Fast algorithm for designing better codebooks in image vector quantization

Mon-Ching Huang
Chang-Biau Yang
National Sun Yat-sen University
Department of Applied Mathematics
Kaohsiung, Taiwan 804
E-mail: 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. Thus 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 code word is a highly computational process, especially for high dimensional vectors. Our efforts are to design a fast algorithm to generate a better codebook and to reduce the computation time compared with the previous algorithms in codebook generation. Our algorithm is a top-down algorithm and is based on the longest distance first concept. Some experiments are shown to illustrate that our algorithm is superior to the previous algorithms.
[S0091-3286(97)01512-2]

Subject terms: image compression; vector quantization; codebook generation; distortion measures.

Paper 13037 received Mar. 10, 1997; revised manuscript received July 14, 1997; accepted for publication July 14, 1997.

1 Introduction

Recently, image compression has become more important because it is highly demanded in many applications such as video conferencing, television transmission, image database and archiving medical images. In image compression, the insignificant information can be removed from images. Because the images are tested by the human visual system, the quality of images depends on perceptual entities such as the contours of regions and edges. Thus, we can reduce much redundant and insignificant information to achieve a high compression ratio. Several scalar quantization techniques, such as differential pulse code modulation, transform coding and hybrid coding[1,2] have been developed. However, Shannon's rate-distortion theory states that using vectors instead of scalars always results in a better performance.[3] Therefore, vector quantization (VQ) techniques are developed for image coding applications.[1,4,5]

The Linde-Buzo-Gray (LBG) algorithm is one of the most famous algorithms for codebook generation.[6] It is an iterative method to modify the codebook to a better one, starting with an initial codebook. It is usually very time-consuming. The pairwise nearest neighbor (PNN) algorithm,[7,8] which is a bottom-up algorithm, is very different from the LBG algorithm. It begins with an individual cluster for each vector in the training set and merges two clusters together at a time until the desired codebook size is achieved or the total distortion error is reduced to a predefined threshold. Its purpose is to find a codebook quickly. However, its codebook quality is not very good. Sometimes, its codebook is used as an initial codebook of another codebook generator. The maximum descent (MD) algorithm[9] is a top-down algorithm for generating codebooks. The codebook quality of MD is usually better than that of PNN.

In this paper, we first propose a new splitting algorithm to split a cluster into two clusters. This algorithm is based on the longest distance partition heuristics. Then, using this splitting algorithm, we propose our codebook generation algorithm, called the longest distance first algorithm. Similar to the MD algorithm, the structure of our algorithm is also top-down, it splits the training set from one cluster into many clusters. By our experiments, the performance of our algorithm, including the execution time and the quality of codebook, is superior to those of the previous algorithms. Here, the distortion measure used to measure the quality of decoding images is the square error (SE).

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

2 Definitions and Notations

In the image VQ (Ref. 2), the first step is to decompose the input images into small blocks. We call these small blocks *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 code word from the codebook. The data compression is achieved by storing the codebook and the index of the best matching code word of each vector. When we want to restore the images, we use the index to find the code word 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 $V=\{\mathbf{v}_1,\mathbf{v}_2,\ldots,\mathbf{v}_N\}$ is the set of reproduction vectors (code words), and 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 $C_i=\{\mathbf{x}|Q(\mathbf{x})=\mathbf{v}_i\}$ is the set of the input vectors whose code word is $\mathbf{v}_i$.

Because we use the code words to reconstruct the image, there is distortion between the original image and the reconstructed image. The distortion measure function $d(\mathbf{x},\mathbf{v}_i)$ is used to measure the distortion between $\mathbf{x}$ and $\mathbf{v}_i$, where $\mathbf{x}$ is an input vector and $\mathbf{v}_i=Q(\mathbf{x})$. Many kinds of distortion measures have been proposed in the literature.[6] The SE is the most commonly used distortion measure because of its mathematical convenience. The definition of distortion is $d(\mathbf{x},\mathbf{v}_i)=\Sigma_{j=1}^{k}|x_j-v_{ij}|^2$, where $\mathbf{x}=(x_1,x_2,\ldots,x_k)$ and $\mathbf{v}_i=(v_{i,1},v_{i,2},\ldots,v_{i,k})$. In this paper, we use the SE as distortion measure.

