A Reversible Data Hiding Method for SMVQ Indices Based on Improved Locally Adaptive Coding

Chin-Chen Chang[1], Jun-Yong Chen[1], Yan-Hong Chen[2], and Yanjun Liu[1]

(Corresponding author: Yanjun Liu)

Department of Information Engineering and Computer Science, Feng Chia University[1]

Taichung 40724, Taiwan

(Email: yjliu104@gmail.com)

School of Information, Zhejiang University of Finance and Economics[2]

Hangzhou 310018, China

(Received Jan. 31, 2018; Revised and Accepted Aug. 18, 2018; First Online July 16, 2019)

Abstract

In this paper, we propose a reversible data hiding scheme based on improved locally adaptive coding for compressed images by side match vector quantization (SMVQ). In the proposed scheme, an indicator is defined to encode the SMVQ indices and embed the secret data. The smaller the index is, the more bits of secret data could be hidden. In order to improve the capacity, we use an improved adaptive coding method after the assigned indices. Experimental results show that our proposed scheme outperforms state-of-the-art VQ-based data hiding schemes in terms of embedding capacity.

Keywords: Data Hiding; Image Compression; Locally Adaptive Coding; Reversibility; Side Match Vector Quantization (SMVQ)

1 Introduction

Due to the advance of Internet technology and the convenience of transmitting information, data hiding has become a popular research issue in recent years. Generally speaking, data hiding schemes can be classified into three types, *i.e.*, spatial domain, frequency domain and compression domain based schemes. Data hiding schemes in the spatial domain embed secret information by directly modifying pixel values of an image, while those in the frequency domain first transform an image from the spatial domain to the frequency domain and then modify coefficient values. However, the compressed domain based schemes embed data into the compressed codes of digital images [9]. Nowadays, many researchers have proposed various information hiding methods designed for the compression domain, such as block truncation coding (BTC) [12, 21, 22], discrete wavelet transform (DWT) [1, 2, 20], discrete cosine transform (DCT) [6, 14, 15] and vector quantization (VQ) [8, 10].

Wu and Sun [22] presented a data hiding method in which each secret bit is embedded into the bitmap of the BTC compression codes. Lin and Liu [12] proposed a data hiding method in BTC compressed images using the order of each pair of gray levels to embed the secret data. Wang *et al.* [21] presented a reversible data hiding scheme for images compressed by BTC based on the correlation among adjacent blocks and prediction-error expansion. Abdelwahab and Hassaan [1] proposed a data hiding method based on DWT that hides secret images inside the cover image using two secret keys. Vijay and VigneshKumar [20] proposed a method that first employs DWT to transform the cover image from the spatial domain to the frequency domain, and then compresses the secret data by the Huffman code for embedding. Baby *et al.* [2] proposed a DWT-based data hiding scheme using multiple color images to hide the secret data. Chang *et al.* [6] proposed a scheme to hide secret information in the DCT coefficients. In this scheme, the image is divided into 8×8 blocks. If two successive coefficients of the medium-frequency components are zero, the information is hidden in each block. Lin [15] proposed a method that embeds the data using the DCT coefficients and integer mapping. Lin [14] presented a method that uses histogram shifting to embed the secret data based on two-dimensional DCT coefficients.

VQ is another efficient image compression technique that was proposed by Gray [10]. In recent years, many researchers have studied data hiding based on VQ [3–5, 7, 8, 11, 13, 16–19, 23, 24]. In 2009, Chang *et al.* [5, 8] presented two reversible data hiding schemes based on VQ. They used joint neighbor coding to hide data [8] and employed a locally adaptive coding scheme (LAS) to reduce the cost of the compression code [5]. Also in 2009, Yang and Lin [23] proposed a VQ-based reversible information hiding method in which an indicator for data hiding is defined according to the embedding rule and the corresponding index. To improve the method in Chang and Chou's scheme [5], in 2010, Yang and Lin [24] presented a data hiding scheme that changes the hiding path to further enhance the effect of LAS. In 2011, Chang *et al.* [4] proposed an improved method that modifies the embedding rule to increase the embedding capacity of the method in [5]. In 2013, Pan *et al.* [17] presented another VQ-based method which uses search order coding (SOC) to compress the indices, and then utilizes the remainder of the space to embed the secret data. In 2014, Chang and Nguyen [7] introduced the concept of side match vector quantization (SMVQ) and proposed a novel data hiding scheme based on SMVQ that uses fewer indices than [17] when the same data are embedded. In 2015, Tu and Wang [19] proposed a method that divides all indices into two clusters for embedding. Their scheme can reduce extra bits and increase the hiding capacity. In 2016, Qin and Hu [18] proposed a method that uses an improved searching order coding (ISOC) encoded VQ index table. This method uses a lower bit rate and extends the degree of the index table to achieve higher hiding capacity.

