: number of training vectors in C_i.
v_i: centroid (mean) of the training vectors in C_i.
W: centroid (mean) of all vectors in the training set.

And the following terms are defined:

d: the square error distortion measure function

$$d(y,z) = \sum_{i=1}^{k} |y_i - z_i|^2, \text{ where}$$

$y = (y_1, y_2, \ldots, y_k)$, $z = (z_1, z_2, \ldots, z_k)$.

E: the overall distortion error,

$$E = \sum_{i=1}^{N} \sum_{x_j \in C_i} d(x_j, v_i).$$

∇: the inter-cluster error,

$$\nabla = \sum_{i=1}^{N} n_i d(v_i, W).$$

I: the training set inertia,

$$I = \sum_{j=1}^{n} d(x_j, W).$$

Akrout had shown that Huygen's theorem is $E = I' - \nabla'$ [2], where $I' = \frac{1}{n} I$ and $\nabla' = \frac{1}{n}\nabla$. But it is wrong. We will prove that the correct Huygen's theorem is $E = I - \nabla$.

Theorem 1 $E = I - \nabla$.

Proof: According to the vector summation (See Figure 1)

$$(W - x_j) = (W - v_i) - (x_j - v_i)$$
$$\Rightarrow (W - x_j) \cdot (W - x_j) = [(W - v_i) - (x_j - v_i)] \cdot [(W - v_i) - (x_j - v_i)]$$
$$\Rightarrow |W - x_j|^2 = (W - v_i) \cdot (W - v_i) + (x_j - v_i) \cdot (x_j - v_i) - 2(W - v_i) \cdot (x_j - v_i)$$
$$= |W - v_i|^2 + |x_j - v_i|^2 - 2(W - v_i) \cdot (x_j - v_i)$$
$$\Rightarrow d(W, x_j) = d(W, v_i) + d(x_j, v_i) - 2(W - v_i) \cdot (x_j - v_i)$$

Consider the vectors in cluster $C_i = \{x_{i,1}, x_{i,2}, \ldots, x_{i,n_i}\}$, $1 \le i \le N$. Then

$$\sum_{j=1}^{n_i} d(W, x_{i,j}) = n_i d(W, v_i) + \sum_{j=1}^{n_i} d(x_{i,j}, v_i) - 2(W - v_i) \sum_{j=1}^{n_i} (x_{i,j} - v_i)$$

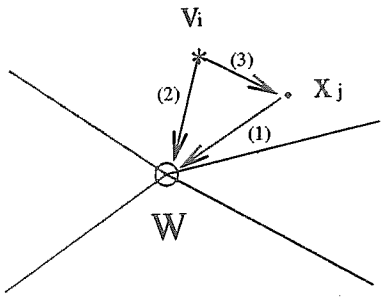

Figure 1: Illustration of the vector summation. Vector (1) = vector (2)− vector (3), where vector (1) = $(W - x_j)$, vector (2) = $(W - v_i)$ and vector (3) = $(x_j - v_i)$.

$$= n_i d(W, v_i) + \sum_{j=1}^{n_i} d(x_{i,j}, v_i),$$

because

$$\sum_{j=1}^{n_i}(x_{i,j} - v_i) = \sum_{j=1}^{n_i} x_{i,j} - n_i v_i$$
$$= \sum_{j=1}^{n_i} x_{i,j} - n_i \cdot \frac{\sum_{j=1}^{n_i} x_{i,j}}{n_i}$$
$$= 0$$

So,

$$\sum_{i=1}^{N}\sum_{j=1}^{n_i} d(W, x_{i,j}) = \sum_{i=1}^{N} n_i d(W, v_i) + \sum_{i=1}^{N}\sum_{j=1}^{n_i} d(x_{i,j}, v_i)$$

$$\Rightarrow \sum_{j=1}^{n} d(W, x_j) = \sum_{i=1}^{N} n_i d(W, v_i) + \sum_{i=1}^{N}\sum_{x_j \in C_i} d(x_j, v_i)$$

That is,

$$I = \nabla + E$$

So,

$$E = I - \nabla$$

∎

As we can see in Huygen's theorem $E = I - \nabla$, the value of I is independent of the clustering processing, because I only involves the training vectors and the centroid of the training set. As I is a constant value, we can minimize the value of E by maximizing the value of ∇. Because ∇ is equal to $\sum_{i=1}^{N} n_i d(v_i, W)$, maximizing ∇ implies that each codeword v_i must be the furthest from W. So, our longest distance partition technique is to try to find two codewords such that these two codewords are far away from the centroid of the cluster being split.

