

# A general codebook design method for vector quantization

Rui Li<sup>1</sup> · Zhibin Pan<sup>1</sup> · Yang Wang<sup>1</sup>

Received: 1 June 2017 / Revised: 18 January 2018 / Accepted: 22 January 2018 /  
Published online: 2 February 2018  
© Springer Science+Business Media, LLC, part of Springer Nature 2018

**Abstract** Vector quantization (VQ) is widely used in image processing applications, the primary focus of VQ is to determine a codebook to represent the original image well. In order to make a codebook perform better on both distortion and bit rate (BR), a general codebook (GCB) for VQ is proposed in this paper. Unlike common codebook (CCB) or private codebook (PCB), GCB is a new structure of codebook where the codewords can either come from CCB or by training the input image. By applying the codewords in CCB that perform well and updating inactive codewords, only the new generated codewords and flags of codewords to be replaced are transmitted along with index table (IT). Therefore, the BR can be significantly reduced while the performance of distortion can be efficiently improved. The experimental results demonstrate that our proposed GCB has a better performance than CCB and various kinds of PCB-based methods.

**Keywords** Vector quantization · General codebook (GCB) · Private codebook (PCB) · Common codebook (CCB) · Bit rate (BR)

## 1 Introduction

The amount of data in an image is getting larger and larger with the rapid development of image resolution. Therefore, the efficient encoding method is more and more necessary. Such as video encoding methods in paper [26] compress the video by eliminating the redundancy in frames. Zhou tries to eliminate the near-duplicate information to compress the data in the

---

✉ Zhibin Pan  
zbpan@mail.xjtu.edu.cn

Rui Li  
pofazhelinui@126.com

Yang Wang  
wty2977892@126.com

<sup>1</sup> School of Electronic and Information Engineering, Xi'an Jiaotong University, No. 28, Xianning West Road, Xi'an 710049, People's Republic of China

sensor networks [37]. The methods to eliminate redundancy are variable, Ma employs clustering algorithm to process the overlapping community structure [22]. Paper [25] compresses the redundancy in frames by predicting. Generally, large amount of storage and bandwidth are needed for image storing and transmitting, thus a lot of image compression algorithms are developed to save the resources for processing an image. The vector quantization (VQ) scheme is a widely used image compression scheme due to its simple architecture, fast decoding ability and high compression rate. In addition to image compression, VQ has also been applied to other fields, such as data hiding [3], facial recognition [5], image resolution enhancement [4], image authentication [7, 31], image indexing and retrieval [36].

The concept of VQ was firstly developed by Gray for image compression [17]. The VQ method cuts an input image into many image blocks and encodes these blocks by a well-designed codebook, then the indices are stored as the compressed image. In this way, an image is compressed because the indices use much less bits than the input image. Basically, the VQ scheme can be divided into three phases: codebook design, image encoding and image decoding. It is obvious that how to design a good codebook is a key research topic to improve the performance of VQ.

The most widely reported and studied algorithm for training a codebook is Linde-Buzo-Gray (LBG) method [17], which was developed by Lloyd from one-dimensional case to k-dimensional case [19]. The optimization of a codebook begins from an initial codebook and it creates a final codebook via several iterations. A lot of codebook design algorithms are studied [16, 24, 30, 35] and LBG algorithm is the most popularly used method due to its good performance and ease of implementation.

Usually, there are two kinds of codebooks, i.e. the common codebook (CCB) and the private codebook (PCB) [12]. They are distinguished by the way that the codebooks are generated and saved. The CCB is generated by a set of training images that are commonly used in image compression and this CCB is saved for all images. CCB is trained off-line and stays the same for all input images, many methods can be employed to obtain CCB, the paper [23] introduce a codebook generating method by artificial bee colony (ABC) and genetic algorithms, the paper [20] gives a series of codebook design methods, such as enhanced LBG, PCA and neural networks methods. When an image is encoded by CCB, only the indices need to be transmitted, the CCB performs well on most images, while PCB is a codebook purposely trained by a given input image to be compressed. Compared to CCB, the PCB has a better distortion performance for the given input image, but the PCB must be transmitted to the decoder and that will lead to a large overhead of transmission. The difference between CCB and PCB is not the training algorithms they employed, but the way codebook generated, CCB is obtained off-line by training a set of test images, while PCB is obtained on-line for each input image.

In this paper, we focus on the codebook design phase of the VQ algorithm. To further improve the image compression performance on both BR and distortion, our approach extends the advantages of CCB and PCB. It is a more general version of codebook, which implies that either PCB or CCB can be viewed as a special case of our proposed approach.

The rest of this paper is organized as follows: in Section 2, the related works and PCB-based methods are briefly reviewed. Our algorithm is presented in Section 3. Section 4 discusses the experimental results. Finally, the conclusions are presented in Section 5.

## 2 Related works

The VQ-based methods for image compression are widely used, according to the methods of codebooks training, there are two kinds of codebooks, CCB is the codebook trained for all input images, and PCB [12] is the codebook trained specially for input image.

### 2.1 Private codebook scheme

In the standard PCB method, the training image is divided into a set of nonoverlapped image blocks of  $n \times n$  pixels, and the LGB algorithm is implemented on these blocks to generate a PCB. The difference between this PCB and CCB is that PCB is trained by the input image, thus it can get a good compression on this input image, but usually performs poorly on other images, and the PCB should be transmitted along with the index table (IT) which requires a large consumption in bandwidth and storage.

The basic PCB generating steps are as follows:

*Input:* An image of  $N \times N$ .

- Step 1: Divide the input image into blocks of  $n \times n$  pixels;  
 Step 2: Initiate  $K$  codewords  $C_k^1, k = 1, 2, \dots, K$  randomly;  
 Step 3: Partition all blocks into  $K$  clusters according to the minimum distances between codewords and blocks, the optimization is as follow:

$$\arg \min_k \left\{ \|\mathbf{v} - C_k^t\|_2 \mid k = 1, 2, \dots, K \right\} \quad (1)$$

where  $\mathbf{v}$  denotes an input image block and  $C_k^t$  denotes an arbitrary codeword in iteration  $t$ ,  $\mathbf{v}$  is partitioned into the cluster of  $C_k^t$

- Step 4: Average all the blocks in each cluster as the new codeword, the equation is below:

$$C_k^t = \sum_{n=1}^{a(k)} \mathbf{v}_n / a(k) \quad (2)$$

where  $a(k)$  is the number of the blocks in cluster of  $C_k^t$ .

- Step 5: If  $\sum_{k=1}^K |C_k^t - C_k^{t-1}| \leq \varepsilon$ , terminate the training algorithm and output  $C_k^t$  as the finalcodewords. Otherwise return to *Step 2*.

The PCB method is different from other VQ-based methods by the codebook. The codewords are obtained from the input image and need to transmit along with the encoding result.

There are a lot of PCB-based methods, such as the widely used self-organizing maps (SOM) [33, 34] and learning vector quantization (LVQ) [11]. Moreover, gene algorithm is also applied to generate the PCB and artificial bee colony (ABC) algorithm as well as particle swarm optimization algorithm are also employed in generating PCB [9, 10, 28]. Paper [32] gives a series of methods that combine principal component analysis (PCA) [6, 14] and evolutionary algorithm (EA) to generate the PCB. An enhanced initialization method to train PCB is also given in [13]. The ideas of these methods are different, but the input image is

employed to generate new codewords and the codebook is transmitted along with the index table in all these methods, thus they are all PCB-based methods.

## 2.2 The drawbacks

In PCB, the improvement of performance on distortion comes from the codebook trained by the input image, in other words, the information of the input image is used to improve the distortion of the PCB. However, in the decoding phase, the PCB must be transmitted along with the IT. Considering that transmitting and storing such a codebook is a large consumption in bandwidth and storage, thus how to decrease this consumption is meaningful to decrease the BR while holding the peak signal to noise ratio (PSNR).