However, the embedding capacities of the aforementioned data hiding schemes based on VQ are limited. Therefore, in this paper, we propose a VQ-based reversible data hiding scheme to enhance the embedding capacity. In the proposed scheme, we combine the SMVQ and LAS techniques to achieve a high embedding capacity.

The remainder of this paper is divided into five sections. Section 2 shows two techniques that our proposed reversible information hiding scheme is based on, *i.e.*, the SMVQ and LAS methods. Section 3 elaborates our proposed scheme. The detailed experimental description and comparative analysis are provided in Section 4, and our conclusions are presented in Section 5.

2 Related Work

In this section, we will introduce VQ, SMVQ and LAS techniques in Sections 2.1, 2.2 and 2.3, respectively.

2.1 VQ

VQ is a simple, lossy compression technology that is commonly used in reversible data hiding schemes [13]. The algorithm of VQ is divided into three phases. In the codebook generation phase, some images are used for training and to generate a codebook using the Linde-Buzo-Gary (LBG) algorithm, which was proposed by Linde *et al.* [16] in 1980. The LBG algorithm is an iterative procedure in which, first, it divides the image into a set of blocks with the size of $r \times r$, and each block can be viewed as an $r \times r-$ dimensional vector. Then, 256 blocks are selected randomly from these blocks as the initial codebook, with these 256 initial vectors as the 256 centroids. The rest of the blocks are grouped into the 256 centroids, *i.e.*, each block is used to find the nearest centroid to form 256 groups. Then, the centroids of these 256 groups are recalculated to get a new codebook until the codebook converges, *i.e.*, when the training of the codebook is complete. In the compression phase, the image is divided into many blocks of the same size containing many pixels. These blocks are used to search for similar codewords in the codebook, and the vector is replaced to form an index table. These procedures are explained by an example in Figure 1. In the decompression phase, the index values are used to extract the corresponding vectors from the codebook, and the vectors are used to form blocks that are then merged into an image.

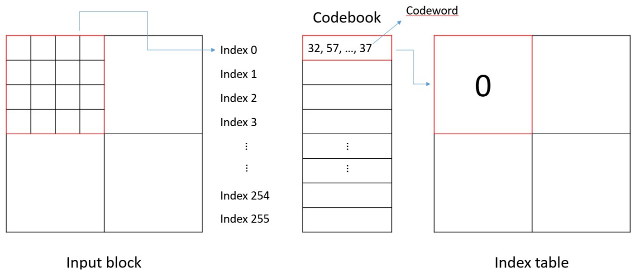


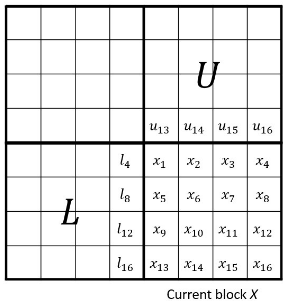
Figure 1: An example of VQ

2.2 SMVQ

As a variant of VQ, SMVQ was proposed by Kim [11] in 1992. In the encoding procedure, the algorithm divides the cover image into a set of non-overlapping blocks with the size of $c \times c$. These blocks are divided into two parts, *i.e.*, seed blocks and residual blocks. The image blocks in the first row and first column are denoted as seed blocks, and the rest of blocks are defined as residual blocks. As shown in Figure 2, the upper block U and the left block L are used to predict the current block X. After the prediction, let $x_1 = \frac{(u_{13}+l_4)}{2}, x_2 = u_{14}, x_3 = u_{15}, x_4 = u_{16}, x_5 = l_8, x_9 = l_{12}$, and $x_{13} = l_{16}$. Then, $x_1, x_2, x_3, x_4, x_5, x_9$ and x_{13} are used to find the most similar codeword from the traditional VQ codebook and a state codebook by using the most similar codewords is constructed. The codeword in the state codebook with the minimum Euclidean distance from X is used to encode X.

2.3 LAS

LAS is a data compression method that was proposed by Bentley *et al.* [3] in 1986. An example of the LAS is shown in Table 1. Suppose we want to compress the message "I HAVE A PEN I HAVE AN APPLE". Then, by using LAS, we can get the compressed message "1 I, 2 HAVE, 3 A, 4 PEN, 4, 4, 5 AN, 6 APPLE." The encoding steps are described as follows. First, list L is empty and is denoted as $L = \{\}$, and the size of list L is $S = 0$.



U
L
u_{13} u_{14} u_{15} u_{16}
l_4 l_8 l_{12} l_{16}
x_1 … x_{16}
Current block X

Figure 2: An illustration of SMVQ

We use the list L to encode the input word x. If the input word x doesn't belong to L, then we will use $S+1$ concatenated with x to encode it and x is inserted at the front of list L. Otherwise, the input word x is encoded by S, indicating the position of the input word x in the list L, and x is moved to the front of list L. The encoding process is repeated for the next word until all input words of the message are encoded.

