


# A New SMVQ-Based Reversible Data Hiding Scheme Using State-Codebook Sorting

Juan-ni Liu , Quan Zhou, Yan-lang Hu, and Jia-yuan Wei

National Key Laboratory of Science and Technology on Space Microwave,  
Xi'an Institute of Space Radio Technology, Xi'an 710100, Shaanxi, China  
liujuanni0@126.com

**Abstract.** A reversible data hiding scheme makes it possible to extract secret data and recover the cover image without any distortion. This paper presents a novel reversible data hiding scheme based on side-match vector quantization (SMVQ). If the original SMVQ index value of an image block is larger than a threshold, a technique called state-codebook sorting (SCS) is used to create two state-codebooks to re-encode the block with a smaller index. In this way, more indices are distributed around zero, so the embedding scheme only needs a few bits to encode the indices, which produces more extra space for hiding secret data. The experimental results indicate that the proposed scheme is superior to some previous schemes in embedding performance while maintaining the same visual quality as that of VQ compression.

**Keywords:** Reversible data hiding · Vector quantization  
Side-match vector quantization · State-codebook sorting

## 1 Introduction

With the rapid development of multimedia and communication technologies, massive amounts of multimedia data are transmitted over the Internet, which is convenient for people to exchange and share information without time or space constraints. However, such easy access to the Internet makes illegal distribution of multimedia easier and poses key challenges on security protection of confidential data. To solve this security problem, many researchers have proposed various data protection approaches, including encryption and data hiding techniques. An encryption method just encrypts the secret data into a meaningless form using cryptographic algorithms. The unrecognizable form may raise the suspicion of malicious attackers. On the contrary, data hiding is used to embed the secret data imperceptibly in meaningful cover media. In this way, attackers will not be suspicious of the media that carries the secret data.

Generally, data hiding techniques can be classified into three domains, i.e., the spatial domain [1–6], the transform domain [7, 8] and the compression domain [9–17]. In the spatial domain, the cover image is modified directly and undetectably to conceal the secret data. In the transform domain, the secret data is embedded by altering the frequency coefficients of the cover image. However, due to the bandwidth limitation, many image compression schemes, such as vector quantization (VQ) [9], block truncation coding (BTC) [10, 11] and JPEG [12] have been introduced into the

compression domain to save storage and bandwidth space. VQ is a widely used image compression technique with properties of simple implementation and high compression rate. In this work, we mainly focus on the data hiding in VQ-related [13–17] image compressed codes. This hiding technique aims to provide a way for low bandwidth transmission, as well as a means of covert communication.

In 2009, Chang et al. [13] proposed a reversible information embedding method for VQ-compressed image using the joint neighboring coding (JNC) technique. Their method only uses the left or upper neighboring indices to embed secret data and obtains a low hiding capacity. In 2011, Yang et al. [14] proposed an MFCVQ-based reversible data hiding scheme, which use the Huffman-code strategy and 0-centered classification to reduce the bit rate of the output code stream, however the hiding capacity is still low. In the same year, Chang et al. [15] designed a locally adaptive coding scheme in the VQ compression domain. This scheme achieves both good reconstructed image visual quality and a high embedding rate. In 2013, Wang et al. [16] proposed a new VQ-based scheme for reversible data hiding. In this scheme, a technique called adjoining state-codebook mapping (ASCM) is used to improve the side-match vector quantization (SMVQ)-based scheme. Therefore, it can reduce the size of the output code stream by 13% on average when 16,129 bits are embedded in each test image. In 2015, Lin et al. [17] proposed a new reversible data hiding scheme for VQ-compressed images. The search-order coding (SOC) and state-codebook mapping (SCM) are used in the proposed scheme for reducing the size of the VQ index table and for hiding a scalable amount of data. However, the scheme does not work effectively when images have low correlation among neighboring indices.

