Lossless Compression Schemes of Vector Quantization Indices Using State Codebook

Chin-Chen Chang
Department of Information Engineering and Computer Science, Feng Chia University, Taiwan, R.O.C.
Email: ccc@cs.ccu.edu.tw

Guei-Mei Chen
[1]Department of Computer Science and Information Engineering, National Chung Cheng University, Taiwan, R. O. C.
Email: ckm90@cs.ccu.edu.tw,

Chia-Chen Lin
Department of Computer Science and Information Management, Providence University, Taiwan, R. O. C.
Email: mhlin3@pu.edu.tw

***Abstract*—In a memoryless vector quantization system, each image block is independently encoded as a corresponding index and then an index table will be generated. In this paper, we propose two novel schemes to compress the index table without introducing extra encoding distortion. Both our schemes exploit the characteristic that the blocks of images are highly correlated to find the same or similar index from the neighboring indices in the compression process. To increase the compression, the principal component analysis (PCA) technique is also employed to sort the codewords in the codebook for minimizing the difference of neighboring indices. In addition, our second scheme excludes the repetitive indices from the search path to further decrease the bit rate. Simulation results show that our schemes are superior to SOC and traditional memoryless VQ on the compression rate.**

***Index Terms*—Vector quantization system, SOC, PCA**

I. Introduction

Vector quantization (VQ) is an effective compression scheme of digital images for the purpose of transmission and storage [3,12]. The major advantages of VQ are that the compression rate is very high and the hardware of decoder is very simple. In a VQ system, some standard images are first divided into smaller blocks for training a codebook, including representative codewords (multi-dimensional vectors), by using some previously proposed techniques such as LBG algorithm [9]. The representative codebook will be further referred in the both encoding and decoding process. In the encoding process, input vectors (multi-dimensional vectors) of the original images are then individually quantized to the closest codewords in the codebook. Compression is achieved by transmitting or storing the indices of closest codewords instead of the original pixels. In the decoding process, reconstruction of the images can be implemented by using index-based codebook look-up procedure.

The VQ scheme [3] can be mainly classified into two categories: memoryless VQ and memory VQ. In a memory VQ, image blocks are encoded dependently such as finite-state VQ (FSVQ) [2, 6]. FSVQ employs the previously encoded blocks to make a selection only from the sub-codebook of a much smaller size. Therefore, it may introduce some extra coding distortion. There are several other related techniques such as predictive VQ (PVQ) [4,5] and address VQ (AVQ) [11]. PVQ can be regarded as an extension of vector quantization. AVQ is based on exploiting the inter-block correlation by encoding a group of blocks together using an address-codebook. But in general, most of the memory VQ schemes are very complicated for hardware implementation and require more computational cost than memoryless VQ.

On the contrary, memoryless VQ schemes exploit the high correlation and the spatial redundancy between neighboring pixels; meanwhile, they ignore the spatial redundancy between neighboring image blocks. In other words, each image block is encoded independently, while its corresponding index is sent to the decoder. Several efficient techniques have been proposed, such as index-compressed VQ (ICVQ) [13], predictive mean search algorithm (PMS) [10], and search-order coding (SOC) [7], etc. The SOC scheme further achieves the goal of compression in the index domain rather than in the pixel domain. Its main concept is to search the same index from the neighboring indices of the current processing index and then use search-order codes as the encoded codes. Compression can be achieved when the length of binary representation of search-order codes is shorter than the number of bits required by the original index value. It has been proved that SOC is efficient in improving the image compression rate [7].

In this paper, we propose two novel schemes to further compress the VQ index table of images based on the SOC scheme without losing any image quality by the original

VQ encoding. To attain our goal, at first, we rearrange the order of the codewords in a codebook by using the principle component analysis (PCA) technique [1, 8] such that neighboring image blocks are likely to be encoded by similar indices. Furthermore, the concept of state codebook is then exploited to keep the closest codewords of these similar indices in the codebook. The main idea behind our schemes is that we try to search a same index with the current processing index from state codebook, if we could not directly find a same index from the neighboring indices. Besides, our second scheme excludes the repetitive indices from the search path to further improve our first scheme. Simulation results indicate that both our newly proposed schemes achieve significant reduction of bit rate without losing any image quality by the original VQ encoding.

This paper is organized as follows. Section 2 presents a brief review of some related works, including the PCA technique [8] and SOC algorithm [7]. In Section 3, the proposed schemes shall be described in detail. Finally, in Section 4, simulation results shall be given, followed by some conclusions in Section 5.

II. Related Works

For the purpose of minimizing the difference of neighboring indices, the principal component analysis technique showed in Subsection A will be first exploited to sort the codewords in a codebook. And then, the VQ and search-order coding (SOC) scheme will be described in Subsections B and C, respectively.