In our longest distance partition, we first find the vector which is the furthest from the centroid of the splitting cluster as a new codeword. Then, we compute the distance between each vector in the splitting cluster and the new codeword. We choose the vector with the longest distance to the new codeword as the other codeword. The cluster now is split into two clusters according to these two codewords. That is, the vectors which are closer to the same codeword form a new cluster, and the other vectors form the other new cluster. Finally, we modify the two codewords to be the centroids of the two new clusters respectively. The last step is to minimize the distortion error in these two new clusters. Our algorithm is formally described as follows.

Algorithm Longest Distance Partition (LDP)

Input: The splitting cluster $C_i = \{x_{i,1}, x_{i,2}, \ldots, x_{i,n_i}\}$.

Output: Two new clusters $C_{i,a}$ and $C_{i,b}$ and two codewords of them.

Step 1: Calculate centroid v_i of the splitting cluster C_i.
Compute $d(x_{i,j}, v_i)$, $1 \le j \le n_i$.
Find $x_{i,p}$ such that $d(x_{i,p}, v_i) = \max_{1 \le j \le n_i} d(x_{i,j}, v_i)$.

Step 2: Compute $d(x_{i,j}, x_{i,p})$, $1 \le j \le n_i$.
Find $x_{i,q}$ such that
$d(x_{i,q}, x_{i,p}) = \max_{1 \le j \le n_i} d(x_{i,j}, x_{i,p})$.

Step 3: Split cluster C_i into $C_{i,a}$ and $C_{i,b}$. That is, if $d(x_{i,j}, x_{i,p}) < d(x_{i,j}, x_{i,q})$ then $x_{i,j} \in C_{i,a}$ else $x_{i,j} \in C_{i,b}$, $1 \le j \le n_i$.

Step 4: Calculate the centroids of $C_{i,a}$ and $C_{i,b}$ as two new codewords.

If our distortion measure function d is square error, Chan proposed a fast method [4] to determine which cluster x should belong to. When we compare our new splitting technique, longest distance partition (LDP), with the 2-level LBG partition technique, our new technique requires $2n_i$ comparisons more than one iteration of the 2-level LBG partition technique. But the 2-level LBG partition technique is an iterative method, it usually executes several iterations before the reduction of distortion of two consecutive iterations changes by a small enough fraction. Thus, our LDP algorithm usually requires much less time than the 2-level LBG partition technique. Moreover, because our LDP technique is trying to find the codewords which is the furthest away from the centroid of the cluster, we usually get a good partition due to Huygen's theorem.

In the codebook generation algorithm, we propose the longest distance first method to choose which cluster should be split. We first calculate the distance between each vector and the centroid of its cluster, and

find the vector which is the furthest away from the centroid of its cluster. Then, we apply our LDP technique to split the cluster.

Algorithm Longest Distance First (LDF)

Input: The number of levels N and the training set.

Output: The codebook.

Step 1: Split the entire training set into two new clusters using our longest distance partition (LDP) technique.

Step 2: Let the two newly formed clusters be C_a and C_b. Find the longest distances in C_a and C_b respectively.

Step 3: Select the cluster with the maximum longest distance in all clusters. Split the selected cluster into two new clusters using our LDP technique.

Step 4: If the number of current clusters is N then output the centroids of all clusters as the codebook and halt; otherwise goto Step 2.

Although the performance of our LDF algorithm is superior to the MD algorithm with 2-level LBG partition technique, we can use the maximum descent strategy combined with our splitting technique, longest distance partition (LDP), to improve the quality of codebooks. According to our experiment results, the codebooks generated by MD algorithm with LDP have higher quality than those generated by MD algorithm with 2-level LBG partition technique and our LDF algorithm. However, the execution time of MD algorithm with LDP is more than that of our LDF algorithm due to the maximum descent strategy. So, if we want to have high coding quality, we would like to use the MD algorithm with our LDP splitting technique. If a fast algorithm is desired, our LDF algorithm is preferred.