Table 1: An example of LAS

Step	Input	The list $L = \{\}$	Output
1	I	$L = \{$I$\}$	1 I
2	HAVE	$L = \{$HAVE I$\}$	2 HAVE
3	A	$L = \{$A HAVE I$\}$	3 A
4	PEN	$L = \{$PEN A HAVE I$\}$	4 PEN
5	I	$L = \{$I PEN A HAVE$\}$	4
6	HAVE	$L = \{$HAVE I PEN A$\}$	4
7	AN	$L = \{$AN HAVE I PEN A$\}$	5 AN
8	APPLE	$L = \{$APPLE AN HAVE I PEN A$\}$	6 APPLE

3 Proposed Scheme

The proposed scheme is elaborated in this section. First, we propose an improved locally adaptive coding scheme (NILAS) by using the feature of SMVQ. Then, the embedding procedure and the extraction procedure are shown in Sub-sections 3.2 and 3.3, respectively.

3.1 NILAS

In the proposed NILAS, first we set the list L as not empty, and its length as equal to the largest value in the SMVQ indices. (For example, $L = \{0, 1, \ldots, N-1\}$, and N is equal to the largest value in the SMVQ indices. If the state codebook size is equal to 256, then the value of N is 256.) Based on the statistical analysis of different types of images, we found that the referred frequencies of all indices in SMVQ index table were close to the value in the range from zero to nine. Therefore, we divide L into two parts, *i.e.*, $L1$ and $L2$, where $L1 = \{0, 1, \ldots, 9\}$ and $L2 = \{10, 11, \ldots, N-1\}$. The order of $L1$ remains unchanged, and the move-to-front method is only applied in the order of $L2$. An example is provided in Figure 3.

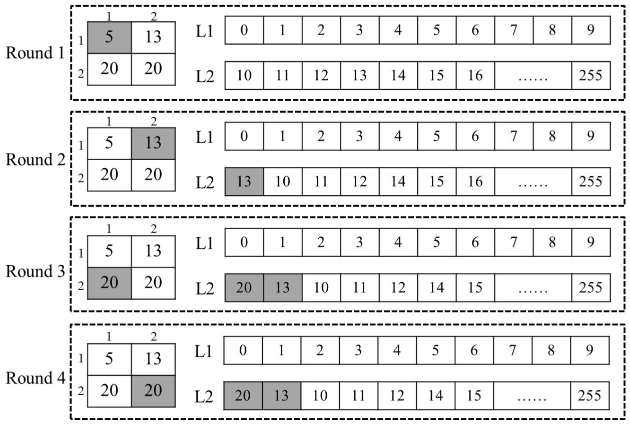


Figure 3: An example of NILAS

In Figure 3, we show a set of 4 blocks, and two lists $L1$ and $L2$. There are four rounds and each round processes a block. In particular, the gray block represents the current processing block. In the first round, the index value of the current block is 5, which belongs to $L1$. Thus, its order is unchanged. In the second round, the index value of the current block is 13. Since it belongs to $L2$, the move-to-front scheme is applied and it moves to the head of $L2$. Then, in the third round, the index value of the current block is 20. Since it belongs to $L2$, it also moves to the head of $L2$. Lastly, the index value of the current block is 20, which belongs to $L2$. Because its order is in the front of $L2$, the sequence will be unchanged.

3.2 Embedding Phase

Figure 4 shows the embedding flowchart of our method. I represents a cover image sized $H \times W$. Secret data S is a bit stream, and s_l represents the bit value in S, where $s_l = 1$ or $0, l = 0, 1, \ldots, M$, and M is the maximum embedding capacity in bits to the cover image I.

Input: A cover image I sized $H \times W$, codebook CB, and secret data S.

Output: The stego code stream CS.

Step 1: Encode I utilizing the SMVQ algorithm to obtain the SMVQ index table, IT. V_i is the value of each index in index table IT, where $i = 0, 1, \ldots, \frac{H}{4} \times \frac{W}{4} - 1$. Set the $L = \{0, 1, \ldots, N-1\}$ and divide L into two parts, $L1 = \{0, 1, \ldots, 9\}$ and $L2 = \{10, 11, \ldots, N-1\}$, where N is the size of codebook CB. Apply the NILAS method to $IT, L1$ and $L2$.

Step 2: Read the current V_i in IT. If the position of the current V_i is in the first row or the first column, i is increased by 1 and read the next V_i.