Assume a given grayscale image of  $M \times N$  pixels is divided into a sequence of nonoverlapped image blocks of  $n \times n$  pixels. In the encoding procedure, each image block is encoded by a codebook of  $K$  codewords, the collection of indices of all image blocks are called IT. The BR of CCB can be calculated as follow [12]:

$$BR_{CCB} = \frac{1}{n \times n} \log_2 K \quad (3)$$

Compared with the CCB method, the calculation of BR in PCB method is more complicated due to the bandwidth consumption of transmitting PCB. It equals the sum of BR in CCB method and the bandwidth of transmitting PCB, and the equation is given as follow [12]:

$$BR_{PCB} = BR_{CCB} + \frac{K \times n \times n \times 8}{M \times N} \quad (4)$$

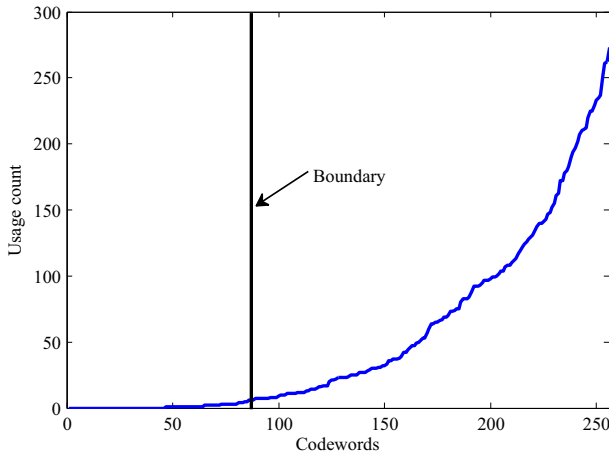
PCB can improve the performance of distortion by taking more bandwidth consumption. However, the usage count of each CCB codeword is quite different and some codewords may be rarely used. A sorted statistical result on the usage count of each CCB codeword on testing image of Lena is given in Fig. 1, and it is obvious that not each codeword is used in the encoding procedure and the counts of some codewords are too low which can be considered as inactive codewords.

Though PCB method can improve the performance of distortion, considering the resources it consumes, in fact the performance of distortion is improved at the expense of a higher BR, which is still inefficient. To generate a more efficient codebook that can obtain a better performance of distortion at a little higher BR, the general codebook design (GCB) algorithm is proposed in Section 3.

## 3 The proposed algorithm

### 3.1 The motivation

The algorithm proposed in this paper is called general codebook (GCB), and it is a more general version codebook that extends PCB and CCB. For most VQ-based methods, the distortion and BR are two important compression criteria to evaluate the performance of the designed codebook [1, 15], and either PCB or CCB can only achieve a good result on one of these two criteria. Therefore, our GCB algorithm aims at decreasing the BR and improving the distortion at the same time. Considering that some codewords in the initial codebook are



**Fig. 1** The sorted usage counts of all codewords, the CCB of 256 codewords is applied on test image of Lena, the points on the left of the boundary are inactive codewords in CCB and the points on the right of the boundary are active codewords in CCB

inefficient for improving the distortion as introduced in Section 2, a new GCB scheme (to inherit the advantages of PCB and CCB) is given, where the codewords should be easy to generate and efficient to represent the input image. The GCB scheme keeps the active codewords of CCB and also updates new codewords of input image. Figure 1 in Section 2 shows that not each codeword needs to be replaced by new generated codeword. In most situations, transmitting each codeword is a waste of BR. If we choose properly, only replacing a few codewords can improve the distortion efficiently, it is a trade-off between BR and distortion. Then, there are two problems remain to be solved: (1) how to generate the new codewords to replace the old ones; (2) which codewords are chosen to be replaced.

**3.2 The proposed GCB algorithm**

Suppose that CCB has already been obtained by LGB algorithm, which is used as the initial codebook in our algorithm, it should be specified that CCB is trained by 12 widely used images in Fig. 5. The input image is used to train the initial codebook with our GCB algorithm. This training image is divided into  $M \times N/(n \times n)$  nonoverlapped image blocks, each block can be viewed as a  $n \times n$  dimensional input vector. The distances between an input vector  $x$  and  $K$  codewords of codebook are calculated, then  $x$  is classified into the class of its nearest codeword. The PSNR of each block is calculated by Eq. (5) and Eq. (6).

$$PSNR = 10 \times \lg \left( \frac{(2^n - 1)^2}{MSE} \right) \tag{5}$$

$$MSE = \frac{1}{n \times n} \sum_{i=1}^n \sum_{j=1}^n \|x(i, j) - c(i, j)\|^2 \tag{6}$$

In Eq. (6),  $x$  denotes an image block,  $c$  denotes the nearest codeword of  $x$ , and  $n$  is the width and height of an image block  $x$ .  $(i, j)$  represents the location of pixel on the image.

As we mentioned above, we solve the problem (1) (how to generate the new codewords to replace the old ones) first. The motivation of our algorithm is to obtain a more general version of codebook than PCB and CCB, which is supposed to achieve the best balance between BR and distortion. For this purpose, the new codewords should be generated from the blocks encoded poorly by initial codewords, and the CCB is usually chosen as the initial codebook. On this basis, all these blocks are firstly divided into two parts by a threshold  $Th$  of PSNR according to the initial codebook. The blocks with higher PSNR are put into the good part, which is called as the set  $S_g$ , and these blocks are not processed further in the following procedure. The blocks with lower PSNR are partitioned into the poor part which is called as the set  $S_p$ , and these blocks obviously have a bad influence on the performance of distortion, which means the initial codebook encodes these blocks poorly. For the purpose of generating the new codewords,  $P$  codewords are then trained from these blocks in  $S_p$  as given in Eq. (7).

$$P = \arg \min_{j=1,2,\dots,K} \left| \frac{\sum_{i=1}^j cn_i}{\sum_{i=1}^K cn_i} - pcount \right| \quad (7)$$

In Eq. (7),  $cn_i$  denotes the usage count of  $i_{th}$  codeword and  $pcount$  is a threshold to decide how many codewords are regarded as inactive.

Then, we solve problem (2) (which codewords are chosen to be replaced). Since it has been given that some codewords in the initial codebook are inactive, we give the Eq. (7) to decide which one is needed to be replaced. In this equation, how to choose  $pcount$  can decide the  $P$ , 5% is an empirical constant to decide the total number of blocks that encoded by the codewords to be replaced. At last a new codebook is generated by replacing the  $P$  codewords mostly inactive. The algorithm is shown below.

- Step 1: Divide the training image into  $M \times N/(n \times n)$  nonoverlapped image blocks.
- Step 2: Copy the CCB as initial codebook.
- Step 3: Set a PSNR threshold  $Th$ .
- Step 4: Process each image block  $x$  by the following substeps:
  - 1) Find the closest codeword  $idx-th$  in codebook for  $x$  and calculate the PSNR of  $x$ .
  - 2) If the PSNR is higher than the threshold  $Th$ , increase the usage count of the  $idx-th$  codeword by one. Classify  $x$  as a member of  $idx-th$  group, put  $x$  into set  $S_g$ ; otherwise put  $x$  into set  $S_p$  immediately.
- Step 5: *Step 5:* Sort all the codewords in an ascending order of usage count and denote the usage count of each codeword as  $cn_i$   $i = 1, 2, \dots, K$ .
- Step 6: Set  $i = 1$ , calculate  $P$  by Eq. (7).
- Step 7: Cluster the blocks of set  $S_p$  into  $P$  classes by LGB iterations.
- Step 8: Replace  $P$  mostly inactive codewords in codebook by  $P$  newly generated codewords in *Step 7*.
- Step 9: Output the codebook as GCB.

