

Multiple Bits Embedding Approach for the Hit-pattern-based VQ Reversible Steganography

Cheng-Hsing Yang*, Cheng-Ta Huang, Shu-Chien Huang, and Sheng-Chang Wu

Department of Computer Science, National Pingtung University of Education,
Pingtung, Taiwan, 900

{chyang, bm096118, schung, bm097115}@mail.npue.edu.tw

Abstract. Data hiding is a technique which embeds secret data into cover media. It is important to the multimedia security and has been widely studied. Recently, some researchers paid attention to reversible data hiding methods. These methods can reconstruct the original image from the stego-image when embedded data are extracted. In this paper, we extend the hit-pattern-based reversible steganographic method for vector quantization (VQ) compressed images. A block is embedded with 2-bit secret data, which efficiently enlarges the embedding capacity. Also, threshold-bounded hit patterns efficiently keep the embedded results with high PSNR values. Compared to Chang et al.'s method and Yang et al.'s method, the experimental results show that the proposed method has both higher capacities and better PSNR values.

Keywords: Data hiding, Reversible data hiding, Steganography, Vector quantization, Side match vector quantization.

1 Introduction

Data hiding is one of the most important technologies for the multimedia security. Digital multimedia can be copied easily or tampered illegally. Therefore, integrity authentication and copyright protection of digital multimedia have become thorny and critical issues. For digital images, a lot of data hiding techniques have been proposed for different purposes, such as secret communication, copyright protection, and tampering detection [1, 2]. These techniques embed secret messages into original images and create stego-images.

Digital images are usually stored in compressed formats, such as JPEG, vector quantization (VQ), and side match vector quantization (SMVQ). Some hiding schemes based on VQ or SMVQ have been proposed [3, 4]. Fig. 1 shows the process of VQ encoding where the codebook size and the vector size are 256 and 16, respectively. The cover image is divided into 4×4 non-overlapping blocks. Then,

* Correspondence addressee: Cheng-Hsing Yang, Dept. of Computer Science,
National Pingtung University of Education, Pingtung, Taiwan 900
E-mail: chyang@mail.npue.edu.tw, Fax: +886-8-7215034, Tel: +886-8-7226141 ext. 33501

each block finds out the closest codeword from the codebook, and outputs the index of this codeword to the index table. For example, codeword cw_1 is selected by the first block, and index 1 is recorded in the index table.

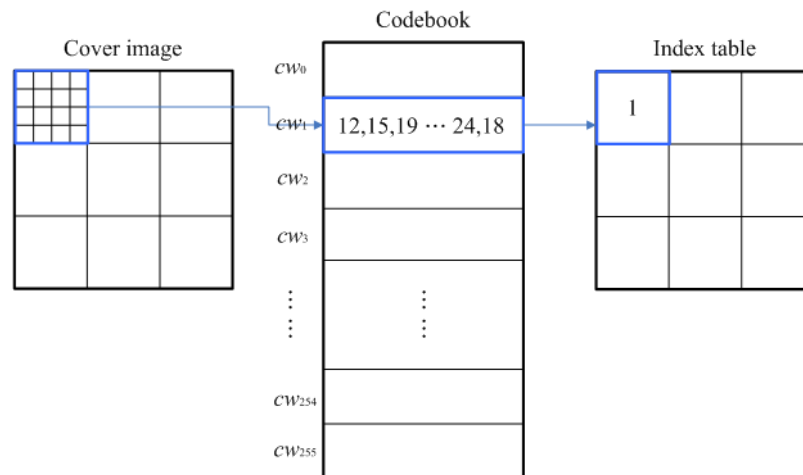


Fig. 1. The process of VQ encoding.

Some data hiding techniques have the reversibility characteristic, which can recover the original image after the embedded message is extracted [5-10]. In these developed reversible hiding methods for original images, the difference expansion [5-7] and the histogram [8-10] are two mainly technologies. On the other hand, some reversible hiding methods for VQ-compressed images have been proposed [11-18]. These methods refer to owning the ability of extracting hidden data and recovering the embedded results into the original VQ codes or the SMVQ codes. In those developed reversible hiding approaches based on VQ or SMVQ, they can be classified into three categories according to their outputs: images as outputs [11, 12], legitimate VQ codes or SMVQ codes as outputs [13, 14], and VQ codes or SMVQ codes with additional control messages as outputs [15-18].

(1) Images as outputs.

After data are embedded, some approaches are limited to producing images as outputs [11, 12]. Chang et al.'s method [11] is SMVQ reversible. Another Chang et al.'s method [12] is VQ reversible.

(2) Legitimate VQ codes or SMVQ codes as outputs.

After data are embedded, the legitimate VQ codes or SMVQ codes are as outputs [13, 14]. In Chang et al.'s reversible hiding method [13], it is based on the side-match approach and utilizes the local characteristics of the image. In Chang et al.'s method [14], they proposed a modified side match vector quantization (SMVQ) technique.