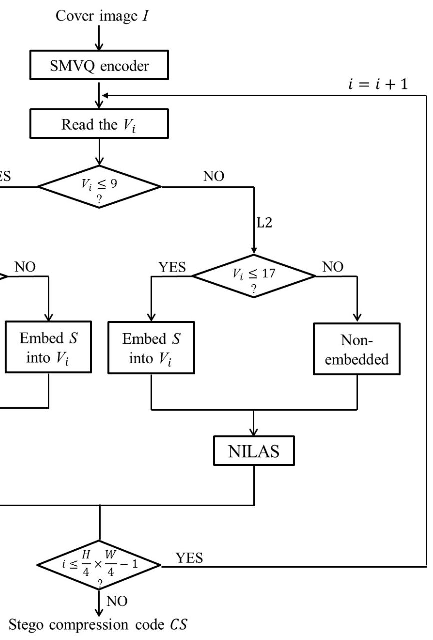


Figure 4: Flowchart of the embedding phase

Step 3: Check the value of V_i. If $V_i \le 1$, go to **Step 4**. If $2 \le V_i \le 9$, go to **Step 5**. Otherwise, go to **Step 6**.

Step 4: Add 00 to the head of the value of V_i as an indicator. That is, the first two bits are indicator bits and the third bit is the value of V_i. The remaining $n - 3$ bits are used to embed the next $s_{l,l+1,\ldots,l+(n-3)-1}$ from secret data S, where $n = \log_2\lfloor N \rfloor$. They will be encoded by $00\|V_i\|s_{l,l+1,\cdots,l+(n-3)-1}$, where $\|$ represents the concatenation operation. Go to **Step 8**.

Step 5: Add 01 to the front as an indicator. The next three bits are the value of $V_i - 2$ in binary form. And the remaining $n - 5$ bits are used to embed the next $s_{l,l+1,\ldots,l+(n-5)-1}$ from secret data S. They will be encoded by $01\|V_i\|s_{l,l+1,\ldots,l+(n-5)-1}$. Go to **Step 8**.

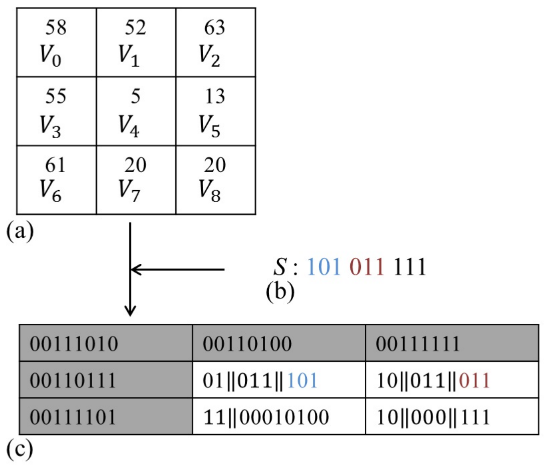
Step 6: Check the position p_i of current V_i in $L2$, where p_i is the position of V_i in $L2$. If $8 \le p_i \le N - 11$, go to **Step 7**. Otherwise, if $0 \le p_i \le 7$, add 10 to the front as an indicator. Then, use three bits to encode the position p_i in binary form. The next $n - 5$ bits are used to embed the next $s_{l,l+1,\ldots,l+(n-5)-1}$ from secret data S. They will be encoded by $10\|p_i\|s_{l,l+1,\ldots,l+(n-5)-1}$. Then, move V_i to the front of $L2$. Go to **Step 8**.

Step 7: Add 11 to the front as an indicator.. Then, encode the position p_i in binary form with n bits. They will be encoded by $11\|p_i$. Then, move V_i to the front of $L2$.

Step 8: Repeat **Steps 2** through **7** until all V_i's have been processed.

Step 9: Output the stego compression code CS.

After all of the steps have been completed, we get the stego compression code CS. To further clarify our embedding phase, an example of the embedding process is provided. In Figure 5(a), we assume that our SMVQ index table, IT, has the size of 3×3. Figures 5(b) and (c) show the secret data S that we want to embed in the SMVQ index table, IT, and the result after embedding, respectively.



58 V_0	52 V_1	63 V_2
55 V_3	5 V_4	13 V_5
61 V_6	20 V_7	20 V_8

(a)

S: 101 011 111

(b)

00111010	00110100	00111111
00110111	01‖011‖101	10‖011‖011
00111101	11‖00010100	10‖000‖111

(c)

Figure 5: An example of the data embedding phase: (a) SMVQ index table IT; (b) Secret data S; (c) Stego compression code CS

3.3 Extraction Phase

The extraction phase, as shown in Figure 6, is explained in this section. Because the indicator of two bits has been inserted during data embedding phase, a decoder can choose the corresponding extraction operation according to the indicator. If the first bit of the current processing stego index V_i', *i.e.*, the first bit of the indicator, is 0, the current processing stego index V_i' is in $L1$. Otherwise, it is in $L2$. Then, according to the second bit of indicator, it determines the extraction and recovery method. Moreover, if the current processing stego index V_i' is in $L2$, it must be moved to the front of $L2$ after extracting and recovery.

Input: Codebook CB and the stego compression code CS.