5 Experiments and Performance Analysis

In this section, we will illustrate the performance of our longest distance first (LDF) algorithm in local and global codebook generation [13]. In the local codebook generation, the training set is formed by the encoding image itself. The special vectors of the encoding image, such as edge and contour, are considered during codebook generation. Therefore, codebooks generated by this local method usually have good quality. In the global codebook generation, we use some representative images to form the training set, then use this training set to generate the global codebook. For an encoding image, we only search the closest codeword in this global codebook for each vector in the encoding image. However, the coding quality in the global codebook generation is usually not very good if the encoding image are not similar to representative images.

All simulations were performed on an IBM RS6000/58H, and the simulation programs were written in C language. All input images are standard monochrome pictures of size 256×256 with 256 grey levels, and the vector (block) size is 4×4. The distortion of the encoded image is measured by the peak signal-to-noise (PSNR), which is defined as PSNR= $10 \log_{10} [\frac{255^2}{\frac{1}{L \times L} \sum_{i=1}^{L} \sum_{j=1}^{L} (y_{ij} - \hat{y}_{ij})^2}]$, where $L \times L$ = size of image, y_{ij} = pixel value of the original picture at coordinate (i, j), and $\hat{y}_{ij}$ = pixel value of the reproduced picture at coordinate (i, j) [4].

In each performance evaluation, six codebooks are designed for comparison. The first one is generated by Linde-Buzo-Gray (LBG) algorithm. And we choose evenly spaced elements in the training set to form the initial codebook of LBG. For example, $x_m, x_{m+(j+1)}, \ldots, x_{m+((N-1)j+1)}$ can be chosen as the initial codebook, where x_i is the ith training vector, m is an offset, n is the number of training vectors, N is the number of levels and $j \doteq \frac{n}{N}$ [7]. The distortion threshold ε is chosen as 0.005. And we use a partial distance elimination method to search for the closest codeword. The second one is generated by the pairwise nearest neighbor (PNN) algorithm [3, 7], which is very different from the LBG algorithm. It begins with a separate cluster for each vector in the training set and merges together two clusters at a time until the desired codebook size is achieved or the total distortion error is reduced to the predefined threshold. Its purpose is to find a codebook quickly. However, its codebook quality is not so good. Sometimes, its codebook is used as an initial codebook of another codebook generator. Thus, in our experiment, the third codebook is generated by running LBG algorithm with PNN codebook as its initial codebook. The fourth one is generated by the maximum descent (MD) algorithm with 2-level LBG partition technique. The 2-level LBG partition technique uses the standard splitting technique as described in Section 3.1 to initialize the codebook, and the distortion threshold ε is chosen as 0.005. The fifth one is generated by our longest distance first (LDF) algorithm. The last one is generated by the MD algorithm with our longest distance partition (LDP) technique.

Figure 2 shows the original image of Lena, one of the images on which we do experiments.

Table 1 illustrates the performance of the six algorithms in local codebook generation. When we use

Figure 2: Original image of Lena

the local codebook generation method as a compression strategy, we usually want to get a codebook with high quality. So, a full search with the partial distance elimination method is used after the codebooks are generated by the MD algorithm with 2-level LBG partition technique, our LDF algorithm and the MD algorithm with LDP. The codebook size of this simulation is 256.

Table 2 illustrates the performance of the four algorithms in global codebook generation. The codebook size of this simulation is 1024. The bit rate, which is the number of bits used to represent one pixel, is 5/8, where bit rate=$(\log_2 N)/k$, N is the codebook size and k is the vector size [2]. The training set was extracted from Lena, Tiffany, Camera and Kenwood.

By the above experiments, the execution time of our LDF is slightly more than that of PNN. However, the codebook quality of our LDF is much better than that of PNN. Besides, both execution time and codebook quality of our LDF is superior to those of LBG, PNN before LBG, MD with 2-level LBG. The most important reasons are that the calculation in the longest distance first method is reduced and the splitting technique based on Huygen's theorem works very well. When we combine the LDP technique and the maximum descent (MD) strategy, we can get higher codebook quality with the loss of execution time. If we want to have high codebook quality, we would like to use the maximum descent strategy with LDP technique. If to reduce the execution time is desired, we can use our LDF algorithm, including our LDP technique. On the other hand, when a global codebook is used, the qualities of the images outside the training set are almost the same among those algorithms. And, they are not so good, which may be due to improper selection of training set in our experiments.