(3) VQ codes or SMVQ codes with additional control messages as outputs.

After data are embedded, the outputs contain some leading messages within VQ codes or SMVQ codes [15-18]. In Chang et al.'s method [15], it is based on a declustering strategy and takes advantage of the local spatial characteristics of the image. In Chang et al.'s method [16], they used a VQ codebook, which was clustered

into three groups, to achieve secret concealment and data recovery. In Yang et al.'s method [17], a sorted VQ codebook is divided into 2^B clusters and half of clusters are used to embed secret data. Chang et al.'s method [18] embedded a secret message into VQ indices of an index table during the process of compressing the index table by the block-by-block manner.

In this paper, we extend the hit-pattern-based VQ reversible hiding scheme to embed more than one bit of secret data into a block [19]. Both input and output of our method are legitimate VQ compressed codes. Also, the original VQ compressed codes can be recovered after the embedded data are extracted. In our method, the codebook is partitioned into state codebooks for each processed block by the extension method, in which each codeword of a state codebook finds its closest codeword from the remained codewords as the corresponding codeword of the next state codebook. In the embedding process, the threshold-bounded hit patterns which deal with conflict cases are small and are easy to be embedded together with secret data. The remainder of this paper is organized as follows. In Section 2, Chang et al.'s method is introduced. Our proposed method is presented in Section 3, and the experimental results are shown in Section 4. Finally, we draw our conclusions in Section 5.

2 Literature review

In this section, Chang et al.'s reversible embedding method is introduced [13]. They used the concept of SMVQ to create three state codebooks G_0 , G_1 , and G_2 for each processed block X . Figure 2 shows the relationship of three 4×4 blocks, where block U is the upper adjacent block of X and block L is the left adjacent block of X . Both U and L have been encoded. First, the border values of X are temporarily assigned by its upper and left adjacent blocks U and L , such that $x_0 = (u_{12} + l_3)/2$, $x_1 = u_{13}$, $x_2 = u_{14}$, $x_3 = u_{15}$, $x_4 = l_7$, $x_8 = l_{11}$, and $x_{12} = l_{15}$. Second, the seven assigned values are used to search the n closest codewords Y_i 's according to the following equation:

$$D(X, Y_i) = \sum_{j=0}^3 (x_j - y_{i,j})^2 + \sum_{j=1}^3 (x_{4j} - y_{i,4j})^2, \quad (1)$$

where n is the size of the state codebook, x_j and $y_{i,j}$ are j th elements of X and Y_i . Then, the n closest codewords are used to construct a state codebook. Finally, the codeword of the state codebook with the minimum Euclidean distance from X is used to encode X .

created by training each block X as follows: If X is equal to the i th codeword of its sorted codebook, the i th bit of the hit map is set to 1. Otherwise, the i th bit of the hit map remains as 0. The sorted codebook consists of the codewords sorted by Eq. (1) in the super codebook. Moreover, in order to enlarge the number of positions with hit value 0, 12 hit maps are proposed and separated according to the smoothness in the neighborhood of each block X .

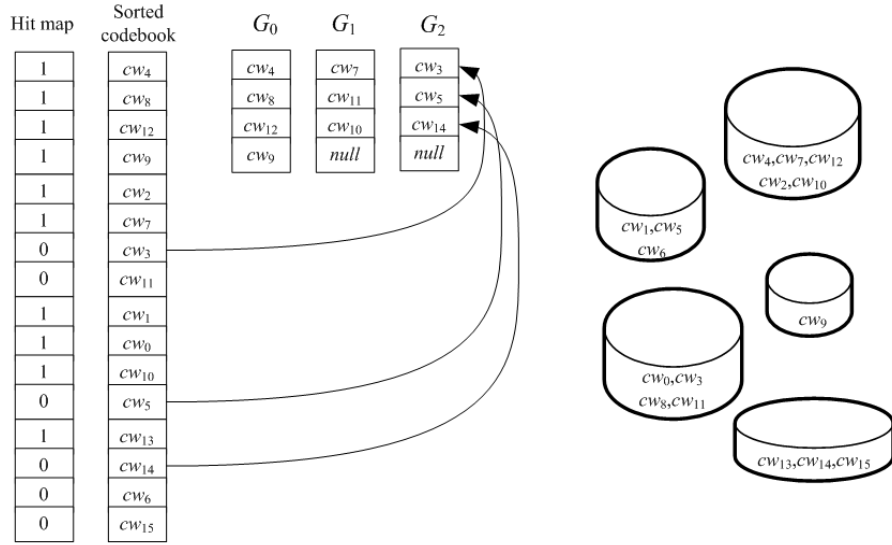


Fig. 3. An example of generating G_0 , G_1 , and G_2 .