3 Previous Algorithms for Codebook Generation

In this section, we briefly review three previous algorithms for codebook generation: the LBG algorithm developed by the Linde et al.[7] the PNN algorithm[7,8] and the MD method proposed by Chan and Ma.[9]

The LBG algorithm[7] is an iterative and nonvariational technique extended from the design of scalar quantizer proposed by Lloyd.[10] This algorithm is as follows. In step 1, N initial code words are chosen. There are many methods to choose the initial code words. The simplest initialization is choosing code words from the training set randomly. We can also choose evenly spaced points in the vector space as the initial codewords. In the 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. Note that the codebook generated by the PNN algorithm[7,8] can also be used as an initial codebook of the LBG algorithm if these two algorithms are concatenated. In step 2, each vector is assigned to its closest code word, based on some distortion measure and an exhaustive search. In step 3, the vectors assigned to the same code word form a set. We find the centroid of each of these sets as a new code word. The *centroid* of a set is the vector that minimizes the overall distortion error of the set. These new code words 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 are needed. 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 PNN algorithm[7,8] is a bottom-up algorithm. Initially, each vector forms an individual cluster, and each vector is the code word of its cluster. Then the clustering process begins. The clustering process merges the two nearest clusters into one cluster at a time, and finds the centroid of this new cluster as its code word. The process is performed repeatedly until the number of clusters is reduced to the desired one. And the centroids of all clusters form the final codebook. The major problem of this algorithm is how to find which two clusters are the nearest. When we merge two clusters, we must consider not only the distance of the centroids of two clusters, but also the numbers of vectors in these two clusters. Because when the numbers of vectors in these two clusters are large, it will produce large distortion error although these two clusters are nearest. The algorithm searches for a vector's near neighbors only within a small region instead of all clusters. The data structure *k-d tree*[8] (short for k-dimensional tree) is used to hasten the searching process.

The MD algorithm, proposed by Chok-Ki Chan and Chi-Kit Ma,[9] generates the codebooks with lower overall distortion errors and with much less 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 desired size of the codebook. And each code word 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 $D(C_i)=\Sigma_{\mathbf{x}\in C_i}d(\mathbf{x},\mathbf{v}_i)$, where $\mathbf{v}_i$ is the centroid of C_i. Then, the reduction of distortion error ΔD_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., $\Delta D_i=D(C_i)-[D(C_{ia})+D(C_{ib})]$. If both clusters C_{ia} and C_{ib} are not empty, ΔD_i is always positive.

Suppose now there are M clusters. The distortion reduction of each clusters is computed. Cluster C_j is selected to be split into two new clusters C_{ja} and C_{jb} if $\Delta D_j \geqslant \Delta D_i$ for $1\leqslant i\leqslant M$, $i\neq j$. Then we have $M+1$ clusters. When we continue to split the clusters, we need only to calculate the ΔD_{ja} and ΔD_{jb} of the newly formed clusters C_{ja} and C_{jb}. The ΔD_i's of the other clusters have been computed in the previous steps. Thus, to form N clusters, $N-1$ splitting operations and $2N-3$ partition searches (to find the value of ΔD) are required.

Most important in the MD algorithm is how to partition one cluster into two new clusters. Since this partition is deeply related to the reduction of the distortion, the quality of the codebook generated by the MD algorithm is dependent on the splitting technique. There are several techniques to perform the splitting operations, such as searching the optimal partitioning hyperplane and the two-level LBG algorithm. If the hyperplane partition technique is applied, the points under the hyperplane form a cluster and the others form another cluster. We have to choose the hyperplane to partition the cluster such that the reduction of distortion is maximized. But searching for this optimal partitioning hyperplane of a multidimensional cluster is difficult and computationally intensive. So, the LBG algorithm

with two levels is used instead of the optimal partitioning hyperplane. Although the reduction of distortion error is only locally optimal, due to the two-level LBG algorithm, in general, the splitting time needed by the two-level LBG algorithm is substantially reduced.