Compared to PCB, our GCB has three advantages: (1) the image blocks are divided into two parts by  $Th$  of PSNR, where the  $S_p$  contains the blocks that have bad distortion encoded by CCB. Only this part is used to update the codewords in the initial CCB. In contrast, all blocks are used in the iterations to update the codewords in PCB. In fact, the key to improve the

performance of the codebook is to encode the blocks with poor distortion. In PCB algorithm, using the blocks with good performances to update the codewords may not perform well, since the poor blocks may still be poorly encoded by the codewords updated in PCB. However, our GCB avoids this disadvantage by only updating the codewords with poorly encoded blocks in  $S_p$ . (2) In GCB, only the most inactive codewords are updated, it means only a small number of codewords are transmitted, thus the bandwidth can be significantly saved. In PCB, all codewords are updated, thus all the codebook needs to be transmitted. In GCB only a few codewords are updated by the blocks in  $S_p$  which means only these codewords are transmitted, the bandwidth is saved and only the blocks in  $S_p$  are improved while the blocks in  $S_g$  are not influenced at all. (3) Compared to PCB, our GCB gets the new updated codewords by iterating the codewords in  $S_p$ , the computational cost is much lower than PCB because the number of image blocks in  $S_p$  is small. The performance analysis in Section 3.3 is given to show the reason why only the poor blocks are improved.

### 3.3 The performance analysis

Compared with the initial CCB, only  $P$  codewords are updated and transmitted for saving the bandwidth in GCB method, therefore the BR of this algorithm is between CCB and PCB. In fact, the experiments in the next section demonstrate that the number of most inactive codewords indexed by 5% blocks is very small, it means only a few codewords are transmitted in GCB, so the BR of the proposed GCB is only a little higher than CCB but much lower than PCB. Moreover, the improved performance of GCB compared with CCB mainly comes from the  $P$  updated codewords. Considering that the blocks in set  $S_p$  have lower PSNR, better codewords are needed to improve their PSNR. There are two classes of codewords in GCB, the old codewords from CCB, and the new codewords trained by image blocks in  $S_p$ . The input blocks can be encoded by any codewords in both classes. If the blocks in  $S_g$  are encoded by old codewords, the performance of these blocks will stay the same. If they are encoded by the new codewords, the performance should be better than old codewords otherwise they must be encoded by the old codewords. It is just opposite for the blocks in  $S_p$ , if these blocks are encoded by the new codewords, they will have a better performance than using the old codewords. No matter the blocks are in  $S_g$  or  $S_p$ , they will be encoded better than using CCB.

The formula to calculate BR of GCB is as follows:

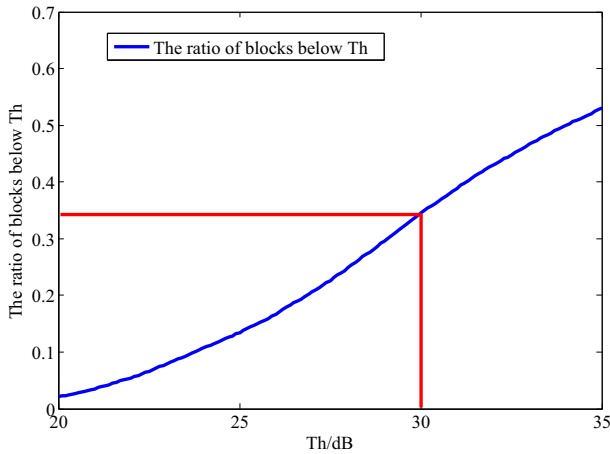
$$BR_{GCB} = BR_{CCB} + \frac{P \times n \times n \times 8 + \log_2 K}{M \times N} \quad (8)$$

Compared to BR of PCB defined in Eq. (4), since  $P$  is much smaller than  $K$  and the bandwidth consumption of the flags for updated codewords is negligible, thus the BR of GCB is much smaller than PCB.

### 3.4 The selection of thresholds of $Th$ and $pcount$

There are two important thresholds in GCB algorithm, the first one is the PSNR threshold  $Th$  that determines how an input block is encoded. Basically, an empirical threshold is good enough,  $Th = 30$  dB is usually used.

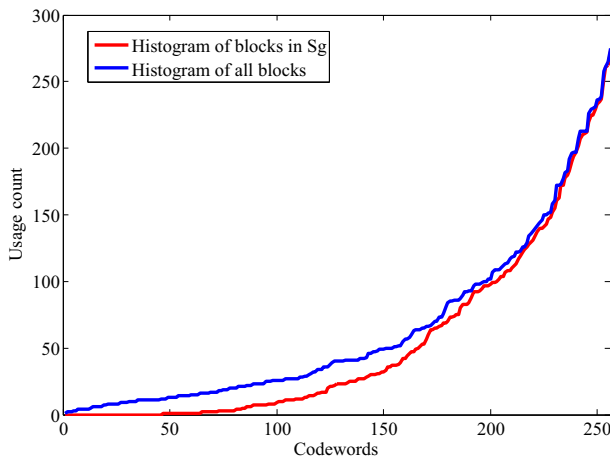
Figure 2 below gives an example of the CCB with 256 codewords on testing image of Lena, which describes the relationship between  $Th$  and the ratio of the blocks in  $S_p$  among the total blocks. When  $Th$  equals 30 dB, 34.3% of the blocks are assigned to  $S_p$ .



**Fig. 2** Relationship between  $Th$  and the ratio of  $S_p$  in total blocks. The CCB of 256 codewords is tested on Lena, the block is  $4 \times 4$  pixels

The second threshold is  $pcount$ .  $P$  can be computed by Eq. (7) with  $pcount$  fixed. In GCB algorithm,  $P$  determines the number of codewords in the initial CCB that needs to be replaced, which is also the number of codewords generated by the blocks in  $S_p$ . The usage count of codeword in initial CCB indexed by the blocks in  $S_g$  is counted in GCB algorithm, the codewords that have less usage counts will be abandoned. In order to determine which and how many codewords are abandoned, the GCB algorithm sorts the codewords in an ascending order by the usage count, then it counts the number of codewords one by one from the beginning of the order until the indexed counts of these codewords is more than 5% of total blocks in  $S_g$ . Finally, the number of codewords that we count is the value of threshold  $P$ . In this scheme, choosing  $pcount$  as 5% to determine  $P$  is relatively effective.

Figure 3 shows usage counts histogram of blocks in  $S_g$  and histogram of all blocks, where all the blocks are encoded by CCB of 256 codewords on testing image of Lena. As the curve shows in Fig. 3, most codewords are used rarely and some are even unused.



**Fig. 3** Histogram of blocks in  $S_g$  and histogram of all blocks are shown, the CCB of 256 codewords is tested on Lena, the block is  $4 \times 4$  pixels



In fact, the idea of our algorithm is based on the property that most image blocks for compression can be encoded well by the codewords in CCB. Since the blocks in  $S_g$  have high PSNR, no further processing is needed, but the poorly encoded blocks in  $S_p$  are the key to be improved.

If the parameter  $P$  equals 0, no codeword in CCB is replaced, the codebook obtained by our GCB degenerates into CCB. If the threshold  $P$  equals  $K$ , all the codewords are replaced by the updated codewords. It is a kind of PCB. The three versions of codebook can be summarized as follows:

- 1) CCB, no codeword is transmitted, low PSNR;
- 2) PCB, all codewords are transmitted, high PSNR;
- 3) GCB, only a few codewords are transmitted, high PSNR.

Our GCB is in the middle of CCB and PCB, only a few codewords in GCB are transmitted, and the performance of distortion is close to PCB. Therefore, GCB is a more general version of codebook and either CCB or PCB is a special case of GCB.