3 Our proposed method

Yang et al. has proposed a hit-pattern approach to perform a VQ reversible scheme [19]. Compared to the size of Chang et al.'s multiple hit-maps [13], the hit-pattern is much smaller. Moreover, the hit pattern is easy to be embedded together with the secret data. The drawback of the hit-pattern approach is that the PSNR values of the embedded results are worse. In this section, we extend the hit-pattern-based VQ reversible hiding scheme to embed multiple bits into a block [19]. Also, threshold-bounded hit patterns are proposed to improve the PSNR results. The following descriptions show the case that the embedding size of a block is 2 bits.

3.1 Reversible hiding scheme

For convenience of description, as shown in Fig. 2, the current processed block is named as block X , and the left block and upper block of X are L and U , respectively. The flowchart of processing block X is shown in Fig. 4. Firstly, codebook C is

partitioned into m state codebooks, S_0, S_1, \dots, S_{m-1} with the same size, where S_0 is called the *embedding* state codebook, S_1, S_2, \dots, S_{m-4} are called the *middle* state codebooks, and $S_{m-3}, S_{m-2}, S_{m-1}$ are called the *final* state codebooks. S_0 is generated based on the SMVQ method, which uses left block L and upper block U to predict and select codewords from codebook C . S_i is generated by using codewords in S_{i-1} to find their closest codewords from remained codewords in C , where $i = 1, 2, \dots, m-1$. This extension method can reduce the distance between each pair of corresponding codewords in S_{i-1} and S_i . Now, X will fall into one of the three following cases.

Case (a): If the block X is equal to the i th codeword of S_0 , the embedding process is invoked. If the to-be-embedded bits are 00, X remains unchanged; on the other hand, if the to-be-embedded bits are 01, 10, and 11, X is transformed into the i th codeword of S_1, S_2 , and S_3 , respectively.

Case (b): If block X is equal to the i th codeword of one middle state codebook S_j , X is transformed into the i th codeword of S_{j+3} . If the original X is in S_{m-6}, S_{m-5} , and S_{m-4} , a hit bit 0 is outputted.

Case (c): If block X is in the final state codebooks $S_{m-3}, S_{m-2}, S_{m-1}$, a hit bit 1 is outputted and X remains unchanged.

The overview of our embedding and transforming strategy for block X is shown in Fig. 5. Arrows show the transforming rules. Two secret bits are embedded when X is in S_0 , and one hit bit is created when X is in $S_{m-6}, S_{m-5}, \dots, S_{m-1}$.

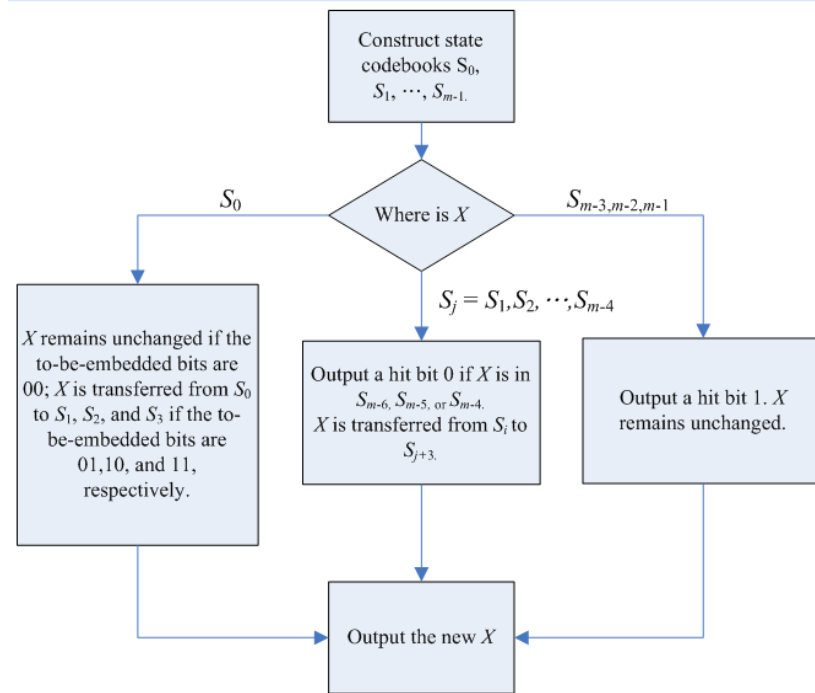


Fig. 4. The embedding flowchart of block X with 2 embedded bits.

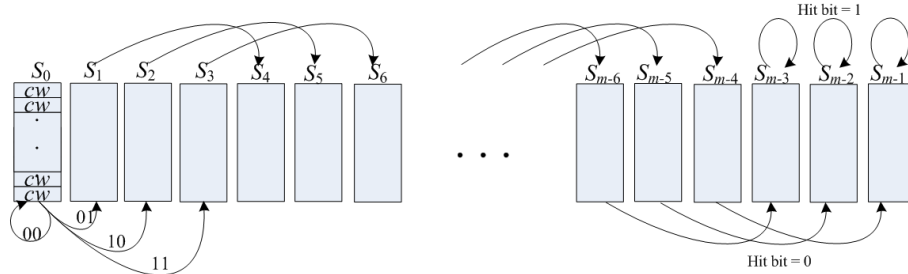


Fig. 5. The transforming rule of our embedding method.