A. The Principal Component Analysis (PCA) Technique

PCA is an efficient technique for sorting multi-dimensional vectors. The concept of PCA is to find a special axis onto which all multi-dimensional vectors can be projected. Through these projections, the corresponding projected points on the special axis can keep the greatest relative differences between any two multi-dimensional vectors. It means that sorting of multi-dimensional vectors can be achieved by sorting their corresponding projected points. Now the main problem is how to find the special projection axis. An algorithm which has been proposed by Lee et al. [8] for finding the special axis is as follows:

Algorithm of PCA

Problem: For m given multi-dimensional vectors $V_1, V_2, \ldots, V_m$, each vector is of n dimensions. The output is a special projection axis $D_1 = (d_1, d_2 \ldots, d_n)$, called as first principal component direction, where $\sum_{i=1}^{n} d_i^2 = 1$ such that $V_1, V_2, \ldots, V_m$, can be projected onto D_1. And all projected points can keep their greatest relative differences between each other.

Step 1: In order to eliminate influence from the different measure, we must first normalize the difference measure if $V_1, V_2, \ldots, V_m$, do not have the same difference measure.

Step 2: Compute the covariance matrix ***C*** according to all normalized multi-dimensional vectors $V_1, V_2, \ldots, V_m$.

Step 3: Find all eigenvalues of the covariance matrix ***C*** and denote them as $\lambda_1, \lambda_2, \ldots, \lambda_n$, where $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$. Compute $D_1, D_2, \ldots, D_n$ which are the corresponding eigenvectors of $\lambda_1, \lambda_2, \ldots, \lambda_n$.

Step 4: Output the special axis, $D_1 = (d_1, d_2 \ldots, d_n)$, called as first principal component direction.

Once the special projection axis $D_1 = (d_1, d_2 \ldots, d_n)$ was generated, for each given multi-dimensional vector $V_i = (v_{i,1}, v_{i,2}, \ldots, v_{i,n})$, where $i = 1, 2, \ldots, m$, compute its projected point onto $D_1 = (d_1, d_2, \ldots, d_n)$ by using the expression $v_{i,1} \times d_1 + v_{i,2} \times d_2 + \ldots + v_{i,n} \times d_n$. Sorting of the multi-dimensional vectors can then be performed according to sorting of their projection points.

B. Review of Vector Quantization Algorithm

In recent years, there is an explosive growth in the networks for transmitting digital media or other types of files. With the rapid development of the network technologies, digital media can be transmitted directly via the computer network. Therefore, how to reduce the amount of the transmitted data becomes an important issue. To handle this issue, data compression is employed. Data compression is a critical technique to diminish the amount of data such as images, audio, and video. One of the most common and popular data compression techniques is VQ, which was proposed by Gray in 1984 [3] and is an important technique for low-bit-rate image compression. In the following, the detail of VQ, which will be applied to our proposed method, is introduced.

VQ consists of three steps: (1) codebook design, (2) the encoding phase and (3) the decoding phase. At first, a codebook K which is composed of the most representative codewords must be constructed. Then K will be employed in both the encoding phase and the decoding phase. Generating a perfect codebook from a large amount of training set is an essential work in VQ. Even though many codebook design algorithms have been proposed, the most famous one is LBG algorithm which was presented by Linde, Buzo and Gray in 1980 [9]. LBG algorithm for designing the codebook is an iterative algorithm that splits the training sets and updates the codebook iteratively. A perfect codebook is generated by a good initial guess and a fast convergence to obtain the most representative codewords.

In the encoding phase, the input image S is divided into many non-overlapping square blocks of size p×p pixels, and each of them stands for a p2-dimension vector. The set of vectors is represented by $S = \{S_1, S_2, \ldots, S_m\}$, where m is the number of vectors in S.

Suppose that the finite set $K = \{k_i \mid i = 0, 1, \ldots, n-1\}$ is the codebook of size n, where $k_i = (k_{i1}, k_{i2}, \ldots, k_{ip^2})$ is the ith codeword. Thereupon, VQ can be regarded as a mapping function Q from a p2-dimension Euclidean space R^{p^2} to a finite subset $K \subset R^{p^2}$:

$Q : R^{p^2} \to K$.

The encoder designs a desired mapping function Q such that the Euclidean distance between the input vector S_j and the mapped codeword $Q(S_j) = k_x$ is the shortest. The Euclidean distance is often used to measure the distortion and defined as follows.

$$\sqrt{\sum_{a=1}^{p\times p}(s_{ja} - k_{xa})^2} = \min_{t=1}^{n}\sqrt{\sum_{a=1}^{p\times p}(s_{ja} - k_{ta})^2}, \text{ for } 1 \le j \le m, \text{ and } 1 \le x \le n.$$

As a result, the corresponding distortion is minimal. Then, as shown in Figure 1, the index *x* of codeword k_x which is mapped from the input vector is transmitted to the decoder rather than the input vector *Sj*. Owing to that the input vector is replaced by the index of k_x, the number of bits used for representing the index *x* of the codeword k_x is smaller than that of the input vector *Sj*. Thus, the original image is compressed. For example, assume that the size of the codebook is set to be 256, and each codeword is represented by a 16-dimension vector in general. And, bpp, defined the number of the required bits for one pixel, is applied to measure the compression ratio. The bpp value of the compressed image is 0.5 since the number of bits used for representing the codeword is eight in the above example. In terms of image compression, the good compression ratio is provided by VQ.