4 Algorithm for Designing a Better Codebook

The structure of our new algorithm is very similar to that of the MD algorithm.[9] We use the longest distance first strategy to choose which cluster should be split instead of the maximum descent criterion in the MD algorithm and we invoke the longest distance partition technique to partition one cluster into two new clusters instead of the two-level LBG partition technique or the hyperplane partition technique.[9]

At the beginning of our new algorithm, we view the whole training set as a global cluster. We apply 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 its longest distance, which is the distance between the centroid and the farthest vector in the cluster. We choose the cluster whose longest distance is the maximum in all clusters to split into two new clusters. This is why we call our method the *longest distance first* method. The preceding process continues until the number of clusters is increased to the desired one.

In the MD algorithm with the two-level LBG partition technique, the two-LBG algorithm is applied 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 computation of the two-level LBG partition in these clusters are useless. If an N-level vector quantizer is designed, $(N-2)$ two-level LBG partitions are performed, but not really used.

The greatest benefit of our longest distance first algorithm is time saved. We find only the longest distance in every cluster and do not calculate the reduction of distortion. Therefore, the longest distance first method requires much less time than the MD method does. Because the longest distance first algorithm 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 code words in such clusters, we can usually achieve greater distortion reduction. Therefore, usually, our longest distance first algorithm has good performance.

Before we describe our longest distance partition technique, let us consider Huygen's theorem.[4,11] We need some notation as follows:

N = number of levels
X = training set $\{\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_n\}$
C_i = i'th cluster of training vectors
n_i = number of training vectors in C_i
$\mathbf{v}_i$ = centroid (mean) of the training vectors in C_i
W = centroid (mean) of all vectors in the training set.

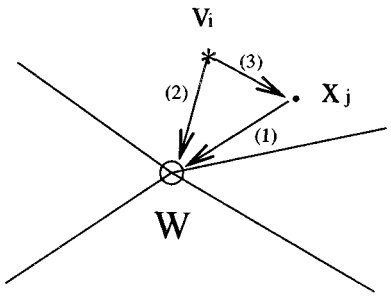


Fig. 1 Illustration of the vector summation. Vector (1)=vector (2)−vector (3), where vector (1)=$(W-\mathbf{x}_j)$, vector (2)=$(W-\mathbf{v}_i)$ and vector (3)=$(\mathbf{x}_j-\mathbf{v}_i)$.

The following terms are also defined:

d = square square error distortion measure function, $d(\mathbf{y},\mathbf{z}) = \sum_{i=1}^{k}|y_i-z_i|^2$, where $\mathbf{y}=(y_1,y_2,\ldots,y_k)$, $\mathbf{z}=(z_1,z_2,\ldots,z_k)$
E = overall distortion error, $E=\sum_{i=1}^{N}\sum_{\mathbf{x}_j\in C_i} d(\mathbf{x}_j,\mathbf{v}_i)$
∇ = intercluster error, $\nabla=\sum_{i=1}^{N} n_i d(\mathbf{v}_i,W)$
I = training set inertia, $I=\sum_{j=1}^{n} d(\mathbf{x}_j,W)$.

Akrout had shown that Huygen's theorem is $E=I'-\nabla'$,[2] where $I'=1/nI$ and $\nabla'=1/n\nabla$. 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),

$$(W-\mathbf{x}_j)=(W-\mathbf{v}_i)-(\mathbf{x}_j-\mathbf{v}_i)$$

$$\Rightarrow(W-\mathbf{x}_j)\cdot(W-\mathbf{x}_j)$$

$$=[(W-\mathbf{v}_i)-(\mathbf{x}_j-\mathbf{v}_i)]\cdot[(W-\mathbf{v}_i)-(\mathbf{x}_j-\mathbf{v}_i)]$$

$$|W-\mathbf{x}_j|^2=(W-\mathbf{v}_i)\cdot(W-\mathbf{v}_i)+(\mathbf{x}_j-\mathbf{v}_i)\cdot(\mathbf{x}_j-\mathbf{v}_i)-2(W-\mathbf{v}_i)\cdot(\mathbf{x}_j-\mathbf{v}_i)$$

$$=|W-\mathbf{v}_i|^2+|\mathbf{x}_j-\mathbf{v}_i|^2-2(W-\mathbf{v}_i)\cdot(\mathbf{x}_j-\mathbf{v}_i)$$

$$d(W,\mathbf{x}_j)=d(W,\mathbf{v}_i)+d(\mathbf{x}_j,\mathbf{v}_i)-2(W-\mathbf{v}_i)\cdot(\mathbf{x}_j-\mathbf{v}_i).$$