Output: The secret data S, reconstructed SMVQ index table IT, and cover image I.

Step 1: Check the position of current processing stego index V_i'. If the position is at the first row or the first column, i is increased by 1 and read the next current processing stego index V_i'.

Step 2: Read the first bit of current processing stego index V_i'. if it is 0, go to **Step 3.** Otherwise, go to **Step 5**.

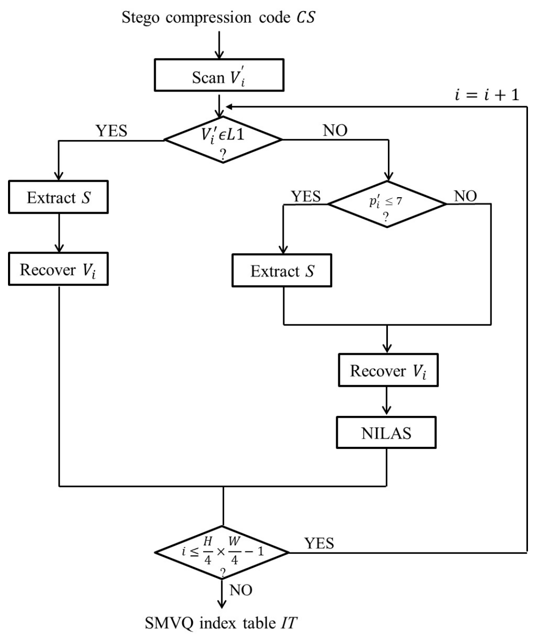


Figure 6: Flowchart of the data extraction phase

Step 3: Check the value of the second bit of current processing stego index V_i'. If it is 0, use the value of the third bit of the current processing stego index V_i' to recover the index V_i. Then, the remainder of $n - 3$ bits, *i.e.*, $s_{l,l+1,\ldots,l+(n-3)-1}$, will be extracted and appended to the secret data S. Go to **Step 7**. Otherwise, go to **Step 4**.

Step 4: Check the value of the second bit of current processing stego index V_i'. If it is 1, the next 3 bits are converted into decimal value to recover the index V_i. The remainder, $n-5$ bits, *i.e.*, $s_{l,l+1,\ldots,l+(n-5)-1}$, will be extracted and appended to the secret data S. Go to **Step 7**.

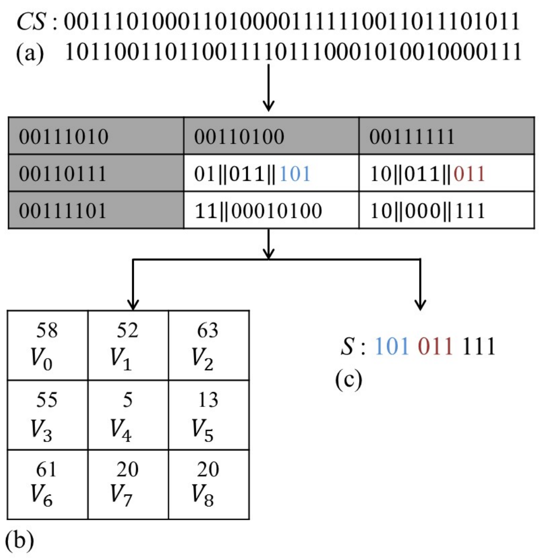
Step 5: Get the second bit of current processing stego index V_i'. If it is equal to 0, the next 3 bits will be transformed into decimal value p_i'. Use the corresponding value of p_i' in $L2$ to recover the index V_i. Extract the remaining $n - 5$ bits, $s_{l,l+1,\ldots,l+(n-5)-1}$, and insert them into secret data S. Then, move V_i to the front of $L2$. Go to **Step 7**. Otherwise, go to **Step 6**.

Step 6: Convert the remaining n bits into decimal value p_i'. Utilize p_i'' corresponding value in $L2$ to recover the index V_i and move V_i to the front of $L2$.

Step 7: Repeat **Steps 1** through **6** until all bits in the stego compression code have been read.

Step 8: Obtain secret data S and the SMVQ index table IT.

After all of the steps have been completed, we can obtain the secret data S and the reconstructed SMVQ index table IT. To further clarify our extraction and recovery phase, we present an example of the extracting process. Figures 7(a), (b) and (c) show the stego code stream, the reconstructed SMVQ index table, IT, and secret data S, that we can extract the reconstructed SMVQ index table IT and secret data S from the stego code stream, respectively.