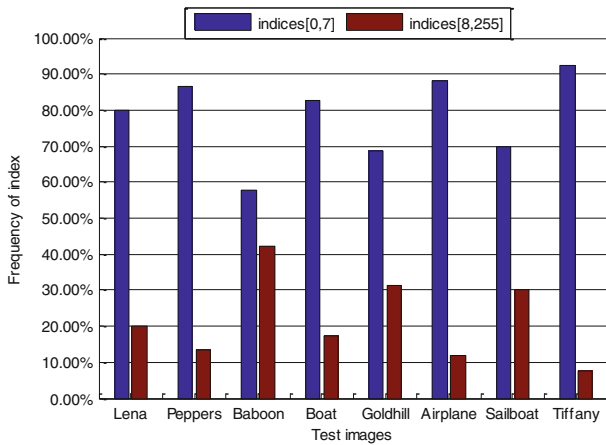
One of the common characteristics of VQ-based data hiding schemes mentioned above is that the secret data are embedded in the compressed code stream. This type of data hiding algorithm focuses on providing larger embedding capacity with lower compressed bit rate. Therefore, considering how to make room to enlarge hiding capacity and how to reduce the index values to further suppress the bit rate are significant subjects of our proposed scheme. Generally, we employ SMVQ to improve the distribution of VQ index table, and adopt a state-codebook sorting (SCS) technique to further reduce the index values of SMVQ index table. So the proposed embedding scheme only needs few bits to encode the indices, thus yielding more space for secret data hiding. Compared with related existing schemes, our proposed method has higher embedding capacity and lower bit rate.

## 2 Proposed Scheme

### 2.1 State-Codebook Sorting

The goals of SMVQ-based data hiding scheme are to embed large amount of secret data and achieve a low compressed bit rate. The main factor in achieving good embedding performance is the distribution of SMVQ index table, which is determined by the state-codebook used for encoding an image block. In the final analysis, the state-codebook is the critical element for SMVQ-based data hiding method. Therefore we propose an SCS method to improve the distribution of SMVQ index table.

As is well-known, correlation among neighboring blocks of a natural image is very strong, which is the key feature exploited in SMVQ, so we can use the border pixels of neighboring blocks to predict the current block. However, when the image block is complex, the prediction result becomes worse, which indicates the state-codebook used for encoding is not appropriate. In order to solve this problem, we adopt a strategy that the block whose index value is larger than threshold  $T$  needs to be re-encoded using new generated state-codebook. The threshold is chosen on the basis of empirical value. The statistics distribution of indices in Fig. 1 shows that most of the high-frequency SMVQ indices of the eight representative images range from 0 to 7, i.e., most of the indices are smaller than 8 (the integer power of 2). With the higher frequency of used indices, better embedding capacity can be achieved while maintaining low bit rate and the good visual quality of the reconstructed image. Therefore we take 8 as the threshold  $T$  to determine whether an SMVQ encoded image block needs to be re-encoded using the new state-codebooks generated by our sorting method. The detailed processes of the state-codebook sorting method are introduced as follows.



**Fig. 1.** High-frequency histogram of state-codebook sorted indices.

To encode a block of an image (except the first row and first column blocks that are encoded by VQ), the scheme first compresses it by the conventional SMVQ algorithm, then compares the index value  $v$  with  $T$ , if it is smaller than  $T$ , the index value is reserved, otherwise our scheme sorts the state-codebook by each codeword’s mean value in both ascending and descending order to create two state-codebooks, which are used again to encode the block to get new index values  $v_1$  and  $v_2$ , then compares them with  $T$ . If only one of  $v_1$  and  $v_2$  is smaller than  $T$ , the scheme replaces  $v$  with that one, or if  $v_1$  and  $v_2$  are both smaller than  $T$ , replaces  $v$  with the smaller one of  $v_1$  and  $v_2$ , and otherwise leaves the original index  $v$  unchanged. In this way, more indices are distributed around zero.

According to the abovementioned SMVQ scheme, a 2-bit indicator is required to distinguish four cases of block encoding results. If a block is encoded using the original

SMVQ index value which is smaller than  $T$ , the indicator is set as “00”, or if a block is re-encoded by the state-codebook sorted in ascending order, the indicator is set as “01”, or the indicator is set as “10” for the case that the state-codebook is sorted in descending order, or else the indicator is set as “11” to denote that a block’s original index value is larger than  $T$  and the state-codebook sorting method cannot effectively reduce the index. In other words, the first three cases obtain the index value that is smaller than  $T$ , while only for the fourth case, the index value is equal to or larger than  $T$ .

Subsequently, the reverse processes of the SCS- based SMVQ are described below. When decoding an SMVQ index  $v$ , firstly judge its 2-bit indicator  $ind$ , if it is “00” or “11”, which means  $v$  is the index value of original state-codebook, so the image block can easily be recovered by table lookup method in original state-codebook. However, if the indicator is “01” or “10”, we need to sort the state-codebook in ascending order or descending order to produce the sorted state-codebook, in which the corresponding image block is found by table lookup. Then these image blocks compose the recovered image.