Consider the vectors in cluster $C_i=\{\mathbf{x}_{i,1},\mathbf{x}_{i,2},\ldots,\mathbf{x}_{i,n_i}\}$, $1\leq i\leq N$. Then

$$\sum_{j=1}^{n_i} d(W,\mathbf{x}_{i,j})=n_i d(W,\mathbf{v}_i)+\sum_{j=1}^{n_i} d(\mathbf{x}_{i,j},\mathbf{v}_i)-2(W-\mathbf{v}_i)\sum_{j=1}^{n_i}(\mathbf{x}_{i,j}-\mathbf{v}_i)$$

$$=n_i d(W,\mathbf{v}_i)+\sum_{j=1}^{n_i} d(\mathbf{x}_{i,j},\mathbf{v}_i),$$

because

$$\sum_{j=1}^{n_i}(\mathbf{x}_{i,j}-\mathbf{v}_i)=\sum_{j=1}^{n_i}\mathbf{x}_{i,j}-n_i\mathbf{v}_i=\sum_{j=1}^{n_i}\mathbf{x}_{i,j}-n_i\cdot\frac{\sum_{j=1}^{n_i}\mathbf{x}_{i,j}}{n_i}=0.$$

Thus,

$$\sum_{i=1}^{N}\sum_{j=1}^{n_i}d(W,\mathbf{x}_{i,j})=\sum_{i=1}^{N}n_i d(W,\mathbf{v}_i)+\sum_{i=1}^{N}\sum_{j=1}^{n_i}d(\mathbf{x}_{i,j},\mathbf{v}_i)$$

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

That is,

$$I=\nabla+E.$$

Thus,

$$E=I-\nabla.$$

As we can see in Huygen's theorem $E=I-\nabla$, the value of I is independent of the clustering processing, because I involves only 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 $\Sigma_{i=1}^{N}n_i d(\mathbf{v}_i,W)$, maximizing ∇ implies that each code word $\mathbf{v}_i$ must be the farthest from W. Thus, our longest distance partition technique is to try to find two code words such that these two code words are far away from the centroid of the cluster being split.

In our longest distance partition, we first find the vector that is the farthest 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 code word as the other code word. The cluster now is split into two clusters according to these two code words. That is, the vectors which are closer to the same code word form a new cluster, and the other vectors form the other new cluster. Finally, we modify the two code words 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.

4.1 *Longest Distance Partition (LDP) Algorithm*

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

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

Step 1. Calculate centroid $\mathbf{v}_i$ of the splitting cluster C_i.
Compute $d(\mathbf{x}_{i,j},\mathbf{v}_i)$, $1\leqslant j\leqslant n_i$.
Find $\mathbf{x}_{i,p}$ such that $d(\mathbf{x}_{i,p},\mathbf{v}_i)=\max_{1\leqslant j\leqslant n_i}d(\mathbf{x}_{i,j},\mathbf{v}_i)$.

Step 2. Compute $d(\mathbf{x}_{i,j},\mathbf{x}_{i,p\mathbf{x}})$, $1\leqslant j\leqslant n_i$.
Find $\mathbf{x}_{i,q}$ such that $d(\mathbf{x}_{i,q},\mathbf{x}_{i,p})=\max_{1\leqslant j\leqslant n_i}d(\mathbf{x}_{i,j},\mathbf{x}_{i,p})$.

Step 3. Split cluster C_i into $C_{i,a}$ and $C_{i,b}$. That is, if $d(\mathbf{x}_{i,j},\mathbf{x}_{i,p})<d(\mathbf{x}_{i,j},\mathbf{x}_{i,q})$ then $\mathbf{x}_{i,j}\in C_{i,a}$; otherwise $\mathbf{x}_{i,j}\in C_{i,b}$, $1\leqslant j\leqslant n_i$.