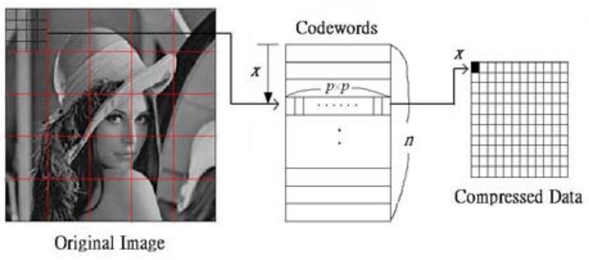


Figure 1. The encoding phase of VQ.

In the decoding phase, the codebook of the decoder is identical to that of the encoder. The decoder receives each index x which is transmitted by the encoder, and an easy look-up table operation is performed to find the corresponding codeword k_x. As a result, the codeword k_x is employed to reconstruct the input vector *Sj*. The original image is reconstructed by repeating the above processes for all the received indices. The computation load of VQ is quite low such that it is superior to other compression techniques. VQ provides the good compression ratio and the good image quality of the reconstructed image. The size of codebook can be adjusted to be larger if we want get the higher PSNR value of the recovered image. Note that the greater the PSNR value of the reconstructed image is, the worse the compression ratio will be.

C. Review of the Search-Order Coding Algorithm

The proposed search-order coding (SOC) algorithm [7] is a technique to further achieve image compression on the index table of VQ [3] and does not cause coding distortion again. Its main concept is based on two image characteristics: the neighboring blocks are highly corresponding to the same indices and an image is generally encoded block by block in a raster scan order (i.e. from left to right and from top to bottom). For an index table, SOC searches the previously generated VQ indices and uses the corresponding search-order codes as the compression codes if any previous index matches the current index. Compression can be achieved when the search-order codes which are binary representations are less than the original VQ index. Now the problems are to find the matched one of previous indices with the current

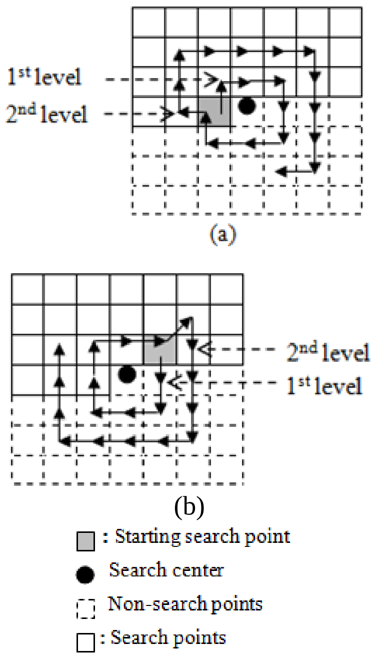


Figure 2. Search paths of SOC (a) SSP = 1 (b) SSP = 4.

index and generate the relative search-order codes.

According to the raster scan order, the current index is denoted as a search center (SC) and one of previous encoding indices is the starting search point (SSP). For example, in Figure 2, there are two 7×7 index tables, where the black dot is the SC and starting search point (SSP) is defined in Figure 3. It is obvious that these two index tables (shown in Figures 2(a) and 1(b)) demonstrate two different search paths for searching a same index value with the SC. The solid boundary squares are the search points (SP) which are previously encoded based on the raster scan order. In other words, non-search points can not be considered in a search path because they are encoded after SC. Note that in each search, it must search the SPs along the search path, 1st level, 2nd level, …, and so on, until the first same index is found or none of the SPs is matched.

In Figure 3, there are four starting search points (SSP) defined by the SOC algorithm. To start a search, the SSP must be determined at first. And then four SPs in the 1st level will be searched for. If any same index is found, the search-order codes will be sent to the decoder. Otherwise, the SPs in the 2nd level will be considered. If none of the SPs is matched, the original index value of SC is sent. To distinguish the search-order codes from the original index value by the decoder, an extra bit, called an indicator, is also sent to the decoder in the front of them.

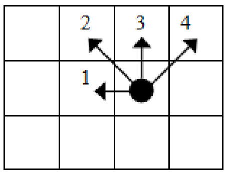


Figure 3. Four starting search points (SSP).