## 2.2 Data Embedding Process

In the data embedding process, the secret bits can be easily embedded into the index table  $IT$ . Similar to the scheme proposed by Wang et al. [16], our scheme uses a flag of two bits to identify the index encoding types, and embeds secret data in the index whose value is smaller than  $T$ . In order to enlarge the embedding capacity while maintaining a low output bit rate, 4 bits secret data is embedded in each embeddable index and another threshold  $T_a$  is applied to further divide the index, whose indicator is “11”, into two parts, i.e.,  $P_1 \in [T, T_a - 1]$  and  $P_2 \in [T_a, M - 1]$ . Since index values are divided into three groups using threshold  $T$  and  $T_a$ , the index smaller than  $T$  ( $T = 8$ ) can be encoded by 3 bits, and the index with indicator “11” (i.e.,  $P_1$  and  $P_2$ ) needs another 1-bit indicator to identify which group the index belongs to, so  $P_1$  is encoded as  $1\|(P_1 - T)_2$  and  $P_2$  is encoded as  $0\|(P_2)_2$ , where “||” denotes the concatenation operation and  $(x)_2$  means the binary representation of  $x$ . It is worth noting that the length of  $P_1$  should be power of 2, i.e.,  $T_a = 2^i + T$  ( $i = 1, 2, \dots, \log_2 M - 1$ ) into the secret data. Though additional 1-bit flag is required, we can effectively reduce the bit rate of final code stream. Because, on the one hand, more indices are smaller than  $T$ , on the other hand, the code length of  $P_1$  is much smaller than the bit size if  $P_1$  and  $P_2$  are encoded as one group.

One thing to note is that the indices in the first row and the first column of  $IT$  are the seed area, where the index values will be directly transformed into binary form of  $\log_2 N$  bits without embedding any secret data. The residual indices of  $IT$  will be processed in order from left-to-right and top-to-bottom by the proposed embedding algorithm which is described in detail as follows:

Input: Grayscale cover image  $I$ ,  $N$ -sized super codebook  $C$ , threshold  $T$  and  $T_a$ , secret data stream  $B$ .

Output: Code stream  $CS$  in binary form.

- Step 1: Encode image  $I$  using SCS-based SMVQ technique to obtain the index table  $IT$  and the block encoding indicator table  $indT$ .
- Step 2: Read index  $v$  which is not in the seed area from the index table  $IT$ .
- Step 3: Read index  $v$ 's corresponding indicator  $ind$  from encoding indicator table  $indT$ .
- Step 3.1: if  $ind$  is "11", this means that  $v$  is equal to or larger than  $T$ , and no secret data will be embedded in this index.
- Step 3.1.1: If  $v < T_a$ ,  $v$  is encoded by  $q = 11\|1\|(v - T)_2$ , where the size of  $(v - T)_2$  is  $\log_2(T_a - T)$ .
- Step 3.1.2: If  $v \geq T_a$ ,  $v$  is encoded by  $q = 11\|0\|(v)_2$ , where the size of  $(v)_2$  is  $\log_2 M$ .
- Step 3.2: Otherwise, i.e., the  $ind$  is "00", "01" or "10", which means the index  $v$  is smaller than  $T$ .
- Step 3.2.1: If  $B$  is not empty, fetch 4 bits secret data as  $s$  and remove them from  $B$ ,  $v$  is encoded by  $q = ind\|(v)_2\|s$ , where the size of  $(v)_2$  is 3.
- Step 3.2.2: If  $B$  is empty,  $v$  is encoded by  $q = ind\|(v)_2$ , where the size of  $(v)_2$  is 3.
- Step 4: Send  $q$  to the code stream  $CS$ .
- Step 5: Repeat Step 2 to Step 4 until all indices in the index table  $IT$  are processed.
- Step 6: Output the code stream  $CS$ .

After the above steps are processed completely, each index of  $IT$  can be encoded to form the code stream  $CS$ . The sender then distributes the encoded binary stream  $CS$  to the receiver.