6 Conclusion

The set clustering problem has been investigated in many areas, examples include operation research, statistic, geometry algorithm and data compression. Because of the applications of each area, the criteria of the set clustering problem are usually different. For example, the minimax location allocation problem (the P-center problem) in operational research [5] is to minimize the maximum radius of all sub-clusters, where the radius of each sub-cluster is the radius of the minimal circle which can cover all points in this sub-cluster. And the vector quantization (VQ) problem in data compression is to minimize the sum of distortion function between each vector and its corresponding codeword. The set clustering problem is usually very hard. For instance, the P-center problem is NP-hard [6]. So, many heuristic algorithms have been proposed to solve the set clustering problem with different criteria. In this paper, our efforts focus on the vector quantization problem in image data compression. And we developed a heuristic algorithm to generate better codebooks and to reduce the computation time as compared with the Linde-Buzo-Gray (LBG) algorithm and the maximum descent (MD) algorithm with 2-level LBG partition technique.

There are two issues that require further investigation. There are several variational methods of vector quantization, such as the tree structure VQ, classified VQ, hierarchical VQ and finite state VQ [10,13]. When we apply our new algorithm on these variations, can we obtain the codebooks which are still superior to the codebooks generated by the previous algorithms? It is the first issue that needs further investigation to answer this question. Second, can we apply the idea of our longest distance first (LDF) algorithm to solve the set clustering problem with different criteria? It is obvious that we can apply our LDF algorithm to solve the P-center problem. Are the solutions generated by our LDF algorithm as good as those generated by the previous algorithms? This is the second issue that requires our further investigation.

References

[1] N. Akrout, C. Allart, C. Diab, R. Prost, and R. Goutte, "A fast algorithm for vector quantiza-

Image	Execution time(sec) Algorithm						Percentage(%)		
	LBG	PNN	PNN-LBG	MD_{LBG}	LDF	MD_{LDP}	$\frac{LDF-PNN}{LDF}$	$\frac{MD_{LBG}-LDF}{MD_{LBG}}$	$\frac{MD_{LDP}-LDF}{MD_{LDP}}$
Lena	15.30	2.48	27.34	5.93	2.95	3.41	15.93	50.25	13.49
Peppers	21.22	2.41	34.00	6.10	2.94	3.48	18.03	51.80	15.52
Tiffany	31.39	2.47	41.45	6.14	3.34	3.76	26.05	45.60	11.17
F16	23.19	2.53	34.37	5.74	3.14	3.53	19.43	45.30	11.05
Kenwood	35.30	2.46	28.83	6.05	2.77	3.22	11.19	54.21	13.98
Camera	24.64	2.50	24.86	5.68	3.10	3.58	19.35	45.42	13.41
Baboon	30.72	2.45	36.82	7.14	4.17	4.51	41.25	41.60	7.54
Shuttle	22.86	2.35	26.02	6.08	3.20	3.57	26.56	47.37	10.36

(a)

Image	PSNR(dB) Algorithm						Difference(dB)		
	LBG	PNN	PNN-LBG	MD_{LBG}	LDF	MD_{LDP}	LDF-PNN	LDF-MD_{LBG}	MD_{LDP}-LDF
Lena	30.55	28.59	31.33	32.85	33.11	33.27	4.52	0.26	0.16
Peppers	30.92	29.21	32.04	33.42	33.52	33.69	4.31	0.10	0.17
Tiffany	33.75	31.43	34.04	35.66	36.05	36.15	4.62	0.39	0.10
F16	29.78	28.34	31.85	34.45	34.78	34.98	6.44	0.33	0.20
Kenwood	31.43	31.17	34.57	36.71	37.57	37.81	6.40	0.86	0.24
Camera	28.57	27.52	31.30	33.13	34.54	34.72	7.02	1.41	0.18
Baboon	25.51	23.21	25.45	25.29	25.36	25.47	2.15	0.07	0.11
Shuttle	25.40	23.13	25.91	26.56	26.82	27.02	3.69	0.26	0.20

(b)

Table 1: Performance comparison of six algorithms for local codebook generation. Image size: 256×256, grey level: 256, vector size: 4×4, codebook size: 256. (a) Execution time (b) Distortion.

tion : application to codebook generation in image subband coding," *Signal Processing VI: Theories and Applications*, Vol. 3, pp. 1227–1230, 1992.

[2] N. Akrout, R. Prost, and R. Goutte, "Image compression by vector quantization: a review focused on codebook generation," *Image and Vision Computing*, Vol. 12, No. 10, pp. 627–637, Dec. 1994.

[3] R. L. Bottemiller, "Comments on a new vector quantization clustering algorithm," *IEEE Transactions on Signal Processing*, Vol. 40, No. 2, pp. 455–456, Feb. 1992.