Based on our observation, there are usually many repetition indices in the neighborhood of the SC. A repetition index means that its index value is equal to the previous SP along the search path. If they are excluded in a search path, it will increase the probability of finding the matched one. For example, in Figure 4, the index of (3,3), which equals to 27, is the SC and SSP equals to 1. Here the repetition indices are (2,2) and (3,1), which are equal to 29 and 32, respectively. Once they are included, the matched index of (2,1), which also equals to 27 will be lost in this search because the number of bits assigned to search-order codes, n, is limited. In this example, n is equal to 2. If we ignore the repetition indices, we can find the corresponding search-order codes, 11, of the SC.

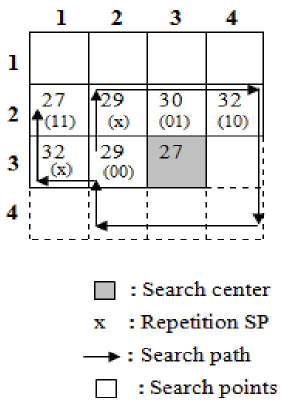


Figure 4. A 4×4 index table for showing the exclusion of repetition points.

In the decoding process, the decoding order is also in the raster scan order. After received the compression codes, the decoder can distinguish the search-order codes form the original index value according to its indicator. Then the similar search procedure with the encoding process is performed. Note that the values of SSP and n are equal to those of the encoding process. When the index table is recovered, the image can be reconstructed by using the traditional VQ codebook look-up procedure.

III. The Proposed Scheme

In this section, we propose two schemes based on the SOC algorithm in improving compression rate of images. The first scheme exploits the basic state codebook to keep the closest codewords, and the second scheme improves the state codebook for further reducing the bit rate. They will be described in Subsection A and B in detail, respectively.

A. The Basic State Codebook Scheme

By the SOC algorithm, it just searches the same index value around the current processing index. In other words, SOC ignores the similar indices, if it fails to find the same one. For the purpose of further increasing the compression rate, our scheme finds the similar indices around the current encoding index and then encodes it as the shorter length than its original index value. To make sure the difference of neighboring indices is minimal, we apply the principal component analysis (PCA) technique to sort the codewords in the codebook that generated by the original VQ encoding. In addition, the raster scan order (i.e., from left to right and top to bottom) is also applied in our scheme to scan the index table.

In our first proposed encoding scheme, all the indices in the index table shall be classified into three groups. In the first group, it contains the indices that can be encoded as search-order codes by using SOC. In the second group, it contains the indices that can be encoded by using state codebook. The state codebook is a much smaller sub-codebook of the original VQ codebook (i.e., super codebook) and consists of the indices of the closest codewords. Note that the codewords in the original VQ codebook have been sorted by the PCA technique, the closest codewords of each codeword may appear nearby. It is easy to find the similar indices for the current index. Furthermore, the state codebook is generated in real-time in the both encoding and decoding processes. It is not necessary to store the state codebooks. In the third group, it contains the rest indices excluded from the first and second group in the index table. In order to distinguish these three groups in the compression codes by the decoder, an extra indicator (one or two bits) should be added in the front of the compression codes of each index.

To begin the encoding process, all the indices in the index table shall be encoded sequentially. For the current processing index, we first check whether this index belongs to the first group or not. The SOC algorithm is executed to search around this index for finding a same one in the predefined search order. If the result is positive, this index is classified into the first group. Then, the compression codes of this index are an indicator plus search-order codes. According to our simulation results, the indicator in the first group is one bit instead of two bits, achieves better compression rate.

If the current processing index does not belong to the first group, we further check whether this index belongs to the second group or not. If this index does not belong to the first group, the similar indices with the current index may exist in the predefined search path. In our scheme, we search a same index with the current index from the closest codewords of those similar indices. It means that the state codebook will be generated in real-time and consists of several closest codewords. Since the codebook has been sorted by PCA technique, the closest codewords of each index in the codebook must appear nearby. Note that the number of those similar indices which we want to search for is predefined and denotes as the parameter, 2^i. The number of closest codewords of each similar index is also predefined and denotes as 2^j. Hence, the size of state codebook will be $2^i \times 2^j$. Once we find a same index with the current index in the state codebook, the current index is classified into the second group. The compression codes of this index are an indicator plus the index, which is corresponding to the found same index, of the state codebook. Here the indicator is two bits and the number of bits of the index for the state codebook is $i + j$.