### 2.3 Data Extracting Process

At the receiving side, with the received code stream  $CS$  and the size of state-codebook  $M$ , the decoder can extract the embedded secret message and reconstruct the original SMVQ indices simultaneously, and after that the cover image can be restored by decoding SMVQ indices with  $N$ -sized super codebook  $C$ . Initially, the indices in the seed area are recovered directly by converting every  $\log_2 N$  bits codes, which are read from  $CS$ , to decimal value. The residual indices in non-seed area and secret data are reconstructed easily through distinguishing four cases with the help of indicators. The detailed processes for extracting secret data and restoring the indices in the non-seed area are described as follows:

Input: The binary code stream  $CS$ ,  $N$ -sized super codebook  $C$ , the size of state-codebook  $M$ , threshold  $T$  and  $T_a$ , the number of unrecovered secret data  $N_x$ .

Output: The secret data stream  $B$  and the original SMVQ-compressed index table  $IT$ .

- Step 1: Set the secret bit stream  $B$  empty.
- Step 2: Read 2 bits indicator from the binary code stream  $CS$  as  $ind$ .

- Step 3: if *ind* is “11”, this means the index value is equal to or larger than  $T$  and no secret data is embedded in the index. Then read the next one bit from *CS* as *lab*.
- Step 3.1: If *lab* is “1”, which means  $T \leq v' < T_a$ , then read next  $\log_2(T_a - T)$  bits from *CS* as  $q$ ,  $v'$  is reconstructed by converting  $q$  to decimal value then adding with  $T$ .
- Step 3.2: If *lab* is “0”, which means  $v' \geq T_a$ , then  $v'$  is reconstructed by reading next  $\log_2 M$  bits from *CS* and converting them to decimal value.
- Step 4: Otherwise, i.e., the *ind* is “00”, “01” or “10”, which means the index value  $v'$  is smaller than  $T$ . And  $v'$  is reconstructed by reading next 3 bits from *CS* and converting them to decimal value.
- Step 4.1: If  $N_x > 0$ , 4 subsequent bits are extracted from *CS* as secret data  $s'$ , then concatenate  $s'$  into the secret data stream  $B$  (i.e.  $B = B || s'$ ) and let  $N_x = N_x - 4$ .
- Step 4.2: If  $N_x = 0$ , there is no secret data need to be extracted.
- Step 5: Put  $v'$  in the recovered index table *IT*.
- Step 6: Repeat Step2 to Step5 until all the bits of code stream *CS* are processed.

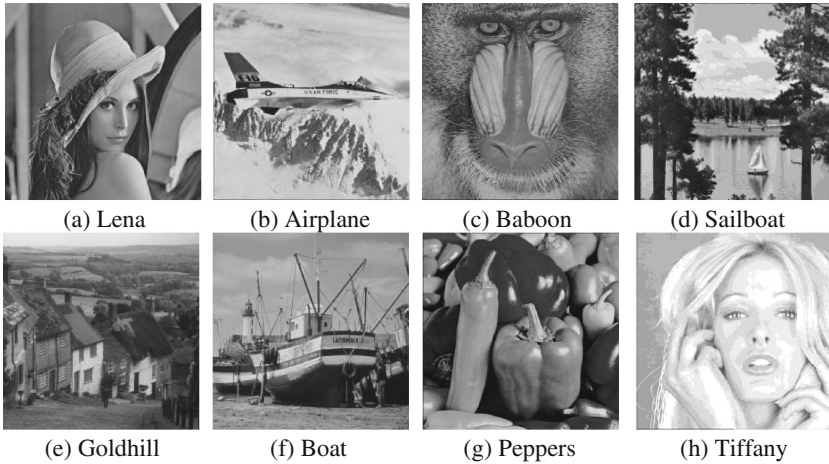
According to the restored SMVQ index table *IT* and the super codebook *C*, we can reconstruct the original image by the reverse process of SCS-based SMVQ. In this way, the whole decoding process is realized.

### 3 Experimental Results and Analysis

In this section, the experimental results are presented to demonstrate the performance of the proposed scheme. Figure 2 shows eight test images of size  $512 \times 512$ , each image is divided into 16384 non-overlapping  $4 \times 4$  blocks in the codebook encoding procedure. The super codebook of sizes 512 with codewords of 16 dimensions were trained by the LBG algorithm. The secret data in the experiment is in binary format, 0 and 1, which are generated by a pseudo-random number generator, and four bits are embedded in each embeddable index. If a high level of security is required, secret data can be encrypted prior to embedding using well-known cryptographic methods as DES or RSA.