### 3.5 The flowchart

Basically, there are two parts in the algorithm, the first part is to separate the blocks into  $S_g$  and  $S_p$ . The second part is to generate the GCB by  $S_g$  and  $S_p$ .

In the first part, the input image  $I$  is split into image blocks. The CCB encoding process is executed first to choose the codewords which can stay unchanged in the GCB, thus only a few codewords need to be updated. Compared to PCB, this procedure makes our algorithm improve the compression performance with lower BR costs.

The next part is to generate new codewords from the image blocks in  $S_p$  with poor performances of distortion by using CCB. There are a lot of practical algorithms for clustering, in this paper LBG algorithm is employed. The final codebook is composed by these two parts: the unchanged codewords in CCB and new codewords generated by LBG algorithm from  $S_p$ .

### 3.6 The example of GCB

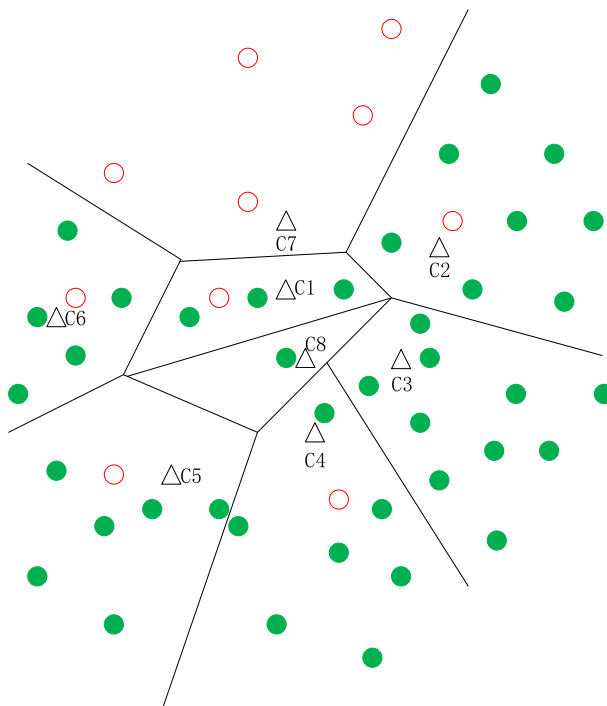
In order to make the whole GCB algorithm more clearly, Fig. 4 depicts an example of the image block clustering process. 50 image blocks and a codebook of 8 codewords are employed to show the clustering process in our algorithm. The blocks are shown as solid and hollow points in Fig. 4, and the codewords are denoted as  $c_1, c_2, \dots, c_8$ , respectively. In the first part of the algorithm, all these 50 blocks are encoded by CCB. They are clustered into 8 groups by minimal distance criterion. A threshold  $Th$  of PSNR is set by empirical knowledge to separate these blocks into two sets of  $S_p$  and  $S_g$ .

Equation (6) can show the similarity between the block and its nearest codeword. The higher PSNR is, the more similar the two objects are. The solid points in Fig. 4 represent the blocks in  $S_g$  that have a higher PSNR, and these points stay unchanged in the next part of algorithm. As discussed before, these points with better PSNR are not the key factors to improve the performance. The hollow points in Fig. 4 represent the blocks in  $S_p$ . These blocks are encoded poorly by the codewords in CCB. A good method to solve this problem is to generate new codewords from these blocks and this is what our proposed GCB does in the second part of the flowchart. Considering that the blocks in  $S_p$  give no help to count the

codeword usage count, the algorithm only counts the blocks in  $S_g$  to get the codewords usage counts. As Fig. 4 shows, there are 40 solid blocks and 10 hollow blocks. The 40 blocks are distributed in 7 groups. Note that group c7 has no solid block in it, and the other groups are sorted in a descending order by the number of solid blocks they have. The sorted sequence is c3, c2, c4, c5, c6, c1 and c8, and they have 10, 8, 7, 6, 5, 3, 1 solid blocks respectively. According to *Step 5* of our GCB, the most inactive codewords are the codewords indexed by no more than 2 blocks in total. Therefore, group c7 and c8 are chosen as the codewords to be replaced and the threshold  $P$  is determined as 2 in this example.

As the threshold  $P$  decides that two codewords need to be replaced, thus the second part of algorithm generates two codewords from the blocks in  $S_p$ . The centroid updating process is executed to improve the distortion of the codebook, and LBG algorithm is employed in this procedure, where  $P$  codewords are generated from the blocks in  $S_p$ . In this example  $P$  equals 2, which means two corresponding codewords c7 and c8 are replaced by the newly generated codewords. As mentioned earlier, these new codewords can improve the PSNR for image blocks in both sets  $S_g$  and  $S_p$ . Finally, the new GCB is obtained by our algorithm.

The GCB codebook design procedure is introduced above. For a VQ process, the next two phases are image encoding and image decoding. In the image encoding procedure, the nearest codeword for each block is searched and its index is transmitted. All of these indices construct the IT. In the encoding procedure of CCB, only IT consumes the bandwidth, and in the encoding procedure of PCB another overhead part of bandwidth is the codewords in PCB, this bandwidth consumption can be effectively reduced by our GCB algorithm. In the example of



**Fig. 4** An example of clustering process by our GCB algorithm

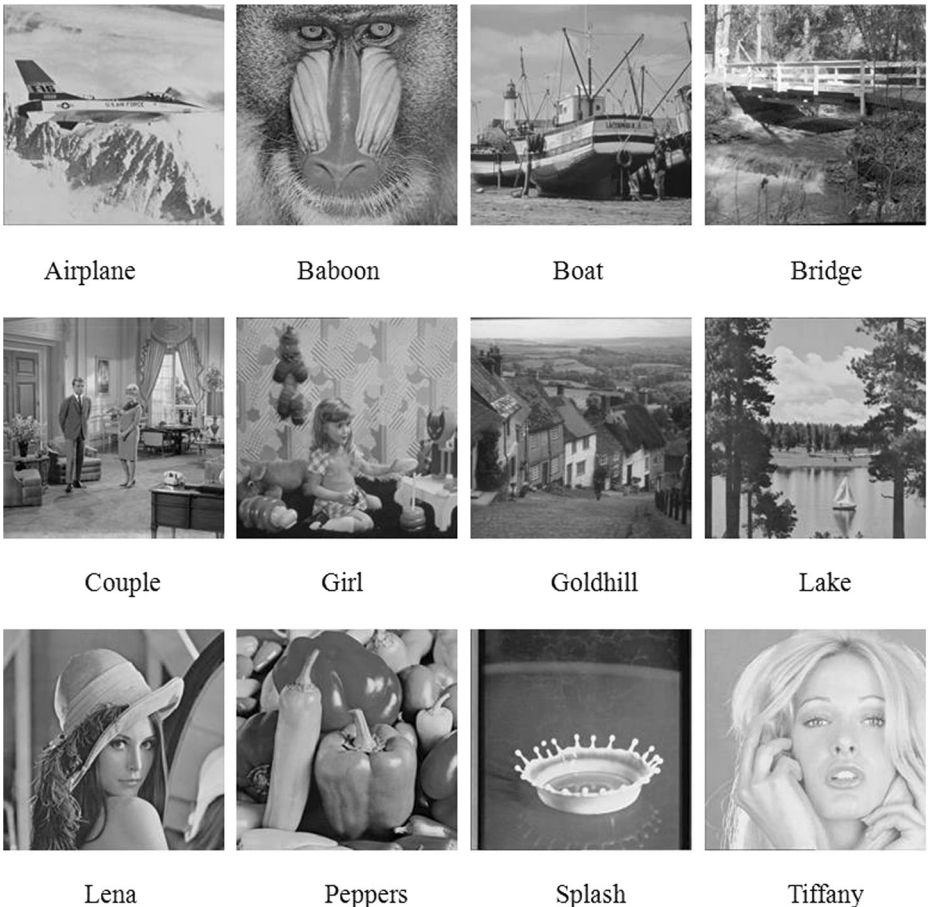
Fig. 4, only two codewords and the flags of the codewords that are replaced need to be transmitted along with IT.