CS : 0011101000110100001111110011011101011
(a) 1011001101100111101110001010010000111

00111010	00110100	00111111
00110111	01‖011‖101	10‖011‖011
00111101	11‖00010100	10‖000‖111

58 V_0	52 V_1	63 V_2
55 V_3	5 V_4	13 V_5
61 V_6	20 V_7	20 V_8

(b)

S : 101 011 111
(c)

Figure 7: An example of the data extraction phase: (a) Stego code stream; (b) Reconstructed SMVQ index table IT; (c) Secret data S

4 Simulation and Analysis

We chose the software MATLAB R2016 to conduct the experiments, and used a computer with an Intel-Core i7-6700 3.40GHz and 32GB RAM. The operating system was the 64-bit Windows 10 Pro. To verify the efficiency of the proposed scheme, several experiments are performed. Six 512 × 512-sized, 8-bit grayscale images were used as the test images. These images were "Lena", "Peppers", "F-16", "Toys", "Girl" and "Sailboat", as shown in Figure 8. These images were smooth images, which have been used extensively in image processing. Rough images would generate a negative value of payload, so we used the smooth images to keep the value positive, the bit rate lower, and the embedding rate higher. The images were partitioned into non-overlapping 4 × 4 blocks, and the codebook with a size of 256 was trained using the LBG algorithm. In our experiments, the secret data S were in binary form and the data were generated randomly by using a pseudo-random number generator and encrypted by conventional methods.

4.1 Experimental Results

In these experiments, the embedding efficiency was evaluated by the payload and the embedding rate, and the

Figure 8: Test images: (a) Lena; (b) Peppers; (c) F-16; (d) Toys; (e) Girl; (f) Sailboat

compression efficiency was evaluated by the bit rate. In order to show that our proposed scheme can hide more secret data than other methods, payload was defined as the number of secret bits that can be hidden when the bit rate is fixed at 0.5, *i.e.*, the larger the payload is, the better the result is, as shown in Equation (1). The bit rate indicates the compression rate of the cover image by using bits per pixel (bpp). So, the smaller the bit rate is, the greater the compression efficiency is, as defined in Equation (2). The embedding rate is a ratio of the size of embedded secret data to the size of code stream, and it denotes whether the size of the code stream is fixed. The higher the embedding rate is, the more secret bits can be embedded, as defined in Equation (3).

$$\begin{aligned} Payload &= Capacity - (0.5 - bit\,rate) \\ &\quad \times Size\,of\,original\,image. \quad (1) \\ Bit\,rate &= \frac{Size\,of\,code\,stream}{Size\,of\,original\,image}. \quad (2) \\ Embedding\,rate &= \frac{Size\,of\,embedded\,secret\,data}{Size\,of\,code\,stream} \\ &\quad \times 100\%. \quad (3) \end{aligned}$$

To investigate the superiority of our proposed scheme, we compared it with some VQ-compressed-based data hiding methods, and the compressed code was used to achieve compression and transformation [4,5,7,18,19,23, 24]. Because the capacities of other methods are not high, we proposed a high capacity method combined with SMVQ and NILAS, and, even though the bit rates were identical, our method still provided the highest capacity. Figure 9, Table 2, Table 3 and Figure 10 compare our proposed scheme's hiding capacity, payload, embedding rate and bit rate with those of Chang *et al.*'s schemes [4,5,7], Yang and Lin's schemes [23,24], Qin and Hu's scheme [18], and Tu and Wang's scheme [19].

As shown in Table 2 and Figure 9, our experimental results indicated that the capacity of our method was higher than that of all of the other methods. We achieved this

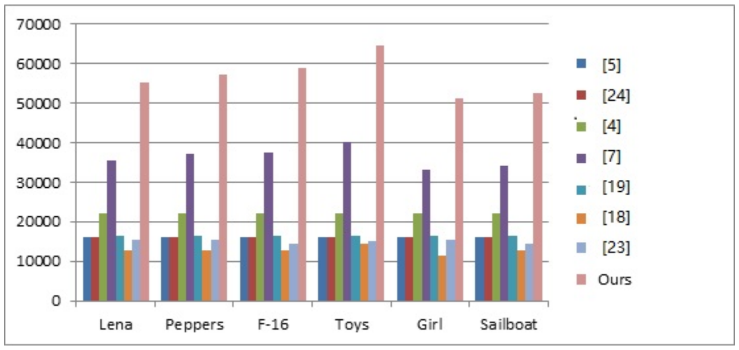


Figure 9: Comparisons of capacity (bits) between our method and other methods

result because we used the frequency of the occurrences of the SMVQ indices and clever application of the indices to hide more secret bits. The capacity of our method was still greater than that of other methods even under the same bit rate. The payload is defined that, when the value of bit rate is 0.5bpp, the number of secret data that can be hidden. When the sizes of code stream of all schemes are identical, we can get the value of payload according to the total number of the embedded secret bits. The experimental results showed that the capacity and payload of our scheme were much better than the other VQ-based methods.

Table 3 compares the embedding rate of our method and other methods. The higher the embedding rate was, the more secret bits could be hidden. It means that the embedding rate depends on the code stream. It can be seen that our proposed method is better than the other methods irrespective of how many code streams are embedded.