3.2 Embedding algorithm and extracting algorithm

The algorithm of our reversible embedding method is presented below. Both input and output of our algorithm are index tables, which are legitimate VQ compressed codes.

Algorithm Embedding

Input: Index table I , codebook C , secret W

Output: Embedded index table I' , hit pattern H

1. Indexes belonging to the first row or the first column of I do not participate in the embedding process. For each index I_i in the first row or the first column, set $I'_i = I_i$.
2. From left to right and from top to down, process each remained index I_i by executing Step 3 ~ Step 6.
3. For index I_i , construct state codebooks S_0, S_1, \dots, S_{m-1} from codebook C .
4. If $I_i \in S_0$, two bits are embedded. If the to-be-embedded bits in W are 00, set $I'_i = I_i$. If the to-be-embedded bits in W are 01, 10, and 11, set I'_i to the corresponding index in S_1, S_2 , and S_3 , respectively.
5. If $I_i \in S_j$ and $j = 1, 2, \dots, m-4$, no embedding. Set index I'_i to the corresponding index in S_{j+3} . If $I_i \in S_{m-6}, S_{m-5}$, and S_{m-4} , output a hit bit 0 to hit pattern H .
6. If $I_i \in S_{m-3}, S_{m-2}$, and S_{m-1} , no embedding. Set $I'_i = I_i$, and output a hit bit 1 to hit pattern H .

Figure 6 shows an example to describe the embedding process, where 32 state codebooks are used. Indexes located in the first row or the first column of the index table remain unchanged and are used as seek blocks. Then, the other indexes are performed by executing Step 3 to Step 6 from left to right and from top to down. Each index needs to construct its own state codebooks S_0, S_1, \dots, S_{m-1} . However, in Fig. 6, state codebooks are fixed for the purpose of ease explanation. The first index 46 belongs to S_0 . Therefore, Step 4 is executed. Because the to-be-embedded secret bits are 10, so I'_i is set to the corresponding index 13 in S_2 . Next, index 44 is processed and it belongs to S_0 . Therefore, Step 4 is executed. The to-be-embedded secret bits are 00, so I'_i is set to 44. The third index is 89 in S_0 and the to-be-embedded secret bits are 11, so I'_i is set to the corresponding index 15 in S_3 . The fourth index is 58 in S_0 and the

to-be-embedded secret bits are 01, so I'_i is set to the corresponding index 69 in S_1 . The next index is 27 in S_1 , Step 5 is executed. There is no embedding and I'_i is set to the corresponding index 29 in S_4 . The sixth index is 13 in S_2 , there is no embedding and I'_i is set to the corresponding index 20 in S_5 . The next index is 10 in S_{26} . Therefore, Step 5 is executed. I'_i is set to the corresponding index 99 in S_{29} and a hit bit 0 is outputted. The eighth index 65 is in S_{30} . Therefore, Step 6 is executed. I'_i is set to 65 and a hit bit 1 is outputted. The ninth index 80 is in S_{27} . Therefore, Step 5 is executed. I'_i is set to the corresponding index 88 in S_{31} and a hit bit 0 is outputted.

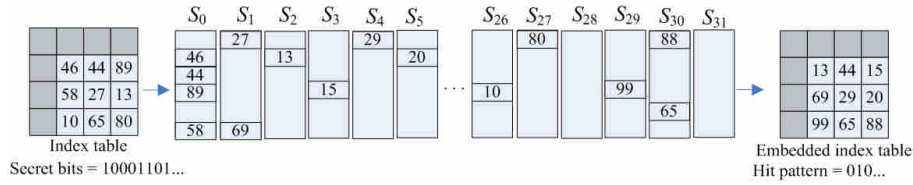


Fig. 6. An example of our embedding method.

The algorithm of the reversible extracting scheme is presented as the following steps:

Algorithm Extracting and Reversing

Input: Embedded index table I' , codebook C , hit pattern H

Output: Original index table I , extracted secret W

- Indexes belonging to the first row or the first column of I' do not participate in the extracting process. For each index I'_i in the first row or the first column, set $I_i = I'_i$.
- From left to right and from top to down, process each remained index I'_i by executing Step 3 ~ Step 9.
- For index I'_i , construct state codebooks S_0, S_1, \dots, S_{m-1} from codebook C .
- If $I'_i \in S_0$, extract bits 00 to W and set $I_i = I'_i$.
- If $I'_i \in S_1$, extract bits 01 to W and set I_i to the corresponding index in S_0 .
- If $I'_i \in S_2$, extract bits 10 to W and set I_i to the corresponding index in S_0 .
- If $I'_i \in S_3$, extract bits 11 to W and set I_i to the corresponding index in S_0 .
- If $I'_i \in S_j$ and $j = 4, \dots, m-4$, set I_i to the corresponding index in S_{j-3} .
- If $I'_i \in S_{m-3}, S_{m-2}, S_{m-1}$ and the to-be-processed hit bit in H is 0, set I_i to the corresponding index in $S_{m-6}, S_{m-5}, S_{m-4}$; if $I'_i \in S_{m-3}, S_{m-2}, S_{m-1}$ and the to-be-processed hit bit in H is 1, set $I_i = I'_i$.