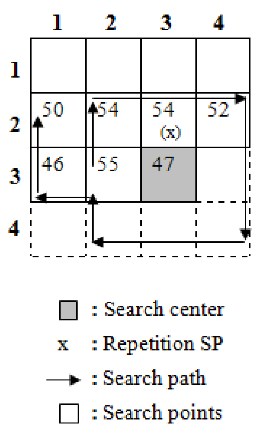


Figure 5. An example of 4×4 index table for generating the state codebook

For the example of Figure 5, first, we assume that the number of bits of search-order codes, *m*, equals to 2 and search direction, SSP, equals to 1 by executing the SOC algorithm. We can find that the current index (3,3) does not belong to the first group because we can not find a same index in the predefined search path. Here its similar neighboring indices are the indices of (3,2), (2,2), (2,4) and (3,1), which equal to 55, 54, 52 and 46. Now we start to check whether the current index belongs to the second group or not. Assume that $i = 2$ and $j = 2$, then the size of the state codebook will be $2^2 \times 2^2 = 16$. The state codebook is shown in Table 1, where we define that there are $2^j / 2$ indices of the closest codewords above each similar index and $2^j / 2$ ones below it.

TABLE I.
THE STATE CODEBOOK OF FIGURE 5

The similar neighboring index (3,2), which equals to "55".	
The index of the state codebook.	The indices of the closest codewords of "55" in the codebook.
"0000"	53
"0001"	54
"0010"	56
"0011"	57
The similar neighboring index (2,4), which equals to "52".	
The index of the state codebook.	The indices of the closest codewords of "52" in the codebook.
"1000"	50
"1001"	51
"1010"	53
"1011"	54
The similar neighboring index (2,2), which equals to "54".	
The index of the state codebook.	The indices of the closest codewords of "54" in the codebook.
"0100"	52
"0101"	53
"0110"	55
"0111"	56
The similar neighboring index (3,1), which equals to "46".	
The index of the state codebook.	The indices of the closest codewords of "46" in the codebook.
"1100"	44
"1101"	45
"1110"	**47**
"1111"	48

From Table I, the same index with the current index in the state codebook was found at the index, "1110", of the state codebook. It means that the current index belongs to the second group. Therefore, in this example, the compression codes of the current index (3,3), which equals to 47, are two bits indicator plus "1110".

Once no match is found in the state codebook for the current index, the current index belongs to the third group. Its compression codes will be a two-bit indicator plus its original VQ index value. Note that each index in the index table will be just classified into only one group. A detail description of the proposed algorithm is given as follows:

The Proposed Encoding Algorithm
Input: Each block of an image using the raster scan order.
Output: The compression codes of the input image.

Step 1: Aiming at the current processing block, exploit traditional VQ encoding scheme to generate its corresponding index. If the index belongs to the first group, a one-bit indicator ("0" is commonly in use) plus the search-order codes are outputted as its compression codes and go to Step 4. Otherwise, go to Step 2.
Step 2: If the index belongs to the second group, a two-bit indicator ("10" is commonly in use) plus the index of the state codebook are outputted and go to Step 4. Otherwise, go to Step 3.
Step 3: The index belongs to the third group. A two-bit indicator ("11" is commonly in use) plus the original VQ index value are outputted. And then go to Step 4.
Step 4: Repeat Steps 1 to 4 until all the blocks in the input image are processed.

As for the decoder, the decoding process is also in the raster scan order. According to the indicator of each index of the received compression codes, the decoder can distinguish each group. If the indicator indicates the first group, the index will be recovered by using the SOC algorithm. On the contrary, if the indicator indicates the second group, the same state codebook with the previous encoding process will be generated for recovering this index. Finally, if the indicator indicates the third group, the index will be recovered by using its original VQ index value.

B. The Optimized State Codebook Scheme

This improvement is designed on the observation that there may exist some repetition indices in the basic state codebook presented in Subsection 3.1. The repetition index signifies an index whose value is equal to that of previous indices that have been stored in the state codebook and the similar indices searched in the predefined search path. For the example of Table 1, the repetition indices are "0101", "0111", "1010" and "1011". Due to these repetitions, the search procedure of the state codebook will yield redundant search and thereby decrease the coding efficiency. Therefore, the exclusion of repetition indices will increase the possibility for finding matched index from the optimized state codebook because the number of closest codewords, 2^j, the same definition with Subsection 3.1 is limited.

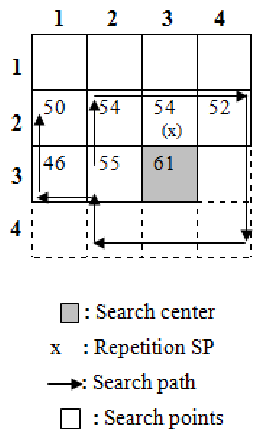
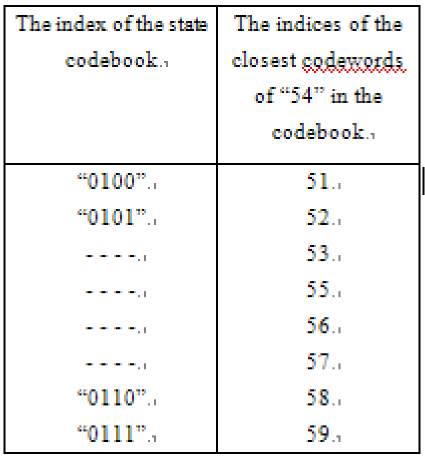