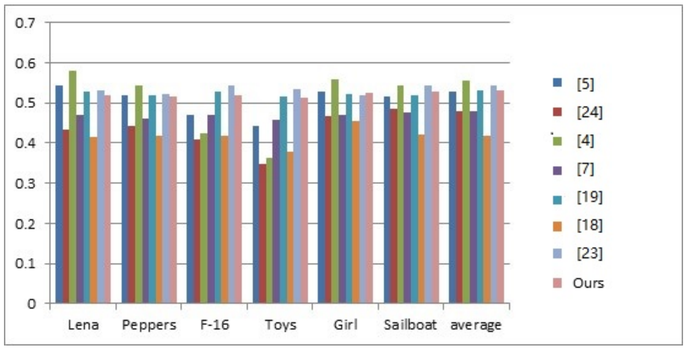


Figure 10: Comparisons of the bit rate (bpp) of our method with other methods

Figure 10 compares the bit rate of our method with other methods. According to Equation (2), we can know that the bit rate represents the compression rate. If the value of bit rate is low, the compression rate is better. Although compression rate of some methods are better than our method, the capacity of our method is better

Table 2: Comparison of the payload (bits) of our method and other methods

Image	[5]	[24]	[4]	[7]	[19]	[18]	[23]	Ours
Lena	4360	33982	850	43442	8738	34692	7521	49794
Peppers	11166	31089	10277	47700	11403	33940	9362	52614
F-16	23763	39754	42274	45421	9426	34555	3546	53786
Toys	31086	55728	58244	51191	11852	46770	6280	61025
Girl	9077	25068	6617	41147	10878	23449	10264	44906
Sailboat	11951	20339	10536	40632	11141	33684	2627	45215
average	8604	21740	7655	37404	8752	34558	2758	42244

Table 3: Comparison of the embedding rate (bpp) of our method and other methods

Image	[5]	[24]	[4]	[7]	[19]	[18]	[23]	Ours
Lena	0.113	0.143	0.145	0.289	0.118	0.116	0.111	0.404
Peppers	0.119	0.139	0.154	0.309	0.120	0.116	0.112	0.421
F-16	0.131	0.150	0.199	0.305	0.119	0.117	0.102	0.433
Toys	0.139	0.176	0.233	0.335	0.121	0.147	0.107	0.480
Girl	0.117	0.132	0.151	0.270	0.120	0.096	0.114	0.372
Sailboat	0.119	0.127	0.155	0.274	0.120	0.115	0.101	0.379
average	0.118	0.133	0.159	0.259	0.118	0.118	0.102	0.363

under the same bit rate. That is the reason why the payload is more important to represent the embedding effect of the data hiding scheme.

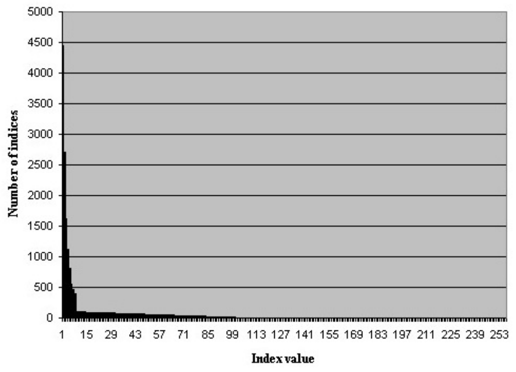


Figure 11: Histogram of the frequency of the occurrences of SMVQ indices

Figure 11 shows that we obtained the highest frequency of SMVQ index occurrences, *i.e.*, from 0 to 17. Therefore, we embedded most of the secret data in indices 0 through 9, and, after index 10, we used NILAS to improve the embedding capacity. Then, an indicator was defined and used to encode the indices and embed the secret data as much as possible in indices 0 or 1. That is to say, for indices 0 and 1, we can hide 5 bits; for indices 2 to 9 and 10 to 17, we can hide 3 bits. Then, in order to improve the embedding capacity after index 10, we used improved adaptive coding to increase the embedding capacity. This technology can use the feature of recurring occurrences of SMVQ indices to increase the embedding capacity. If the test image is smooth, we can hide more bits. Therefore, the experimental results showed that more secret data were embedded while the bit rate was the same, providing a higher capacity than schemes in the previous literature.

4.2 Discussions

In Chang and Wu's method [8], each index only can be embedded with one secret bit, but their method requires more bits to present the index in some cases. So, the payload of their method is lower than that of our method. In Chang and Chou's method [5], the largest capacity of each index is 1, and some indices cannot be embedded with secret bits if the index is not in the list L. Although their method can reduce the bit rate, the payload of their method is lower. Yang and Lin's method [24] improved the method in [5] by changing the run path of the normal VQ encoder. Although their bit rate is lower than Chang and Chou's method [5], their payload is still lower than that of our proposed method. The improved run path and the normal path are shown in Figure 12.