Figure 7 shows an extracting and reversing example, where the embedded index table is created from Fig. 6. Similarly, indexes located in the first row or the first column of the embedded index table remain unchanged. The other indexes are performed by executing Step 3 to Step 9. Each embedded index I'_i needs to construct its own state codebooks, which are the same to the state codebooks created by index I_i for the reason that indexes I'_i and I_i have the same left index and the same upper index. In Fig. 7, state codebooks are fixed as Fig. 6. The first embedded index I'_i is 13 and it belongs to S_2 . Therefore, Step 6 is executed. The secret bits 10 are extracted and I_i is

set to 46. Next, the embedded index I'_i is 44 and it belongs to S_0 . Therefore, Step 4 is executed. The secret bits 00 are extracted and I_i is set to 44. In the third index, $I'_i = 15$ belongs to S_3 . Therefore, Step 7 is executed. The secret bits 11 are extracted and I_i is set to 89. In the fourth index, $I'_i = 69$ is in S_1 . Therefore, Step 5 is executed. The secret bits 01 are extracted and I_i is set to 58. The next index is 29 in S_4 . Therefore, Step 8 is executed. I_i is set to the corresponding index 27 in S_1 . In the sixth index, $I'_i = 20$ is in S_5 . Therefore, Step 8 is executed. I_i is set to the corresponding index 13 in S_2 . The next index is 99 in S_{29} . Therefore, Step 9 is executed. Because the to-be-processed hit bit is 0, I_i is set to the corresponding index 10 in S_{26} . The eighth index 65 is in S_{30} . Therefore, Step 9 is executed. Now, the to-be-processed hit bit is 1. Therefore, I_i is set to 65. Next index is 88 in S_{30} . Therefore, Step 9 is executed. Because the to-be-processed hit bit is 0, I_i is set to the corresponding index 80 in S_{27} .

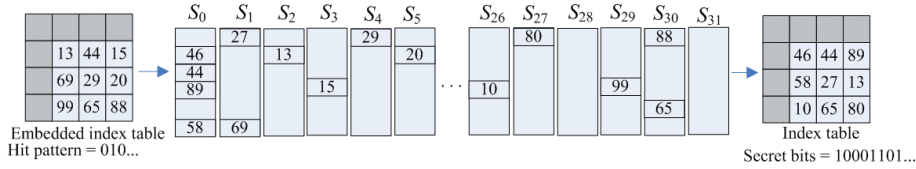


Fig. 7. An example of our extracting and reversing method.

3.3 Threshold-bounded hit patterns

Our hit pattern is different to Chang et al.'s hit map. In Chang et al.'s hit map, its size is fixed and equal to $|C|$, the number of codewords in codebook C . Twelve hit maps are used in their method. Therefore, the total size of their hit maps is $12 \times |C|$ bits. In our hit-pattern strategy, a hit bit is created only when X is in $S_{m-6}, S_{m-5}, \dots, S_{m-1}$, where its probability is very little.

In order to adjust our hit pattern size flexibly, a threshold TH is used and the transferring rule and hit-bit-generating rule shown in Fig. 4 are modified. For each block X , suppose that X 's closest codeword belongs to S_i and is denoted as cw_i . Let $cw_0, cw_1, \dots, cw_{m-1}$ be codewords in S_0, S_1, \dots, S_{m-1} , respectively, with the same position which equals to cw_i 's position in S_i . As shown in Fig. 8, all codewords are listed in order, including two additional dummy codewords cw_{-1} and cw_m . Let $DS(cw_i, cw_{i+3})$ mean the Euclidian distance between codewords cw_i and cw_{i+3} . If $DS(cw_i, cw_{i+3}) > TH$, $edge(cw_i, cw_{i+3})$ is denoted by a bold line. Otherwise, $edge(cw_i, cw_{i+3})$ is denoted by a thin line. Note that $edge(cw_{-1}, cw_0)$ is defined as a thin line and $edge(cw_{m-1}, cw_m)$ is defined as a bold line. Then, the new transferring rule and hit-bit-generating rule are shown below:

- (1) Block X can be transferred from S_i to S_{i+3} only if $DS(cw_i, cw_{i+3}) \leq TH$. Otherwise, the transfer is forbidden and X remains unchanged.
- (2) Let cw_j be the final encoded result of X . Note that cw_j could be cw_i or cw_{i+3} . Then, a hit bit is created when $DS(cw_j, cw_{j+3}) > TH$ and $DS(cw_{j-3}, cw_j) \leq TH$ as follows:
 - Case (a): $cw_j = cw_0$ and $DS(cw_0, cw_1), DS(cw_0, cw_2)$, or $DS(cw_0, cw_3) > TH$

If the to-be-embedded bit is 0, bit 0 is embedded and a hit bit 0 is created. If the to-be-embedded bit is 1, bit 1 is embedded and a hit bit 1 is created.