In these experiments, four criteria are employed to evaluate the performance of our proposed scheme, i.e., visual quality, compressed bit rate, embedding rate and embedding efficiency. The peak signal-to-noise (PSNR) ratio is used to estimate the degree of distortion between the original cover image and the stego-image. The compressed bit rate  $R_b$  is used to evaluate the compression performance of an encoding algorithm, which is defined as  $R_b = ||CS|| / (H \times W)$ , where  $||CS||$  represents the total length of the output code stream,  $H$  and  $W$  is the height and width of the cover image. The embedding rate  $R_e$ , which describes how many secret bits can be embedded into one index of the SMVQ index table, is defined as  $R_e = ||S|| / N_{IDX}$ , where  $||S||$  is the number of confidential bits embedded into the SMVQ index table, namely  $C_p$ , and  $N_{IDX}$  is the number of indices in the index table generated by SMVQ. The last criterion is the

embedding efficiency  $E_e$ , which describes the ratio of the size of secret data to the length of output code stream, is defined as  $E_e = ||S||/||CS||$ .



**Fig. 2.** Eight grayscale test images of size  $512 \times 512$ .

### 3.1 Performance of SCS

The motivation for the SCS technique is to increase the probability of getting small indices. Experimental results in Table 1 illustrate the feasibility of reducing the index value by using SCS method.  $M$  is the size of state-codebook used to generate SMVQ index table. When  $M = 512$ , it means the image quality of SMVQ is as same as that of VQ, and the index value is reduced by 7.18% on average. On the other hand, a smaller size of state-codebook ( $M = 32$ ) can obviously achieve an average of 64.42% reduction of the index value.

**Table 1.** Image quality and percentage of index value improvement.

Image	$M = 32$		$M = 512$	
	PSNR (dB)	Percentage (%)	PSNR (dB)	Percentage (%)
Lena	30.1	65.17	32.2	2.89
Peppers	29.6	60.82	31.4	3.65
Baboon	23.1	59.72	24.4	7.44
Boat	28.5	67.36	30.2	12.32
Goldhill	29.2	63.39	30.8	0.23
Airplane	29.7	70.79	31.6	13.49
Sailboat	27.3	61.02	28.6	2.13
Tiffany	30.6	67.08	32.3	15.29
Average	28.5	64.42	30.2	7.18

### 3.2 Parameters Choosing

In order to maintain the same image quality as VQ, we make the size of state-codebook equal to that of super codebook, i.e.,  $M = N = 512$ . Table 2 is a comparison between our scheme and Wang et al.’s scheme [16]. Evidently, the visual quality of our proposed scheme is same to that of Wang et al.’s scheme and VQ compression. Since there are no distortions in the index encoding and data hiding process and all distortions only occur in the SMVQ-index table generating process, the PSNR value of the stego-image is exactly equal to that of the SMVQ compressed cover images.

**Table 2.** Visual quality results (dB).

Image	Proposed	Wang et al.’s scheme	VQ
Lena	32.2	32.2	32.2
Peppers	31.4	31.4	31.4
Baboon	24.4	24.4	24.4
Boat	30.2	30.2	30.2
Goldhill	30.8	30.8	30.8
Airplane	31.6	31.6	31.6
Sailboat	28.6	28.6	28.6
Tiffany	32.3	32.3	32.3

The threshold  $T_a$  is another important parameter which affects the compressed bit rate of the output code stream, therefore we also test which threshold can lead to a better result of our proposed scheme. The bit rate results with different  $T_a$  are shown in Table 3, where no secret data is embedded in the test images. The bit rate values are in the range of 0.338–0.506 and the best one for each test images is shown in bold. We can easily find that, with the increase of  $T_a$  the bit rate decreased first then increased, therefore the better value for  $T_a$  would be 40 since more than half of the test images have better compressed bit rate when  $T_a$  is 40. The following experiments are done under these conditions.