In the decoding phase, to reconstruct the input image, GCB and IT are needed. Firstly, we reconstruct GCB with the transmitted codewords and their replacing flags. The decoding side has an initial CCB in advance, the codewords that are replaced can be determined by the flags, GCB is obtained by replacing these codewords. When GCB is obtained on decoder side, the decoding procedure can be then executed.

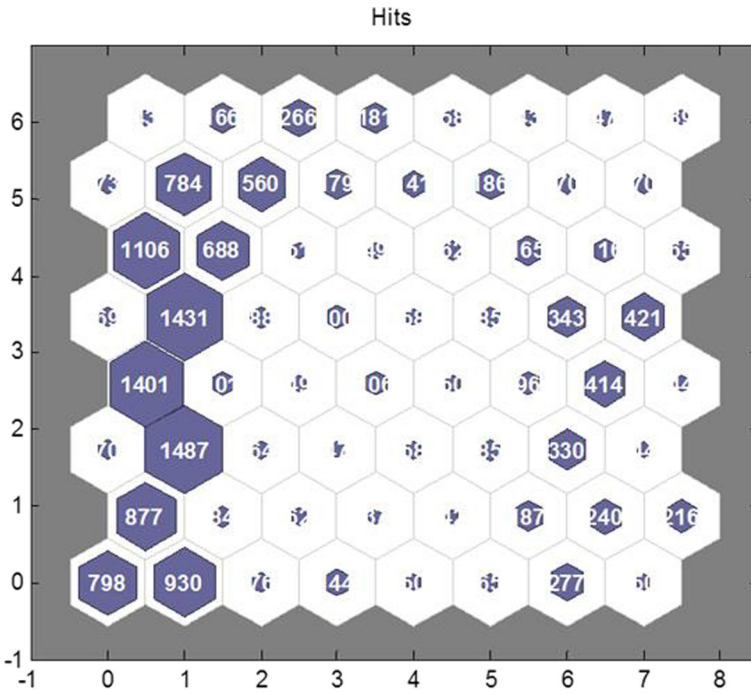
## 4 Experiment

### 4.1 Setup of simulation

The experiments are performed on HP personal computer with a CPU of core E7500 @ 2.93GHz and 8G RAM. In the experiments, LBG algorithm is employed to generate the initial CCB of different sizes. Twelve input images of Airplane, Baboon, Boat, Bridge, Couple, Girl, Goldhill, Lake, Lena, Peppers, Splash, Tiffany



**Fig. 5** Twelve images used in the experiment



**Fig. 6** The neurons hit by the input vectors in SOM networks. There are 256 neurons (codewords) and input vectors are blocks of Lena

Goldhill, Lake, Lena, Pepper, Splash, Tiffany of  $512 \times 512$  pixels are shown in Fig. 5. They are used as test images to compare the performances of our GCB algorithm and four codebook generating algorithms, which are Hu’s PCB, differential evolution algorithm (DEA), SOM algorithm and CCB. Hu’s PCB, DEA and SOM are 3 PCB-based algorithms. The termination threshold of LBG algorithm is set to  $10^{-3}$ . The CCB, SOM algorithm and Hu’s PCB have 6 kinds of sizes as  $K = 128, 256, 512, 1024, 2048, 4096$ . In GCB algorithm,  $Th$  is fixed as 30 dB,  $pcount$  and  $P$  are chosen according to the experiment demands, which will be given in the comparison. The CCB is trained by the 12 images directly, the encoder side and decoder side can obtain the CCB offline which means no transmission is needed.

As mentioned before, each test image is divided into nonoverlapped image blocks of  $4 \times 4$  pixels, so there are 16,384 blocks in total for each test image. There are two mainly used criteria to evaluate the performances of above methods in our experiments, where PSNR is employed to measure the distortion of reconstructed images and BR is also given to compare the proposed GCB method with four other algorithms.

**Table 1** The initial parameters of DEA

Initial parameters of DEA	
Mutation rate	0.5
Crossover rate	0.9
Number of iteration	100
Dimension of vector	16
Population size	128
Fitness function	MSE

**Table 2** The distortion of PSNR (dB) of twelve test images achieved by GCB

BR	128 0.461bpp	256 0.549bpp	512 0.641bpp	1024 0.750bpp	2048 0.888bpp	4096 1.150bpp
Airplane	30.767	31.884	33.000	34.248	34.592	36.153
Baboon	24.167	24.841	25.582	26.371	26.596	27.524
Boat	28.674	29.601	30.579	31.608	31.920	33.231
Bridge	24.451	25.159	25.835	26.545	26.804	28.009
Couple	27.293	28.225	29.155	30.151	30.472	31.731
Girl	31.680	32.730	33.773	34.772	35.186	36.365
Goldhill	30.312	31.182	32.072	32.928	33.304	34.601
Lake	28.495	29.325	30.172	31.035	31.314	32.415
Lena	30.815	31.796	32.744	33.719	35.399	36.240
Pepper	31.057	32.197	33.309	34.451	34.877	35.396
Splash	33.565	34.696	35.810	36.972	37.384	38.221
Tiffany	32.383	33.278	33.727	34.494	35.200	35.204

## 4.2 A brief introduction of compared algorithms

In the experiments, four algorithms are employed to verify the performance of our GCB. Hu's PCB is implemented as [30] described. CCB is obtained by training the 12 test images in Fig. 5 with LBG algorithm.

A self-organizing map (SOM) is a type of artificial neural network that is trained using unsupervised learning to produce a low-dimensional, discretized representation of the input space of the training samples. Since the codebook obtained by SOM is special for the input image, SOM is a PCB-based algorithm.

The Fig. 6 is the SOM network, a neuron in competitive layer can be looked as a codeword, and the final weight vector of competitive layer is a codebook. Whenever a neuron is activated, the hit counter adds 1, most neurons are barely hit and only a few of them are frequently used. This also supports our point in Section 2: some codewords are so barely used that they can be considered as inactive.

The differential evolution algorithm (DEA) is a kind of gene algorithm [2, 8, 18]. In this algorithm, every codeword is looked as an individual in a population. In each iteration, every individual is updated by mutation and crossover, a better individual is selected by the fitness function. The parameter is given in Table 1. DEA is also a PCB-based algorithm.