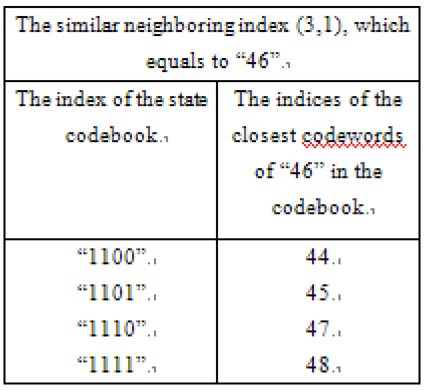
Figure 6. An example of 4×4 index table for generating the optimized state codebook

For the example of Figure 6, the current index (3,3), which equals to 61, will be classified into the third group if we do not exclude repetition indices from the state codebook. Table 2 shows the optimized state codebook of the current index (3,3). In this case, (3,3) will be classified into the second group because the matched index is "1011". According to the simulation results, Both the basic state codebook and optimized state codebook schemes significantly achieve lower bit rate than the SOC scheme.

TABLE II.
THE OPTIMIZED STATE CODEBOOK OF FIGURE

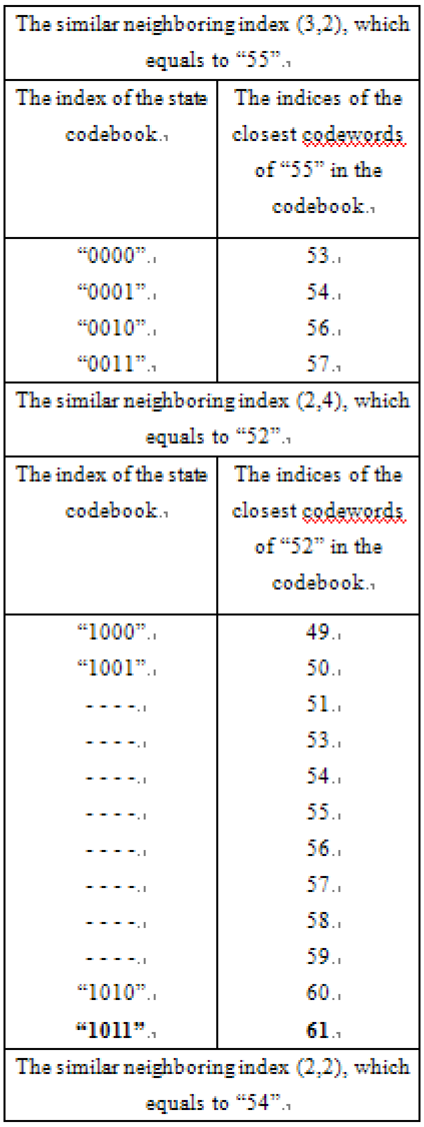


The index of the state codebook.	The indices of the closest codewords of "54" in the codebook.
"0100".	51.
"0101".	52.
- - - -.	53.
- - - -.	55.
- - - -.	56.
- - - -.	57.
"0110".	58.
"0111".	59.



The similar neighboring index (3,1), which equals to "46".	
The index of the state codebook.	The indices of the closest codewords of "46" in the codebook.
"1100".	44.
"1101".	45.
"1110".	47.
"1111".	48.

6



The similar neighboring index (3,2), which equals to "55".	
The index of the state codebook.	The indices of the closest codewords of "55" in the codebook.
"0000"	53
"0001"	54
"0010"	56
"0011"	57
The similar neighboring index (2,4), which equals to "52".	
The index of the state codebook.	The indices of the closest codewords of "52" in the codebook.
"1000"	49
"1001"	50
- - - -	51
- - - -	53
- - - -	54
- - - -	55
- - - -	56
- - - -	57
- - - -	58
- - - -	59
"1010"	60
"1011"	**61**
The similar neighboring index (2,2), which equals to "54".	

IV. EXPERIMENTAL RESULTS

Experiments have been performed for the two proposed lossless index compression schemes using several 512×512 monochrome standard images with 256 gray levels shown in Figure 7. As for our codebook, which was designed by using six standard images, Airplane, Boat, Lena, Sailboat, Tiffany and Toys, of 512×512 pixels with 256 gray levels and the well-known LBG algorithm [9]. In our experiments, the size of each image block is 4×4 pixels. The comparisons of compression rate between each scheme shall be shown with bit rate (bits/pixel, i.e., bpp) system. In other words, the bit rate of the proposed scheme is given by

$$BR = \frac{NO_1 \times (1+m) + NO_2 \times (2+i+j) + NO_3 \times (2+\log_2 N_c)}{M \times N}$$

where N_c: codebook size; $M \times N$: image size; *m, i, j* : the same definitions as in Section 3; NO_1, NO_2, NO_3: the total number of indices classified into the first group, second group and third group, respectively.