**Table 3.** The bit rate of proposed scheme with different  $T_a$  (bpp).

$T_a$	12	16	24	40	72	136
Lena	0.402	0.392	0.385	<b>0.383</b>	0.387	0.398
Peppers	0.382	0.371	<b>0.366</b>	0.368	0.376	0.387
Baboon	0.506	0.494	0.481	0.472	<b>0.470</b>	0.480
Boat	0.393	0.385	0.378	<b>0.374</b>	0.377	0.386
Goldhill	0.459	0.441	0.427	<b>0.422</b>	0.429	0.447
Airplane	0.366	0.361	0.356	<b>0.354</b>	0.356	0.362
Sailboat	0.455	0.441	0.427	<b>0.418</b>	0.422	0.438
Tiffany	0.345	0.341	0.338	<b>0.338</b>	0.340	0.345



### 3.3 Comparison Results with Related Schemes

We firstly considered only the scheme of Wang et al. [16] for performance comparisons of bit rate and embedding efficiency by embedding various sizes of secret data into the SMVQ index table. The bit rate comparison results of the two schemes are shown in Table 4. It is observed that, with the increase of embedding capacity, the encoding length for each index becomes longer, therefore the bit rates of the two schemes become gradually increased. In addition, under the same embedding capacity, our scheme achieves a lower bit rate than that of Wang et al.’s scheme for each test image. And the reduction of bit rate can be 1.7% to 9.1%, the average improvement of bit rate for our scheme is nearly 6.2%. The reason for achieving a lower bit rate is that in our proposed scheme the indices belonging to  $P_1 \in [T, T_a - 1]$ , which embeds no secret data, can be encoded using few bits. On the other hand, the number of indices whose values are smaller than  $T$  is increased by SCS method, accordingly the number of indices that cannot be used for data embedding is decreased, hence the length of code stream is reduced.

**Table 4.** Bit rate comparison for embedding various sizes of secret data (bpp).

	Capacity	Lena	Peppers	Baboon	Boat	Goldhill	Airplane	Sailboat	Tiffany
Wang et al.’s scheme ( $SCI = 3$ )	0	0.417	0.405	0.482	0.397	0.458	0.382	0.458	0.358
	5000	0.436	0.424	0.501	0.416	0.477	0.4	0.477	0.377
	10000	0.455	0.443	0.52	0.435	0.496	0.419	0.496	0.396
	20000	0.493	0.481	0.558	0.473	0.534	0.456	0.535	0.434
	30000	0.532	0.519	0.596	0.511	0.572	0.496	0.573	0.472
Proposed scheme	0	0.383	0.368	0.472	0.374	0.422	0.354	0.418	0.338
	5000	0.402	0.387	0.491	0.393	0.441	0.372	0.437	0.357
	10000	0.421	0.406	0.51	0.412	0.46	0.391	0.456	0.376
	20000	0.459	0.444	0.548	0.45	0.498	0.43	0.495	0.414
	30000	0.498	0.482	0.586	0.489	0.536	0.466	0.533	0.452

**Table 5.** Embedding efficiency comparison for embedding various sizes of secret data (%).

	Capacity	Lena	Peppers	Baboon	Boat	Goldhill	Airplane	Sailboat	Tiffany
Wang et al.’s scheme ( $SCI = 3$ )	5000	4.4	4.5	3.8	4.6	4.0	4.8	4.0	5.1
	10000	8.4	8.6	7.3	8.8	7.7	9.1	7.7	9.6
	20000	15.5	15.9	13.7	16.1	14.3	16.7	14.3	17.6
	30000	21.5	22.0	19.2	22.4	20.0	23.1	20.0	24.2
Proposed scheme	5000	4.7	4.9	3.9	4.9	4.3	5.1	4.4	5.3
	10000	9.1	9.4	7.5	9.3	8.3	9.8	8.4	10.2
	20000	16.6	17.2	13.9	16.9	15.3	17.7	15.4	18.4
	30000	23.0	23.7	19.5	23.4	21.4	24.6	21.5	25.3

Table 5 presents the corresponding embedding efficiency for the comparison experiment. The results show that, the embedding efficiencies are respectively higher than that of Wang et al.'s scheme, therefore a better performance can be achieved by using our SCS-based SMVQ technique. The increased percentage of embedding efficiency ranges from 1.5% to 10.0%, on average 6.3%, which indicates if the sizes of output code stream produced by the two schemes are same, the number of secret bits that can be embedded in our scheme will increase by 6.3% on average compared with Wang et al.'s scheme.

We have conducted another experiment to evaluate the performance of the proposed scheme by embedding the maximum numbers of secret bits into the index table. The results in Table 6 show that, the smooth image can get better results, such as image "Tiffany", which has the highest hiding capacity and embedding efficiency. While the results of complex image become worse to some extent, the embedding rate of image "Baboon" is only 2.00 bpi, which is the lowest one of the test images. However, the embedding rate of proposed scheme is about 2.85 bpi on average, which means almost 2.85 bits secret data can be embedded in one index. Additionally, the average bit rate is 0.560 bpp, and the average embedding efficiency is 31.4%.