**Table 3** Comparative results of PSNR with Hu's PCB, SOM algorithm at the BR of 1.125bpp

	GCB	Hu's PCB	SOM
Airplane(PSNR)	35.991	34.379	33.793
Baboon(PSNR)	27.427	26.786	26.353
Boat(PSNR)	33.116	32.297	31.494
Bridge(PSNR)	27.622	26.969	26.530
Couple(PSNR)	31.620	30.781	29.946
Girl(PSNR)	36.121	35.991	34.970
Goldhill(PSNR)	34.341	33.539	32.989
Lake(PSNR)	32.325	31.418	30.848
Lena(PSNR)	35.883	33.719	34.159
Pepper(PSNR)	35.043	34.770	34.159
Splash(PSNR)	38.191	37.995	37.029
Tiffany(PSNR)	36.224	35.962	35.014

**Table 4** Comparative results of PSNR with Hu's PCB, SOM algorithm at the BR of 1.688bpp

	GCB	Hu's PCB	SOM
Airplane(PSNR)	38.198	35.806	35.051
Baboon(PSNR)	29.012	27.692	27.320
Boat(PSNR)	35.220	33.354	32.597
Bridge(PSNR)	29.384	27.933	27.434
Couple(PSNR)	33.647	31.713	31.228
Girl(PSNR)	38.434	37.167	36.400
Goldhill(PSNR)	36.051	34.646	34.092
Lake(PSNR)	34.086	32.299	31.740
Lena(PSNR)	36.369	35.399	35.573
Pepper(PSNR)	36.385	35.539	35.673
Splash(PSNR)	39.124	38.904	38.718
Tiffany(PSNR)	36.321	36.202	36.591

### 4.3 Comparison results

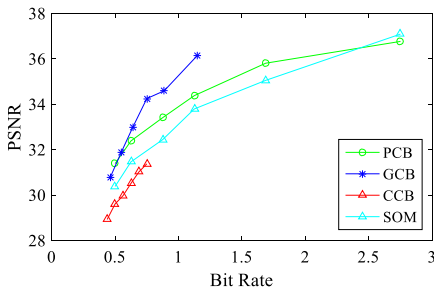
Table 2 lists the results of the distortion on PSNR and BR by using GCB of different sizes. It is shown that the distortion of the reconstructed image decreases as the codebook size increases. The corresponding updated codewords counts  $P$  are 48, 100, 128, 256, 512 and 820, and the corresponding required BR of GCB are 0.4609, 0.5488, 0.6406, 0.750, 0.8875, 1.150bpp respectively.

Tables 3 and 4 list the results of distortion of the compressed images by using Hu's PCB, SOM algorithm and GCB. All results of three methods are compared at the same BR of 1.125bpp and 1.688bpp respectively. To get the same BR of PCB-based methods, the GCB is tested on the codebook of 2048 and 4096 codewords while the codebooks of PCB-based algorithms are 1024 and 2048 correspondingly,  $P$  is tuned to get the BR of 1.125bpp on codebook of 2048 and 1.688bpp on codebook of 4096. It is shown that the performance of GCB is best at the same BR.

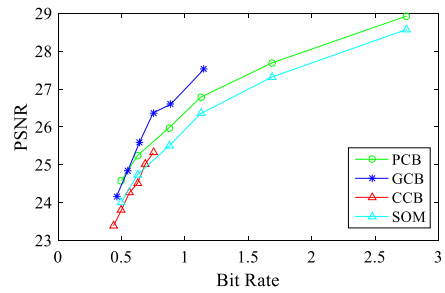
We also test DEA against our GCB, the DEA is tested on codebook of 128 while GCB is tested on codebook of 256. The results are listed in Table 5. The BR of our GCB is 0.46 bpp while the BR of DEA is 0.50 bpp. The results show that our GCB performs better even at a lower BR. Considering the computational complexity, the bigger size of codebook is not tested and the codebook of 128 is an example to verify the better performance of GCB.

**Table 5** The comparative result of DEA and our GCB on the codebook size of 128

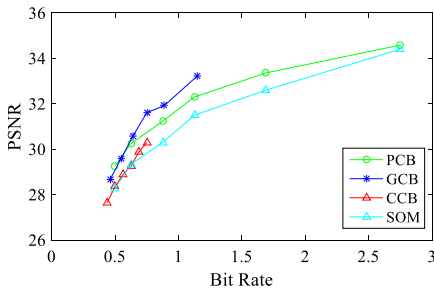
	DEA(0.50bpp)	GCB(0.46bpp)
Airplane(PSNR)	30.168	30.767
Baboon(PSNR)	24.107	24.167
Boat(PSNR)	28.012	28.674
Bridge(PSNR)	24.380	24.451
Couple(PSNR)	27.121	27.292
Girl(PSNR)	30.974	31.680
Goldhill(PSNR)	29.942	30.311
Lake(PSNR)	28.062	28.495
Lena(PSNR)	30.280	30.816
Pepper(PSNR)	30.278	31.057
Splash(PSNR)	31.530	33.565
Tiffany(PSNR)	31.778	32.383



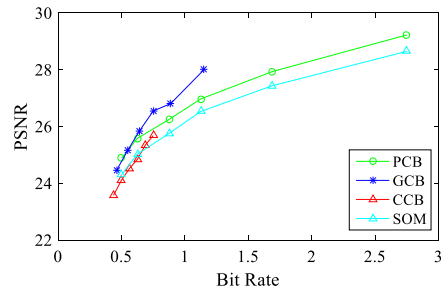
Airplane



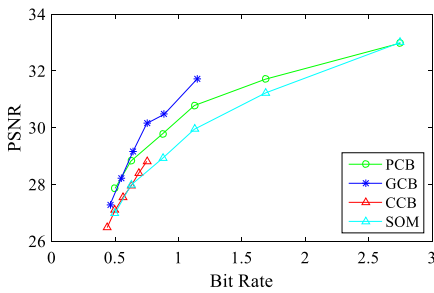
Baboon



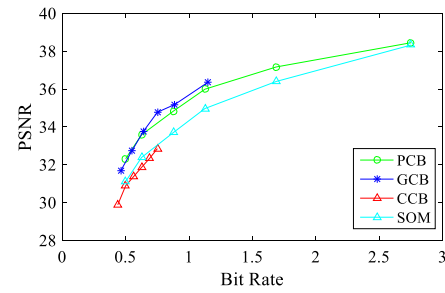
Boat



Bridge



Couple

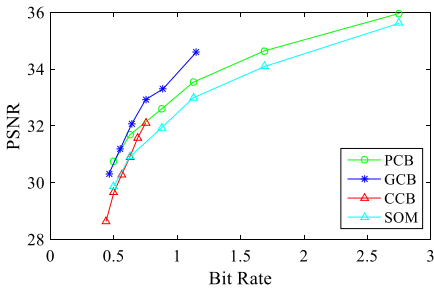


Girl

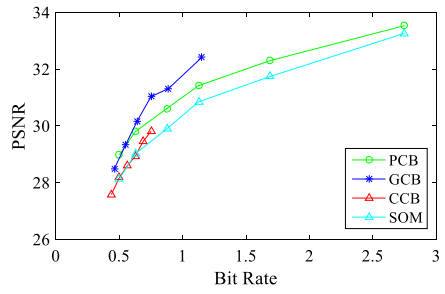
**Fig. 7** Comparative results of VQ on twelve test images with four algorithms of CCB, Hu’s PCB, SOM and GCB. In each sub-figure, the four algorithms are tested in different sizes of 128, 256, 512, 1024, 2048 and 4096. Each point means a size of codebook

To verify the effectiveness of our GCB algorithm, comparative results of rate-distortion of Hu’s PCB, SOM algorithm, CCB and our GCB are listed in Fig. 7. The bandwidth of the CCB is not included in calculating the BR, therefore the BR of CCB is lower than other algorithms. The advantage of our GCB algorithm is that only the necessary  $P$  codewords are transmitted, which make it have a lower BR than 3 kinds of PCB-based methods and the PSNR of GCB and PCB-based methods are closed.