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

If our distortion measure function d is SE, Chan proposed a fast method[9] to determine which cluster $\mathbf{x}$ should belong to. When we compare our new splitting technique, LDP, with the two-level LBG partition technique, LDP requires $2n_i$ comparisons more than one iteration of the two-level LBG partition technique. But the two-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 time much less than the two-level LBG partition technique does. Moreover, because our LDP technique is trying to find the two code words that are the farthest 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 algorithm 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 that is the farthest away from the centroid of its cluster. Then, the cluster with maximum longest distance is found and we apply our LDP technique to split the cluster.

4.2 *Longest Distance First (LDF) Algorithm*

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

Output: The codebook.

Step 1. Split the entire training set into two new clusters by using our 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 by 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 go to step 2.

Although the performance of our LDF algorithm is superior to the MD algorithm with the two-level LBG partition technique, we can apply the MD strategy combined with our splitting technique, LDP, to improve the quality of codebooks. According to our experiment results, shown in the next section, the codebooks generated by the MD algorithm with LDP have higher quality than those generated by the the MD algorithm with the two-level LBG partition technique and our LDF algorithm. However, the execution time of the MD algorithm with LDP is more than that of our LDF algorithm due to the maximum descent strategy. Thus, if we want to have high coding quality, we should

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 illustrate the performance of our LDF algorithm in local and global codebook generation.[2] 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 code word 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 images are not similar to representative images.

All simulations were performed on an IBM RS6000/58H workstation, and the simulation programs were written in the C language. All input images are standard monochrome pictures with 256 gray levels, and the vector (block) size is 4×4. The distortion of the encoded image is measured by the peak signal-to-noise ratio (PSNR), which is defined as

$$\mathrm{PSNR}=10\log_{10}\left[\frac{255^2}{\frac{1}{L\times L}\sum_{i=1}^{L}\sum_{j=1}^{L}(y_{ij}-\hat{y}_{ij})^2}\right],$$

where $L\times L$ is the size of image, y_{ij} is the pixel value of the original picture at coordinate (i,j), and $\hat{y}_{ij}$ is the pixel value of the reproduced picture at coordinate (i,j) (Ref. 2).

In each performance evaluation, six codebooks generated by different algorithms are compared. The first one is generated by the LBG algorithm, and we choose evenly spaced elements in the training set to form the initial codebook of LBG. For example, $\mathbf{x}_m,\mathbf{x}_{m+(j+1)},\ldots,\mathbf{x}_{m+[(N-1)j+1]}$ can be chosen as the initial codebook, where $\mathbf{x}_i$ is the i'th training vector, m is an offset, n is the number of training vectors, N is the number of levels and $j=n/N$ (Ref. 8). 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 PNN algorithm.[7,8] 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 the LBG algorithm with the PNN codebook as its initial codebook. The fourth one is generated by the MD algorithm, using the two-level LBG partition technique. The two-level LBG partition technique uses the standard splitting technique as described in Section 3 to initialize the codebook, and the distortion threshold ε is chosen as 0.005. The fifth one is generated by our LDF algorithm. The last one is generated by the MD algorithm, using our LDP technique.

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

Tables 1 and 2 illustrate the performance of the six algorithms in local codebook generation. In Table 1, the size of each image is 256×256 and the codebook size is 256, while in Table 2, the size of each image is 512×512 and the codebook size is 1024. When we use the local codebook generation method as a compression strategy, we usually want a codebook with high quality. Thus, a full search with the partial distance elimination method is applied after the codebooks are generated.

Fig. 2 Original image of ''Lena.''

Table 3 illustrates the performance of the six 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 1 pixel, is 5/8, where bit rate $=(\log_2 N)/k$, N is the codebook size and k is the vector size.[4] The training set was extracted from ''Lena,'' ''Tiffany,'' ''Camera'' and ''Kenwood.''

According to the preceding experiments, the execution time of our LDF algorithm is less than that of each previous algorithm. And, in the local codebook generation, the codebook quality of our LDF is also superior to that of each previous algorithm, most importantly because the calculation in the LDF method is reduced and the splitting technique based on Huygen's theorem works very well. When we combine the LDP technique and the MD strategy (MD calls LDP as its partition procedure), we can get slightly higher codebook quality with little loss of execution time. If we want high codebook quality, we should to use the MD strategy with our LDP technique. If it is desired to reduce the execution time, we can use our LDF algorithm, including our LDP technique. On the other hand, when a global codebook is used, as shown in the bottom panel of Table 3, the qualities of the images outside the training set are almost the same among those algorithms. They are also not so good, which may be due to improper selection of the training set.