**Table 6.** The results for embedding maximum sizes of secret data in different test images.

Image	$C_p$ (bit)	$R_e$ (bpi)	$R_b$ (bpp)	$E_e$ (%)
Lena	48040	2.93	0.566	32.4
Peppers	50096	3.06	0.559	34.2
Baboon	32788	2.00	0.597	20.9
Boat	50356	3.07	0.566	33.9
Goldhill	37868	2.31	0.566	25.5
Airplane	55320	3.38	0.565	37.4
Sailboat	39636	2.42	0.569	26.6
Tiffany	58884	3.59	0.562	39.9
Average	46624	2.85	0.569	31.4

To demonstrate the superiority of the proposed scheme, three related schemes, i.e., Chang et al.'s scheme [13], Wang et al.'s scheme [16] and Lin et al.'s scheme [17] were finally compared with the proposed scheme. The sizes of codebooks adopted in all schemes are 512 and the suggested optimal settings are used as given in their corresponding algorithms. Table 7 shows the comparison results for embedding 16,129 bits in different test images. For most test images, the proposed scheme achieves the optimal result compared with other schemes.

In terms of bit rate, we can see that the proposed scheme generally produces the smallest one. Although Lin et al.'s scheme gets lower bit rate for simple images such as "Airplane" and "Tiffany", the average bit rate is 1.5% higher than that of the proposed scheme, because the proposed scheme can greatly reduce the bit rate of complex images. Compared with Wang et al.'s scheme and Chang et al.'s scheme, the reduction of bit rate is 6.4% and 18.8% respectively.

On the other hand, the embedding efficiency of our scheme has been increased in comparison with other schemes. Since under the same embedding capacity, the embedding efficiency is only determined by the total length of the output code stream, therefore, a data hiding scheme with low bit rate will achieve a high embedding efficiency. In our scheme, the average embedding efficiency for embedding 16,129 bits is nearly 13.7%, which outperforms all the other schemes: Chang et al.'s scheme (11.1%), Wang et al.'s scheme (12.8%), and Lin et al.'s scheme (13.5%).

**Table 7.** Comparison on bit rate and embedding efficiency of different data hiding schemes for embedding 16,129 bits.

Image	Chang et al.'s scheme ( $m = 6$ )		Wang et al.'s scheme ( $SCI = 3$ )		Lin et al.'s scheme ( $SOL = 2, SCL = 8$ )		Proposed scheme	
	$R_b$ (bpp)	$E_e$ (%)	$R_b$ (bpp)	$E_e$ (%)	$R_b$ (bpp)	$E_e$ (%)	$R_b$ (bpp)	$E_e$ (%)
Lena	0.552	11.2	0.479	12.8	0.440	14.0	0.445	13.8
Peppers	0.545	11.3	0.466	13.2	0.426	14.4	0.429	14.3
Baboon	0.608	10.1	0.552	11.2	0.546	11.3	0.534	11.5
Boat	0.556	11.1	0.471	13.1	0.459	13.4	0.436	14.1
Goldhill	0.581	10.6	0.519	11.9	0.511	12.0	0.483	12.7
Airplane	0.537	11.5	0.445	13.8	0.407	15.1	0.415	14.8
Sailboat	0.582	10.6	0.520	11.8	0.508	12.1	0.480	12.8
Tiffany	0.503	12.2	0.419	14.7	0.385	16.0	0.399	15.4
Average	0.558	11.1	0.484	12.8	0.460	13.5	0.453	13.7

In short, the above experiments demonstrate that the proposed algorithm has a better performance in both embedding capacity and bit rate. Additionally, in certain test images or especially in complex images, our scheme achieves a lower bit rate than other schemes because large amount of index values are smaller than  $T$  due to the SCS-based SMVQ technique.