Case (b): $cw_j \neq cw_0$
The created hit bit is 1 or 0 according to $cw_j = cw_i$ or $cw_j = cw_{i+3}$, respectively.

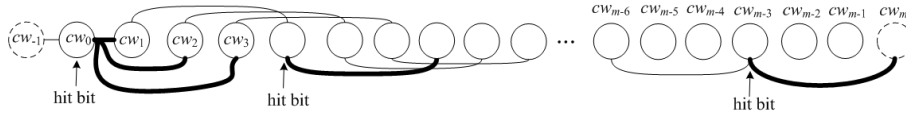


Fig. 8. The transferring rule and hit-bit-generating rule.

4 Experimental results

In this section, some experiments were performed on an Intel Core 2 T5500 1.66 GHz NB with 2 GB RAM and the Microsoft XP operating system. Our methods are implemented by C language. In the experiments, cover images are 512×512 gray images. Also, the secret is a random bit strings. The codebook has size 512 and is created by the *Linde-Buzo-Gray* (LBG) algorithm, and the size of a state codebook is 16. We used the peak signal-to-noise ratio (PSNR) to quantify the stego-image quality. Figure 9 shows the VQ cover images Lena, Sailboat, Baboon, and Peppers. In the experiments, we set a threshold (TH) to limit the transfer from a state codebook to its next state codebook. The bounded rules are shown in Section 3.3. Also, state codebook rows participating in the embedding and transferring operations are limited. A state codebook row means the codewords located in the same position of all state codebooks S_0, S_1, \dots, S_{m-1} . An active size s is used such that only the front s rows participate in the embedding and transferring operations. If block X is equal to a codeword of the $m - s$ hind rows, no embedding and no transferring is done.

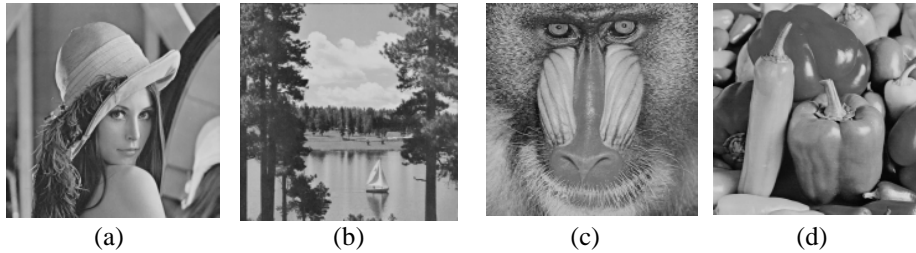


Fig. 9. Four VQ cover images. (a) Lena with a PSNR of 32.24 dB. (b) Sailboat with a PSNR of 29.25 dB. (c) Baboon with a PSNR of 24.70 dB. (d) Peppers with a PSNR of 31.40 dB.

Table 1 shows the results of our method with different active sizes and Chang et al.'s method [13]. When the PSNR values are similar, our hit-pattern sizes with active sizes is 2 and 4 are smaller than their hit-maps size and all our payloads are much higher than theirs. Chang et al.'s multiple hit maps need $512 \times 12 = 6144$ bits, where 512 is the codebook size and 12 is the number of hit maps. Note that secret data and hit maps cannot be embedded together in Chang et al.'s method. However, it is very possible to embed hit patterns successfully in our methods. In the experiment, only few hit-pattern bits cannot be embedded for images Baboon and Sailboat. The numbers shown in parentheses represent the numbers of un-embedded hit bits. For example, four bits of the 697 hit bits cannot be embedded for images Baboon when the active size is 2. When active sizes are larger, our payloads become larger. But, the hit patterns become larger too. When the active size is 8, our averaged hit-pattern size is close to Chang et al.'s hit-map size and our averaged payload is about 253% of theirs.

Table 1. Comparisons between our method with different active sizes and Chang et al.'s method when PSNR results are similar.

Methods	Measures	Lena	Baboon	Sailboat	Pepper	Average
Chang et al.'s method	Payload (bits)	8707	3400	7601	8421	7032
	PSNR (dB)	30.23	24.04	28.00	29.15	27.86
	Hit-maps size	6144	6144	6144	6144	6144
Active size = 2	Payload (bits)	10606	3058	10702	11248	8904
	PSNR (dB)	30.25	24.04	28.00	29.16	27.86
	Hit-pattern size	906	697(4)	3058(2)	1653	1579
	TH	205	240	185	235	216
Active size = 4	Payload (bits)	15882	5258	15166	16538	13211
	PSNR (dB)	30.24	24.05	28.04	29.20	27.88
	Hit-pattern size	4877	1940(1)	5394(1)	5314	4381
	TH	155	185	151	150	160
Active size = 8	Payload (bits)	21494	8574	19504	21850	17856
	PSNR (dB)	30.26	24.05	28.10	29.21	27.90
	Hit-pattern size	7646	3640(3)	7395(2)	7938	6655
	TH	130	150	137	125	136
Active size = 12	Payload (bits)	23900	10812	21760	24392	20216
	PSNR (dB)	30.27	24.04	28.04	29.18	27.88
	Hit-pattern size	8719	4887(5)	8327(3)	9039	7743
	TH	124	139	136	123	131
Active size = 16	Payload (bits)	25380	12516	23296	26004	21799
	PSNR (dB)	30.25	24.08	27.99	29.19	27.88
	Hit-pattern size	9342	5734(1)	9081(10)	9692	8462
	TH	122	128	135	121	127