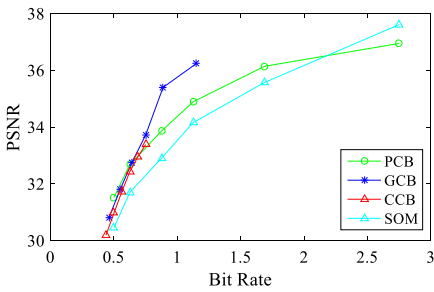
PCB and CCB can be viewed as the two special cases of our GCB algorithm. When  $P$  equals  $K$ , GCB becomes a PCB. When  $P$  equals 0, GCB becomes a CCB. It is shown that the



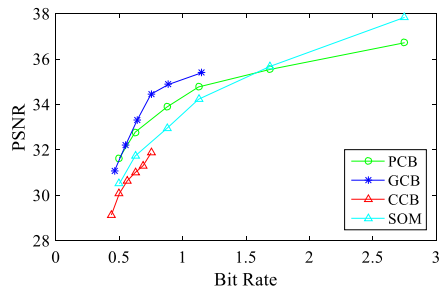
Goldhill



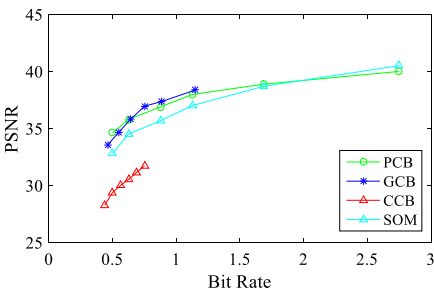
Lake



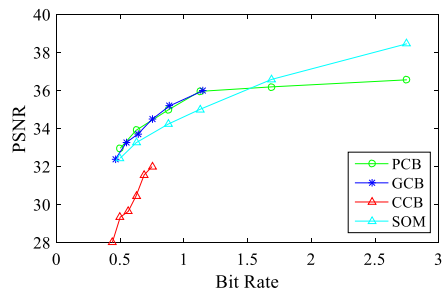
Lena



Peppers



Splash



Tiffany

Fig. 7 continued.

curve of our GCB algorithm is the best among all rate-distortion curves in Fig. 7. It is also shown that the curve of CCB performs worst among four codebooks. When BR is lower than 0.6 bpp, PCB and GCB achieve almost the same performances, but they still perform better than CCB. When BR is larger than 0.6 bpp, GCB provides the best performance among these three comparative schemes.

The computational cost is also an important aspect in this research. Many methods focus on reducing the computational cost, such as paper [27] puts the computational cost as a significant concern in the algorithm designing. To verify the good performance of our GCB, the computational cost is collected. As a VQ-based compression method, there are three stages: codebook generating, encoding and decoding.

The cost of encoding is same for all VQ-based methods since this stage is searching the closest codeword to the block. The cost of decoding is also same for all VQ-based methods



**Table 6** The computational cost of five methods tested on Lena

	CCB	GCB	Hu's PCB	SOM	DEA
Codebook generating	0	$\frac{N^2K(\theta_1t+1)}{16}$	$\frac{N^2K(1+t+\theta_2t)}{16}$	$\frac{N^2Kt}{16}$	–
Encoding	$N^2K/16$	$N^2K/16$	$N^2K/16$	$N^2K/16$	$N^2K/16$
Decoding	$N^2/16$	$N^2/16$	$N^2/16$	$N^2/16$	$N^2/16$
Total time(s)	0.986	8.622	19.976	14.026	3245.2

since this stage is a look-up table operation. The difference among common codebook (CCB), Hu's private codebook (PCB), self-organizing maps (SOM), differential evolution algorithm (DEA) and general codebook (GCB) is the codebook generating stage.

For CCB, the codebook is pre-generated, thus, its cost is 0. For DEA, the computational cost is still a research topic [2, 8, 18]. The cost of DEA is usually very large, we only give the real computational time in our experiment. The remaining three methods: GCB, Hu's PCB and SOM are compared by using computational cost and real computational time (Table 6).

The experiment is conducted on Lena with a codebook of 256 codewords, the experiment platform is a computer with 16GB RAM and Intel core i7–6700 CPU@3.40 GHz. These five methods are tested on Matlab 2015b.

The size of test image is  $N \times N$ , The max iterator in codebook generating is  $t$ , the block size is  $4 \times 4$ , and the number of codeword is  $K$ . The computational cost of SOM in codebook generating stage is  $N^2Kt/16$ . The computational cost of Hu's PCB is  $(N^2Kt + N^2K + \theta_2N^2Kt)/16$  where  $\theta_2 \in (0, 1]$ . For a PCB is first generated (computational cost is  $N^2Kt/16$ ), the blocks are encoded by this PCB (computational cost is  $N^2K/16$ ) and then a few codewords are updated (computational cost is  $\theta_2N^2Kt/16$ ). The computational cost of GCB is the smallest among Hu' PCB, GCB and SOM. The initiate codebook is CCB, the blocks are first encoded by CCB (the computational cost is  $N^2K/16$ ), then a few codewords are trained from the blocks in  $S_p$  (the blocks poorly encoded determined by  $Th$ ), the computational cost is  $\theta_1N^2Kt/16$ , where  $\theta_1 \in (0, 1]$ . The cost of generating codewords is usually large, but only a few codewords are trained and the number of blocks for training is small, thus our GCB consumes the shortest time in SOM, Hu's PCB and GCB methods.

Our GCB is compared to two state-of-the-art algorithms. The embedded zerotree wavelet algorithm (EZW) comes from the paper [29] which introduces an encode algorithm based on wavelet transforme. The enhanced side match vector quantization (ESMVQ) is an effective low-bit rate coding algorithm [21]. In ESMVQ algorithm, the concept of complementary state codebook (CSC) is proposed to improve the reconstructed image quality. By using CSC, ESMVQ can achieve almost the same encoding visual quality as using the conventional VQ. These three algorithms are compared at BR close to 0.5 bpp, our GCB ourperforms on most test images (Table 7).

**Table 7** The comparison results of PSNR for EZW, ESMVQ and GCB

	Lena	Baboon	Peppers	Airplane	Girl	Goldhill	Splash
EZW	<b>32.89</b>	23.95	<b>32.67</b>	30.58	31.25	30.66	33.08
ESMVQ	31.76	24.49	30.54	31.75	31.59	30.34	29.62
GCB	31.79	<b>24.84</b>	32.19	<b>31.88</b>	<b>32.79</b>	<b>31.18</b>	<b>34.69</b>

## 5 Conclusion

In this paper, a scheme to generate an efficient GCB codebook is proposed. The generated codebook decreases the BR of the PCB-based methods by only transmitting a small number of updated codewords and the replacing flags. The blocks are firstly encoded by CCB, and then they are divided into two sets by their PSNR. The blocks well encoded are put into set  $S_g$  and the blocks poorly encoded are put into set  $S_p$ . The blocks in  $S_p$  are trained by LBG algorithm to generate  $P$  codewords afterwards, where the threshold  $P$  is determined by Eq. (5). Then the  $P$  mostly inactive codewords are replaced by the  $P$  generated codewords, finally GCB is obtained. The proposed GCB is tested on 12 test images with various kinds of PCB-based algorithms and CCB. Compared to these algorithms, the curve of our GCB is higher, so the distortion of GCB is the best at the same BR.

Experimental results demonstrate that by updating some inactive codewords in the CCB with codewords generated by the LBG algorithm, GCB saves the BR and achieves a better performance than various kinds of PCB-based algorithms and CCB at the same BR.

**Acknowledgements** This work is supported in part by the Open Project Program of the State Key Lab of Novel Software Technology (Grant No. KFKT2016B14), Nanjing University, the Priority Academic Program Development of Jiangsu Higher Education Institutions (PAPD), the Open Research Fund of Key Laboratory of Spectral Imaging Technology, Chinese Academy of Sciences (Grant No. LST201606D) and the Industrial Program of Zhejiang Province (Grant No. 2016C31G4180003).