Figure 7. Standard images of 512×512 pixels

TABLE III.
BIT RATE OF THE PROPOSED SCHEMES AND THE SOC ALGORITHM USING A VQ CODEBOOK WITH 256 CODEWORDS.

Schemes / Images	SOC (*n*=2) [7]	Scheme 1	Scheme 2
Airplane	0.3241	0.3151	0.3105
Lena	0.3567	0.3469	0.3369
Peppers	0.3510	0.345	0.3366
Sailboat	0.3651	0.3579	0.3525
Tiffany	0.2813	0.2627	0.2581
Toys	0.2680	0.2607	0.2572

Note: Scheme 1 is the basic state codebook scheme and scheme 2 is the optimized state codebook scheme.

Table III shows the comparison of the bit rate by using our designed codebook with 256 codewords. Note that for the traditional memoryless VQ, the bit rate is 0.5 bits/pixel for codebook of size 256. In Table III, the implemental parameters of SOC, *n* and SSP, equal to 2 and 1 suggested by the authors [7]. The former means the number of bits assigned to the search-order codes is 2 and the latter means the starting search point is 1. In addition, our Scheme 1 and Scheme 2 are designed using the basic state codebook and the optimized state codebook proposed in Subsection 3.A and 3.B, respectively. Here, the parameters, *m*, *i*, *j*, are 2, 2, 3, which are predefined. From Table III, the bit rates of our schemes are obviously less than that of the SOC scheme. Furthermore, the bit rate of Scheme 2 is also less than Scheme 1 since the exclusion of the repetition indices in the state codebook.

TABLE IV.
BIT RATE OF THE PROPOSED SCHEMES AND THE SOC ALGORITHM USING A VQ CODEBOOK WITH 512 CODEWORDS.

Schemes / Images	SOC (*n*=2) [7]	Scheme 1	Scheme 2
Airplane	0.4331	0.4082	0.3964
Lena	0.4523	0.4386	0.4263
Peppers	0.4425	0.4489	0.4385
Sailboat	0.4577	0.4474	0.4374
Tiffany	0.3474	0.3238	0.3162
Toys	0.3007	0.2906	0.2875

Note: Scheme 1 is the basic state codebook scheme and scheme 2 is the optimized state codebook scheme.

Table IV and V show the experimental results in which the parameters are chosen the same with Table 3, except the size of codebook are 512 and 1024, individually. Note that for codebook of sizes 512 and 1024, the bit rates of the traditional memoryless VQ scheme are 0.5625 and 0.625. From the experiment, it is obvious that our scheme 1 and 2 still present better performance than the SOC algorithm and traditional memoryless VQ scheme no matter even in the worse scenario.

TABLE V.
BIT RATE OF THE PROPOSED SCHEMES AND THE SOC ALGORITHM USING A VQ CODEBOOK WITH 1024 CODEWORDS.

Schemes / Images	SOC (*n*=2) [7]	Scheme 1	Scheme 2
Airplane	0.4925	0.4634	0.4499
Lena	0.5484	0.5413	0.5330
Peppers	0.5288	0.5242	0.5154
Sailboat	0.5428	0.5363	0.5281
Tiffany	0.4075	0.3694	0.3592
Toys	0.3604	0.3401	0.3371

Note: Scheme 1 is the basic state codebook scheme and scheme 2 is the optimized state codebook scheme.

V. CONCLUSIONS

Two newly schemes have been proposed for VQ index compression. They are designed based on the observation that the high correlation of neighboring blocks in an image. In order to employ this characteristic, the only thing we need to do is sorting of the codewords in a codebook. With the help of the PCA technique, this aim is easily attained. To increase the compression rate, both our schemes first uses the search-order coding algorithm to find the same one in the neighboring indices. After that, the technique of processing the similar indices is to exploit state codebook for recording of the closest codewords of these similar indices. Hence, for generating the compression codes and recovering the original index, the index-based state codebook look-up procedure will be performed. Besides, to further reduce the bit rate, our scheme 2 excludes the repetitive indices from the search path. Our experiments prove the scheme 2 indeed present better performance than both of the SOC scheme and our scheme 1. Comparing to the traditional memoryless VQ system and the SOC algorithm, both our proposed schemes truly achieve the goal of reducing the bit rate indeed.

REFERENCES

[1] C. C. Chang, D. C. Lin and T. S. Chen, “An improved VQ codebook search algorithm using Principal Component Analysis,” *J. Vis. Commun. Image Representation*, Vol. 8, No. 1, 1997, pp. 27-37.