Table 2 shows the results of our method and Yang et al.'s method [19] when the PSNR values are similar. The parameters of our method are that the active size is 16 and TH is 250. The results show that our averaged pure payload $21799 - 3497 = 18302$ is much higher than their averaged pure payload $10677 - 110 = 10567$, where the pure payload means the size of payload minus the size of the hit pattern.

Table 2. Comparisons between our method with active size = 16 and TH = 250 and Yang et al.'s method when PSNR results are similar.

Methods	Measures	Lena	Baboon	Sailboat	Pepper	Average
Yang et al.'s method	Payload (bits)	27.72	21.83	25.72	25.99	25.32
	PSNR (dB)	12665	6242	11621	12178	10677
	Hit-maps size	74	198	114	52	110
Our method	Payload (bits)	27.31	22.58	25.59	27.02	25.62
	PSNR (dB)	25380	12516	23296	26004	21799
	Hit-pattern size	3311	3358(3)	4192	3126	3497

Table 3 shows the run time of Chang et al.'s method, Yang et al.'s method and our method, where Yang et al.'s method and our method pre-construct the lookup table recording the distance between each pair of codewords. The time of pre-constructing the lookup table is about two seconds, which is included into the time shown in Table 3, in spite of the fact that lookup table can be created once then used for each embedding process. Our averaged run time is calculated from all cases with TH = 100, 150, 200, 250, and ∞ . The run-time results show that our method is faster than Chang et al.'s method and is little slower than Yang et al.'s method.

Table 3. The comparison of averaged run time.

	Chang et al.'s method	Yang et al.'s method with lookup table	Our proposed method with lookup table
Run time (seconds)	36.6	30.3	31.3

Figure 10 shows the relationships among the PSNR, threshold, and pure payload (numbers above the curves) of our proposed method with different active sizes. For a given active size, the payload of a test image is fixed. Larger TH values result in smaller hit-pattern sizes. Therefore, the pure payloads become larger. However, larger TH values result in lower PSNR values. Compared to Chang et al.'s averaged pure payload $7032 - 6144 = 888$, our method creates a lot of spaces for embedding secret data.

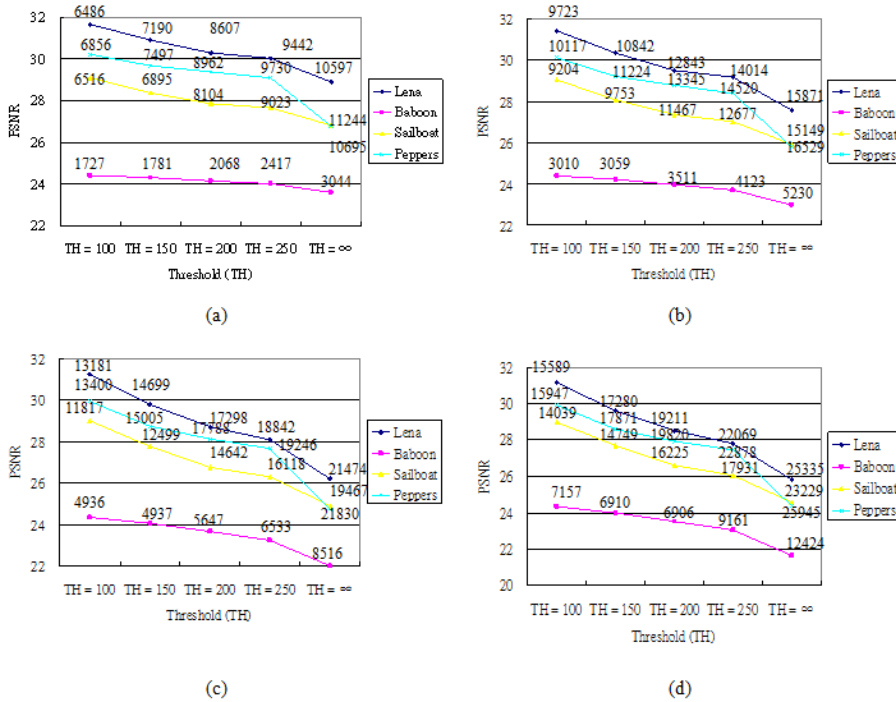


Fig. 10. Relationships among the PSNR values, thresholds, and pure payloads (numbers above the curves) of our proposed method with different active sizes: (a) size = 2, (b) size = 4, (c) size = 8, and (d) size = 16.