Table 1 Performance comparison of six algorithms for local codebook generation.

	Execution Time								
	Execution time (s)						Percentage		
Image	LBG	PNN	PNN-LBG	MD_{LBG}	LDF	MD_{LDP}	PNN−LDF/PNN	MD_{LBG}−LDF/MD_{LBG}	MD_{LDP}−LDF/MD_{LDP}
"Lena"	15.50	5.77	28.00	6.14	2.97	3.40	48.53	51.63	12.65
"Peppers"	21.84	5.84	34.19	6.42	3.01	3.45	48.46	53.12	12.75
"F16"	23.42	6.58	34.15	5.81	3.19	3.55	51.52	45.09	10.14
"Baboon"	31.05	7.02	37.26	7.24	4.29	4.64	38.89	40.75	7.544

	Distortion								
	PSNR (dB)						Difference (dB)		
Image	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
"F16"	29.78	28.34	31.85	34.45	34.78	34.98	6.44	0.33	0.20
"Baboon"	25.51	23.21	25.45	25.29	25.36	25.47	2.15	0.07	0.11

Image size: 256×256, gray level: 256, vector size: 4×4, codebook size: 256.

6 Conclusion

The set clustering problem has been investigated in many areas, examples include operational research, statistics, geometric algorithms, and data compression. The criteria of the set clustering problem are usually different due to the applications of each area. For example, the minimax location allocation problem (the P-center problem) in operational research[12] is to minimize the maximum radius of all subclusters, where the radius of each subcluster is the radius of the minimal circle that can cover all points in this subcluster. The VQ problem in data compression is to minimize the sum of distortion function between each vector and its corresponding code word. The set clustering problem is usually very difficult. For instance, the P-center problem is NP-hard.[13] Thus, many heuristic algorithms have been proposed to solve the set clustering problem with different criteria. In this paper, our efforts focused on the VQ problem in image data compression, i.e., we developed a fast heuristic algorithm to generate better codebooks.

There are two issues that require further investigation. There are several varying methods of VQ, such as the tree structure VQ, classified VQ, hierarchical VQ and finite

Table 2 Performance comparison of six algorithms for local codebook generation.

	Execution Time								
	Execution time (s)						Percentage		
Image	LBG	PNN	PNN-LBG	MD_{LBG}	LDF	MD_{LDP}	PNN−LDF/PNN	MD_{LBG}−LDF/MD_{LBG}	MD_{LDP}−LDF/MD_{LDP}
"Lena"	226.15	74.64	402.72	49.51	29.40	34.06	60.61	40.62	13.68
"Peppers"	185.08	71.14	358.68	47.42	29.54	33.76	58.48	37.71	12.50
"F16"	282.08	81.19	348.67	47.82	28.88	33.15	64.43	39.61	12.88
"Baboon"	363.48	84.06	452.85	60.72	43.32	47.20	48.47	28.66	8.43

	Distortion								
	PSNR (dB)						Difference (dB)		
Image	LBG	PNN	PNN-LBG	MD_{LBG}	LDF	MD_{LDP}	LDF−PNN	LDF−MD_{LBG}	MD_{LDP}−LDF
"Lena"	33.26	31.89	34.53	35.47	36.13	36.22	4.24	0.66	0.09
"Peppers"	32.41	31.40	34.01	35.28	35.81	35.90	4.41	0.53	0.09
"F16"	32.85	32.19	35.34	37.14	38.49	38.63	6.30	1.35	0.14
"Baboon"	25.96	23.75	25.97	26.08	26.07	26.19	2.32	−0.01	0.12

Image size: 512×512, gray level: 256, vector size: 4×4, codebook size: 1024.

Table 3 Performance comparison of six algorithms for global codebook generation.