## References

1. Ayoobkhan MUA, Chikkannan E, Ramakrishnan K (2017) Lossy Image Compression Based on Prediction Error and Vector Quantisation. EURASIP Journal on Image and Video Processing 5
2. Chang CC, Lin PY (2007) Significance-preserving codebook using generic algorithm. In: Proceeding FSKD '07 proceedings of the fourth international conference on fuzzy systems and knowledge discovery, August 24 - 27, vol 03. IEEE Computer Society, Washington, DC, pp 660–664
3. Chang CC, Tai WL, Lin CC (2006) A reversible data hiding scheme based on side match vector quantization. IEEE Trans Circuits Syst Video Technol 16(10):1301–1308
4. Chen L, Liu JTZ (2013) A statistical image resolution enhancement approach for mixed-resolution video. Int J Electron 101(9):1205–1216
5. Chen Q, Kotani K, Lee F, Ohmi T (2010) Feature facial image recognition using VQ histogram in the DCT domain. Proc. SPIE 7546, second international conference on digital image processing, 75460J. p 7. <https://doi.org/10.1117/12.855647>
6. Chris D, He X (2004) K-means clustering via principal component analysis. In: Proceeding ICML '04 proceedings of the twenty-first international conference on machine learning, Banff, Alberta, Canada — July 04 - 08. ACM, New York, pp 29
7. Chuang JC, Hu YC (2011) An adaptive image authentication scheme for vector quantization compressed image. J Vis Commun Image Represent 22(5):440–449
8. Dao SD, Abhary K, Marian R (2017) An innovative framework for designing genetic algorithm structures. Expert Syst Appl 90:196–208

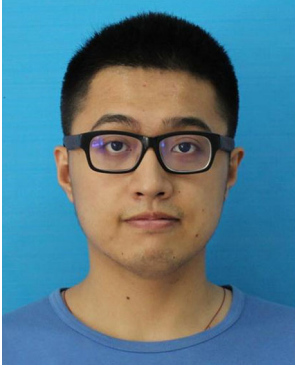
9. Feng HM, Chen CY, Ye F (2007) Evolutionary fuzzy particle swarm optimization vector quantization learning scheme in image compression. *Expert Syst Appl* 32(1):213–222
10. Fränti P (2000) Genetic algorithm with deterministic crossover for vector quantization. *Pattern Recogn Lett* 21(1):61–68
11. Hammer B et al (2014) Learning vector quantization for (dis-)similarities. *Neurocomputing* 131(7):43–51
12. Hu YC, Chen WL, Tsai PY (2015) Refined codebook for grayscale image coding based on vector quantization. *Opt Eng* 54(7):073110
13. Hu KC, Chen CC, Tsai CW, Chiang MC (2015) An enhanced initialization method for codebook generation. In: *Proc. IEEE International Conference on Consumer Electronics - Taiwan (ICCE-TW)*, 6–8 June. IEEE, Taipei, pp 92–93
14. Jolliffe IT (1986) *Principal component analysis for special types of data*. Springer, New York, pp 199–222
15. Lakshmi M, Senthilkumar J, Suresh Y (2016) Visually lossless compression for Bayer color filter array using optimized vector quantization. *Appl Soft Comput* 46(C):1030–1042
16. Li C, Liu JTZ (2013) A statistical image resolution enhancement approach for mixed-resolution video. *Int J Electron* 101(9):1205–1216
17. Linde Y, Buzo A, Gray RM (1980) An algorithm for vector quantizer design. *IEEE Trans Commun* 28(1):84–95
18. Liu J, Lampinen J (2005) A fuzzy adaptive differential evolution algorithm. *Soft Comput* 9(6):448–462
19. Lloyd S (1982) Least squares quantization in PCM. *IEEE Trans Inf Theory* 28(2):129–137
20. Lu T-C, Chang C-Y (2010) A survey of VQ codebook generation. *J Inf Hiding Multimed Signal Process* 1(3):190–203
21. Ma X, Pan Z, Hu S, Wang L (2014) Enhanced side match vector quantisation based on constructing complementary state codebook. *IET Image Process* 9(4):290–299
22. Ma T, Wang Y, Tang M, Cao J, Tian Y, Al-Dhelaan A, Al-Rodhaan M (2016) LED: a fast overlapping communities detection algorithm based on structural clustering. *Neurocomputing* 207: 488–500
23. Mohamed Uvaze Ahamed A, Eswaran C and Kannan R (2016) Lossy image compression based on vector quantization using artificial bee Colony and genetic algorithms. *International Conference on Computational Science and Technology*, Kota Kinabalu
24. Pan JS, McInnes FR, Jack MA (1995) VQ codebook design using genetic algorithms. *Electron Lett* 31(17):1418–1419
25. Pan Z, Zhang Y, Kwong S (2015) Efficient motion and disparity estimation optimization for low complexity multiview video coding. *IEEE Trans Broadcast* 61(2):166–176
26. Pan Z, Lei J, Zhang Y, Sun X, Kwong S (2016) Fast motion estimation based on content property for low-complexity H.265/HEVC encoder. *IEEE Trans Broadcast* 62(3):675–684
27. Pan Z, Jin P, Lei J, Zhang Y, Sun X, Kwong S (2016) Fast reference frame selection based on content similarity for low complexity HEVC encoder. *J Vis Commun Image Represent* 40, Part B:516–524
28. Park H, Park JI (2017) Rapid generation of the state codebook in side match vector quantization. *IEICE Trans Inf Syst* E100-D(8):1934–1937
29. Peng Z, Wang G, Jiang H, Meng S (2017) Research and improvement of ECG compression algorithm based on EZW. *Comput Methods Programs Biomed* 145:157–166
30. Song J, Choi J, Love DJ (2017) Common codebook millimeter wave beam design: designing beams for both sounding and communication with uniform planar arrays. *IEEE Trans Commun* 65(4):1859–1872
31. Tsai P (2009) Histogram-based reversible data hiding for vector quantisation-compressed images. *IET Image Process* 3(2):100–114
32. Tsai JT, Chou PY, Chou JH (2015) Performance comparisons between PCA-EA-LBG and PCA-LBG-EA approaches in VQ codebook generation for image compression. *Int J Electron* 102(11):1–21
33. Vesanto J (1999) SOM-based data visualization methods. *Intelligent Data Analysis* 3(2):111–126
34. Wang Z, Xu X (2012) Improved SOM-based high-dimensional data visualization algorithm. *Comput Inf Sci* 5(4):112–115
35. Watanabe K (2015) Vector quantization based on  $\epsilon$ -insensitive mixture models. *Neurocomputing* 165:32–37
36. Zheng WM, Lu ZM, Burkhardt H (2006) Color image retrieval schemes using index histograms based on various spatial-domain vector quantizers. *Int J Innov Comput Inf Control* 2(6):1317–1326
37. Zhou Z, Wu QMJ, Huang F, Sun X (2017) Fast and accurate near-duplicate image elimination for visual sensor networks. *Int J Distrib Sens Netw*. <https://doi.org/10.1177/1550147717694172>



**Rui Li** received the B.S. and Master degree from Xidian University, Xi'an, P.R. China, in 2012 and 2015 respectively. He is currently working towards the Ph.D. degree in School of Electronic and Information Engineering at Xi'an Jiaotong University, Xi'an, P.R. China. His research interests include vector quantization and hyper-spectral image processing.



**Zhibin Pan** received the B.S. degree in Information and Telecommunication Engineering in 1985 and the M.S. degree in Automation Science and Technology in 1988 from Xi'an Jiaotong University, P.R. China, respectively. He received the Ph.D. degree in Electrical Engineering in 2000 from Tohoku University, Japan. He is a professor in the Department of Information and Telecommunication Engineering, Xi'an Jiaotong University, P.R. China. His current research interests include image compression, multimedia security and object recognition.



**Yang Wang** received his BS degree from Xi'an Jiaotong University, Xi'an, P.R. China, in 2010. He is currently working towards the Ph.D. degree in School of Electronic and Information Engineering at Xi'an Jiaotong University, Xi'an, P.R. China. His research interests include image coding and image processing.