[4] C. K. Chan and C. K. Ma, "A fast mehtod of design better codebooks for image vector quantization," *IEEE Transactions on Communications*, Vol. 42, No. 2/3/4, pp. 237–243, Feb./Mar./Apr. 1994.

[5] R. Chen, "Solution of minisum and minimax location-allocation problems with euclidean distance," *Naval Research Logistics Quarterly*, Vol. 30, pp. 449–459, 1983.

[6] M. E. Dyer and A. M. Frieze, "A simple heuristic for the P-center problem," *Operations Research Letters*, Vol. 3, No. 6, pp. 285–288, Feb. 1985.

[7] W. H. Equitz, "A new vector quantization clustering algorithm," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. 37, No. 10, pp. 1568–1575, Oct. 1989.

[8] Y. Linde, A. Buzo, and R. M. Gray, "An algorithm for vector quantizer design," *IEEE Transactions on Communications*, Vol. C-28, No. 1, pp. 84–95, Jan. 1980.

[9] S. P. Lloyd, "Least squares quantization in PCM," *IEEE Transactions on Information Theory*, Vol. IT-28, pp. 129–137, Mar. 1982.

[10] N. M. Nasrabadi and R. A. King, "Image coding using vector quantization: a review," *IEEE Transactions on Communications*, Vol. 36, No. 8, pp. 957–971, Aug. 1988.

[11] S. Panchanathan and M. Goldberg, "Min-max algorithm for image adaptive vector quantisation," *IEE Proceedings-I*, Vol. 138, No. 1, pp. 53–60, Feb. 1991.

[12] W. K. Pratt, *Digital image processing*. New York, American: John Wiley & Sons, second ed., 1991.

[13] S. C. Tai, *Data compression*. Taipei, Taiwan: Unalis, first ed., 1996.

Image	Execution time(sec) Algorithm						Percentage(%)		
	LBG	PNN	PNN-LBG	MD_{LBG}	LDF	MD_{LDP}	$\frac{LDF-PNN}{LDF}$	$\frac{MD_{LBG}-LDF}{MD_{LBG}}$	$\frac{MD_{LDP}-LDF}{MD_{LDP}}$
Training set	243.54	25.14	401.26	30.14	9.17	14.43	63.52	69.58	36.45

(a)

Image	PSNR(dB) Algorithm						Difference(dB)		
	LBG	PNN	PNN-LBG	MD_{LBG}	LDF	MD_{LDP}	LDF–PNN	LDF–MD_{LBG}	MD_{LDP}–LDF
Lena	30.83	27.80	31.35	33.98	35.14	35.51	7.34	1.16	0.37
Tiffany	33.89	30.65	33.21	34.63	35.41	35.42	4.76	0.78	0.01
Camera	29.70	26.65	31.01	34.19	36.55	36.62	9.90	2.36	0.07
Kenwood	31.66	29.25	33.40	36.07	36.38	37.20	7.13	0.31	0.82

(b)

Image	PSNR(dB) Algorithm						Difference(dB)		
	LBG	PNN	PNN-LBG	MD_{LBG}	LDF	MD_{LDP}	LDF-PNN	LDF–MD_{LBG}	MD_{LDP}–LDF
Peppers	28.02	28.23	28.03	28.50	28.53	28.54	0.30	0.03	0.01
F16	26.44	26.92	26.63	26.89	26.91	26.96	-0.01	0.02	0.05
Baboon	22.89	23.04	22.80	22.96	22.99	23.00	-0.05	0.03	0.01
Shuttle	22.71	22.98	22.83	23.08	23.11	23.12	0.13	0.03	0.01

(c)

Image	Additional time(sec) Algorithm			PSNR(dB) Algorithm		
	MD_{LBG}	LDF	MD_{LDP}	MD_{LBG}	LDF	MD_{LDP}
Training set	109.42	233.72	187.33	35.44	36.44	36.58

(d)

Table 2: Performance comparison of six algorithms for global codebook generation. Image size: 256×256, grey level: 256, vector size: 4×4, codebook size: 1024. (a) Execution time for images within the training set. (b) Distortion for images within the training set. (c) Images outside the training set. (d) Using the codebooks generated by the MD with 2-level LBG partition algorithm, the LDF algorithm and MD with LDP algorithm as initial codebooks of the LBG algorithm.