## 4 Conclusion

We propose a new reversible data hiding method for SMVQ compressed images. By employing the SCS-based SMVQ technique, our scheme achieves a higher embedding capacity with lower bit rate than related schemes. Since the SCS method is used to further reduce the index values of SMVQ index table, the proposed embedding scheme only needs few bits to encode the indices, thus yielding more space for secret data hiding. Though a 2-bit flag is required to distinguish the types of indices, we can effectively reduce the bit rate of the final code stream because large amounts of index values are smaller than  $T$ , and the indices belonging to  $[T, T_a - 1]$  are used to reduce the code length, with no secret data embedded. Several experiments were conducted with 512 codewords to evaluate the performance of proposed scheme. The comparison

results show that the proposed scheme outperforms other schemes both in compression rate and embedding efficiency. Especially, for the complex images “Baboon” and “Boat”, the embedding efficiency is improved by 1.8% and 5.2% compared with Lin et al.’s scheme [17]. Moreover, the proposed scheme maintains the same visual quality as that of VQ compression.

**Acknowledgement.** This work is supported by the National Natural Science Foundation of China (No. 61372175).

## References

1. Rudder, A., Kieu, T.D.: A lossless data hiding scheme for VQ indexes based on joint neighboring coding. *KSII Trans. Internet Inform. Syst.* **9**(8), 2984–3004 (2015)
2. Li, X., Zhou, Q.: Histogram modification data hiding using chaotic sequence. In: Wang, W. (ed.) *Mechatronics and Automatic Control Systems*. LNEE, vol. 237, pp. 875–882. Springer, Cham (2014). [https://doi.org/10.1007/978-3-319-01273-5\\_98](https://doi.org/10.1007/978-3-319-01273-5_98)
3. Li, X.L., et al.: General framework to histogram-shifting-based reversible data hiding. *IEEE Trans. Image Process.* **22**(6), 2181–2191 (2013)
4. Hong, W., Chen, T.S., Luo, C.W.: Data embedding using pixel value differencing and diamond encoding with multiple-base notational system. *J. Syst. Softw.* **85**(5), 1166–1175 (2012)
5. Hong, W., Chen, T.S.: A novel data embedding method using adaptive pixel pair matching. *IEEE Trans. Inf. Forensics Secur.* **7**(1), 176–184 (2012)
6. Hong, W.: Adaptive image data hiding in edges using patched reference table and pair-wise embedding technique. *Inform. Sci.* **221**(1), 473–489 (2013)
7. Fang, H., Zhou, Q., Li, K.J.: Robust watermarking scheme for multispectral images using discrete wavelet transform and tucker decomposition. *J. Comput.* **8**(11), 2844–2850 (2013)
8. Su, P.C., Chang, Y.C., Wu, C.Y.: Geometrically resilient digital image watermarking by using interest point extraction and extended pilot signals. *IEEE Trans. Inf. Forensics Secur.* **8**(12), 1897–1908 (2013)
9. Wang, W.J., Huang, C.T., Wang, S.J.: VQ applications in steganographic data hiding upon multimedia images. *IEEE Syst. J.* **5**(4), 528–537 (2011)
10. Guo, J.M., Liu, Y.F.: High capacity data hiding for error-diffused block truncation coding. *IEEE Trans. Image Process.* **21**(12), 4808–4818 (2012)
11. Guo, J.M., Tsai, J.J.: Reversible data hiding in low complexity and high quality compression scheme. *Digit. Signal Proc.* **22**(5), 776–785 (2012)
12. Mobasser, B.G., et al.: Data embedding in JPEG bit stream by code mapping. *IEEE Trans. Image Process.* **19**(4), 958–966 (2010)
13. Chang, C.C., Kieu, T.D., Wu, W.C.: A lossless data embedding technique by joint neighboring coding. *Pattern Recogn.* **42**(7), 1597–1603 (2009)
14. Yang, C.H., et al.: Huffman-code strategies to improve MFCVQ-based reversible data hiding for VQ indexes. *J. Syst. Softw.* **84**(3), 388–396 (2011)
15. Chang, C.C., Nguyen, T.S., Lin, C.C.: A reversible data hiding scheme for VQ indices using locally adaptive coding. *J. Vis. Commun. Image Represent.* **22**(7), 664–672 (2011)
16. Wang, W.J., et al.: Data embedding for vector quantization image processing on the basis of adjoining state-codebook mapping. *Inform. Sci.* **246**(14), 69–82 (2013)
17. Lin, C.C., Liu, X.L., Yuan, S.M.: Reversible data hiding for VQ-compressed images based on search-order coding and state-codebook mapping. *Inform. Sci.* **293**, 314–326 (2015)