Execution Time for Images within the Training Set

	Execution time (s)						Percentage		
Image	LBG	PNN	PNN-LBG	MD_{LBG}	LDF	MD_{LDP}	LDF−PNN/LDF	MD_{LBG}−LDF/MD_{LBG}	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

Distortion for Images within the Training Set

	PSNR (dB)						Difference (dB)		
Image	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

Images Outside the Training Set

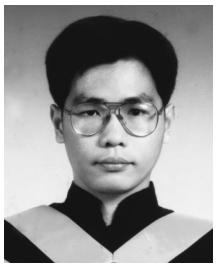
	PSNR (dB)						Difference (dB)		
Image	LBG	PNN	PNN-LBG	MD_{LBG}	LDF	MD_{LDP}	LDF−PNN	LDF−MD_{LBG}	MD_{LDP}−LDF
''Peppers''	28.02	28.23	28.03	28.50	28.53	28.54	0.30	0.03	0.01
''F16''	26.44	26.92	26.63	26.89	26.91	26.96	−0.01	0.02	0.05
''Baboon''	22.89	23.04	22.80	22.96	22.99	23.00	−0.05	0.03	0.01
''Shuttle''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

Image size: 256×256, gray level: 256, vector size: 4×4, codebook size: 1024.

state VQ (Refs. 1 and 2). When we apply our new algorithm to these variations, can we obtain the codebooks that are still superior to the codebooks generated by the previous algorithms? This is the first issue that requires further investigation to answer this question. Second, can we apply the idea of our 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 issue may be worth further study.

References

1. N. M. Nasrabadi and R. A. King, ''Image coding using vector quantization: a review,'' *IEEE Trans. Commun.* **36**(8), 957–971 (1988).
2. S. C. Tai, *Data Compression*, Unalis, Taipei Taiwan (1996).
3. W. K. Pratt, *Digital Image Processing, 2nd ed.*, John Wiley & Sons, New York (1991).
4. N. Akrout, R. Prost, and R. Goutte, ''Image compression by vector quantization: a review focused on codebook generation,'' *Image Vis. Comput.* **12**(10), 627–637 (1994).
5. S. Panchanathan and M. Goldberg, ''Min-max algorithm for image adaptive vector quantization,'' *IEE Proc. 1* **138**(1), 53–60 (1991).
6. Y. Linde, A. Buzo, and R. M. Gray, ''An algorithm for vector quantizer design,'' *IEEE Trans. Commun.* **C-28**(1), 84–95 (1980).
7. R. L. Bottemiller, ''Comments on a new vector quantization clustering algorithm,'' *IEEE Trans. Signal Process.* **40**(2), 455–456 (1992).
8. W. H. Equitz, ''A new vector quantization clustering algorithm,'' *IEEE Trans. Acoust. Speech Signal Process.* **37**(10), 1568–1575 (1989).
9. C. K. Chan and C. K. Ma, ''A fast method of design better codebooks for image vector quantization,'' *IEEE Trans. Commun.* **42**(2/3/4), 237–243 (1994).
10. S. P. Lloyd, ''Least squares quantization in PCM,'' *IEEE Trans. Inf. Theory* **IT-28**, 129–137 (Mar. 1982).
11. N. Akrout, C. Allart, C. Diab, R. Prost, and R. Goutte, ''A fast algorithm for vector quantization: application to codebook generation in image subband coding,'' *Signal Process. VI: Theories Appl.* **3**, 1227–1230 (1992).
12. R. Chen, ''Solution of minisum and minimax location-allocation problems with Euclidean distance,'' *Naval Res. Log. Q.* **30**, 449–459 (1983).
13. M. E. Dyer and A. M. Frieze, ''A simple heuristic for the *P*-center problem,'' *Operat. Res. Lett.* **3**(6), 285–288 (1985).



Mon-Ching Huang received the BS and MS degrees in applied mathematics from National Sun Yat-sen University, Kaohsiung, Taiwan, in 1994 and 1996, respectively. He is currently in the army for the military service. His research interests include image processing, data compression, and parallel processing.

Chang-Biau Yang received the BS degree in electronic engineering from National Chiao-Tung University, Hsinchu, Taiwan, in 1982 and the MS degree in computer science from National Tsing-Hua University, Hsinchu, Taiwan, in 1984. Then, he obtained his PhD degree in computer science from National Tsing-Hua University in 1988. He is currently an associate processor of the Department of Applied Mathematics, National Sun Yat-sen University. His research interests include computer algorithms, interconnection networks, fault-tolerant computing, and data compression.