[2] J. Foster, R. M. Gray and M. O. Dunham, “Finite-state vector quantization for waveform coding,” IEEE Trans. on Information Theory, Vol. IT-31, May 1985, pp. 348-359.

[3] R. M. Gray, “Vector quantization,” *IEEE ASSP Mag.*, Vol. 1, No. 2, Apr. 1984, pp. 4-29.

[4] R. M. Gary and Y. Linde, “Vector quantization and predictive quantizers for Gauss-Markov sources,” *IEEE Trans. on Communs.*, Vol. COM-30, Feb. 1982, pp. 381-389.

[5] H. M. Hang and J. M. Woods, “Predictive vector quantization of images,” *IEEE Trans. on Communs.*, Vol. COM-33, No. 11, Nov. 1985, pp. 1208-1219.

[6] C. H. Hsieh and J. S. Shue, “Frame adaptive finite-state vector quantization for image sequence coding,” Image Commun., Vol. 7, Mar. 1995, pp. 13-26.

[7] C. H. Hsieh and J. C. Tsai, “Lossless compression of VQ index with search-order coding,” *IEEE Trans. on Image Processing*, Vol. 5, No. 11, 1996, pp. 1579-1582.

[8] R. C. T. Lee, Y. H. Chin and S. C. Chang, “Application of Principal Component Analysis to Multikey Searching,” *IEEE Trans. on Software Engineering*, Vol. SE-2, No. 3, 1976, pp. 185-193.

[9] Y. Linde, A. Buzo and R. M. Gray, “An algorithm for vector quantizer design,” *IEEE Trans. on Communs.*, Vol. 28, 1980, pp. 84-95.

[10] K. T. Lo and J. Feng, “Predictive mean search algorithms for fast VQ encoding of images,” *IEEE Trans. on Consumer Electronics*, Vol. 41, No. 2, 1995, pp. 327-331.

[11] N. M. Nasrabadi and Y. Feng, “Image compression using address-vector quantization,” *IEEE Trans. on Communs.*, Vol. 38, No. 12, 1990, pp. 2166-2173.

[12] N. M. Nasrabadi and R. A. King, “Image coding using vector quantization : A review,” *IEEE Trans. on Communs.*, Vol. 36, No. 8, Aug. 1988, pp. 957-971.

[13] J. Shanbehzadeh and P. O. Ogunbona, “Index-compressed vector quantization based on index mapping,” *IEE Proceedings – Vision, Image and Signal Processing*, Vol. 144, 1997, pp. 31-38.

Chin-Chen Chang received his BS degree in applied mathematics in 1977 and the MS degree in computer and decision sciences in 1979, both from the National Tsing Hua University, Hsinchu, Taiwan. He received his Ph.D in computer engineering in 1982 from the National Chiao Tung University, Hsinchu, Taiwan. During the academic years of 1980-1983, he was on the faculty of the Department of Computer Engineering at the National Chiao Tung University. From 1983-1989, he was on the faculty of the Institute of Applied Mathematics, National Chung Hsing University, Taichung, Taiwan. From August 1989 to July 1992, he was the head of, and a professor in, the Institute of Computer Science and Information Engineering at the National Chung Cheng University, Chiayi, Taiwan. From August 1992 to July 1995, he was the dean of the college of Engineering at the same university. From August 1995 to October 1997, he was the provost at the National Chung Cheng University. From September 1996 to October 1997, Dr. Chang was the Acting President at the National Chung Cheng University. From July 1998 to June 2000, he was the director of Advisory Office of the Ministry of Education of the R.O.C. From 2002 to 2005, he was a Chair Professor of National Chung Cheng University. Since February 2005, he has been a Chair Professor of Feng Chia University. In addition, he has served as a consultant to several research institutes and government departments. His current research interests include database design, computer cryptography, image compression and data structures.

Guei-Mei Chen received her MS degree in computer science and information engineering in 2003 from National Chung Cheng University, Taiwan, R. O. C. Her research interests include image compression and image processing.

Chia-Chen Lin received her BS degree in information management in 1992 from the Tamkang University, Taipei, Taiwan. She received both her MS degree in information management in 1994 and Ph.D. degree in information management in 1998 from the National Chiao Tung University, Hsinchu, Taiwan. Dr. Lin served as Visiting Associate Professor at the Business School of the University of Illinois at Urbana Champaign during the period of August 2006 to July 2007. The visiting scholarship was appointed and sponsored by the Ministry of Education, Taiwan. During her visit, she worked with Professor Klara Nahrstedt at the Department of Computer Science on copy detection and the collaboration continues. Dr. Lin is currently a professor of the Department of Computer Science and Information Management, Providence University, Sha-Lu, Taiwan. Since August 2008, she is the Associate Dean of Academic Affairs of Providence University. She is also a member of IEEE and a member of ACM. Her research interests include image and signal processing, image hiding, mobile agent, and electronic commerce.