5 Conclusions

In this paper, we successfully extend our VQ reversible data hiding scheme to embed multiple bits into a block. The embedding capacity is largely increased by this method. Threshold-bounded hit patterns are proposed to adjust hit-pattern sizes and PSNR values. Compared to Chang et al.'s method, our proposed method has larger capacities and better PSNR values. Moreover, most of our hit patterns can be embedded together with the secret data.

Acknowledgments

This research was partially supported by the National Science Council of the Republic of China under the Grants NCS 97-2221-E-153-001.

References

1. C. Lin, Y. B. Lin and C. M. Wang, "Hiding data in spatial domain images with distortion tolerance," *Comput. Stand. Interfaces*, vol. 31, pp. 458-464 (2009)
2. C. H. Yang, C. Y. Weng, S. J. Wang and H. M. Sun, "Adaptive data hiding in edge areas of images with spatial lsb domain systems," *IEEE Transactions. Information Forensics and Security*, vol. 3, pp. 488-497 (2008)
3. C. C. Chang, W. C. Wu and Y. H. Chen, "Joint coding and embedding techniques for multimedia images," *Inf. Sci.*, vol. 178, pp. 3543-3556 (2008)
4. C. C. Lin, S. C. Chen and N. L. Hsueh, "Adaptive embedding techniques for VQ-compressed images," *Inf. Sci.*, vol. 179, pp. 140-149 (2009)
5. H. J. Kim, V. Sachnev, Y.Q. Shi, J. Nam and H. G. Choo, "A novel difference expansion transform for reversible data embedding," *IEEE Transactions. Information Forensics and Security*, vol. 3, pp. 456-465 (2008)
6. D. M. Thodi and J. J. Rodriguez, "Expansion embedding techniques for reversible watermarking," *IEEE Transactions. Image Processing*, vol. 16, pp. 721-730 (2007)
7. J. Tian, "Reversible data embedding using a difference expansion," *IEEE Transactions. Circuits and Systems for Video Technology*, vol. 13, pp. 890-896 (2003)
8. C. C. Lin and N. L. Hsueh, "A lossless data hiding scheme based on three-pixel block differences," *Pattern Recogn.*, vol. 41, pp. 1415-1425 (2008)
9. Z. Ni, Y. Q. Shi, N. Ansari and W. Su, "Reversible data hiding," *IEEE Transactions. Circuits and Systems for Video Technology*, vol. 16, pp. 354-362 (2006)
10. P. Tsai, Y. C. Hu and H. L. Yeh, "Reversible image hiding scheme using predictive coding and histogram shifting," *Signal Processing*, vol. In Press, Corrected Proof.
11. C. C. Chang, W. L. Tai and C. C. Lin, "A reversible data hiding scheme based on side match vector quantization," *IEEE Transactions. Circuits and Systems for Video Technology*, vol. 16, pp. 1301-1308 (2006)
12. C. C. Chang and W. C. Wu, "A reversible information hiding scheme based on vector quantization," in *Proceedings of Knowledge-Based Intelligent Information and Engineering Systems (KES 05)*, Melbourne, Australia, pp. 1101-1107 (2005)

13. C. C. Chang and C. Y. Lin, "Reversible steganography for VQ-compressed images using side matching and relocation," *IEEE Transactions. Information Forensics and Security*, vol. 1, pp. 493-501 (2006)
14. C. C. Chang, W. L. Tai and M. H. Lin, "A reversible data hiding scheme with modified side match vector quantization," *Proceedings of the 19th International Conference on Advanced Information Networking and Applications* , pp. 947-952 (2005)
15. C. C. Chang and C. Y. Lin, "Reversible steganographic method using SMVQ approach based on declustering," *Inf. Sci.*, vol. 177, pp. 1796-1805 (2007)
16. C. C. Chang, W. C. Wu and Y. C. Hu, "Lossless recovery of a VQ index table with embedded secret data," *J. Vis. Comun. Image Represent.*, vol. 18, pp. 207-216 (2007)
17. C. H. Yang, Y. C. Lin, S. C. Wu and M. H. Wu, "Reversible data hiding based on a sorted VQ index table," *Internal Computer Symposium* , pp. 456-461 (2008)
18. C. C. Chang, D. Kieu and Y. C. Chou, "Reversible information hiding for VQ indices based on locally adaptive coding," *Journal of Visual Communication and Image Representation*, vol. 20, pp. 57-64 (2009)
19. C. H. Yang, C. T. Huang, and S. J. Wang, "Reversible Steganography Based on Side Match and Hit Pattern for VQ-Compressed Images," to be presented in *The Fifth International Conference on Information Assurance and Security*, Xi'an, China, August 18-20 (2009)