In Chang *et al.*'s method [4], although two secret bits can be embedded in some cases, it still requires more bits to represent the index. The largest capacity of their method is two, which is lower than that of our method. In Chang and Nguyen's method [7], the largest capacity of the indices is 3, while it is 5 for our proposed method. In their method, although the indices belong to the cases that can embed secret bits, the indices only need to use 7 bits to represent the index. But the payload of their method is lower than that of our proposed method. In Tu and Wang's method [19], although they used trained images to obtain a better distribution of codewords, cluster 2 and cluster 3 had to embed an extra 2 bits; so the payload of our proposed method was better than their method. In Qin and Hu's method [18], an improved search order coding-encoded VQ index table was used that could effectively embed secret data. However, their bit rate and payload were lower than those of our proposed method. In Yang and Lin's method [23], they resorted the VQ codebook according to the referred frequency of each index. , Their method can be mapped directly to the other clusters though the index is in the first cluster, thereby reducing the presented bits of the index. If the index is not in the first cluster, their method must use more bits to represent the index. The largest capacity of the indices in their method is 2, which is lower than that of our method. So our method is better than Yang and Lin's method [23]. However, our proposed method utilized only one bit to represent the indices and hided five secret bits most frequently, and three secret bits can be hidden combined with NILAS and SMVQ. Therefore, our proposed method achieved a higher capacity than any of the other methods while simultaneously providing a greater embedding rate and payload than the other methods.

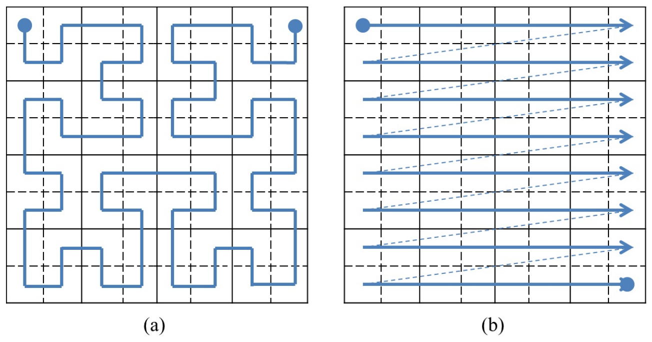


Figure 12: Run paths: (a) Improved path in Yang and Lin's method; (b) Normal VQ encoder path

5 Conclusions

In this paper, we proposed a reversible data hiding method for SMVQ indices. In the proposed method, using SMVQ indices can embed five bits or three bits at a time. In order to increase the embedding capacity, we combined SMVQ with improved locally adaptive coding. The experimental results showed that our proposed method effectively improved the capacity and outperformed state-of-the-art VQ-based data hiding schemes. In the future, we will focus on studying a simpler technique than locally adaptive coding to realize data hiding for compressed images while keeping high embedding capacity.

References

[1] A. A. Abdelwahab and L. A. Hassaan, "A discrete wavelet transform based technique for image data hiding," in *The 25th National Ratio Science Conference*, pp. 1–9, Jun. 2008.

[2] D. Baby, J. Thomas, G. Augustine, E. George, and N. R. Michael, "A novel DWT based image securing method using steganography," *Procedia Computer Science*, vol. 46, pp. 612–618, 2015.

[3] J. L. Bentley, D. D. Sleator, R. E. Tarjan, and V. K. Wei, "A locally adaptive data compression scheme," *Communications of the ACM*, vol. 29, no. 4, pp. 320–330, 1986.

[4] T. S. Nguyen C. C. Chang and C. C. Lin, "A reversible data hiding scheme for vq indices using locally adaptive coding," *Journal of Visual Communication and Image Representation*, vol. 22, no. 7, pp. 664–672, 2011.

[5] C. C. Chang and Y. C. Chou, "Reversible information hiding for vq indices based on locally adaptive coding," *Journal of Visual Communication and Image Representation*, vol. 20, no. 1, pp. 57–64, 2009.

[6] C. C. Chang, C. C. Lin, C. S. Tseng, and W. L. Tai, "Reversible hiding in DCT-based compressed images," *Information Sciences*, vol. 177, no. 13, pp. 2768–2786, 2007.

[7] C. C. Chang and T. S. Nguyen, "A reversible data hiding scheme for smvq indices," *Informatica*, vol. 25, no. 4, pp. 523–540, 2014.

[8] C. C. Chang and W. C. Wu, "A lossless data embedding technique by joint neighboring coding," *Pattern Recognition*, vol. 42, no. 7, pp. 1597–1603, 2009.

[9] I. C. Chang, Y. C. Hu, W. L. Chen, and C. C. Lo, "High capacity reversible data hiding scheme based on residual histogram shifting for block truncation coding," *Signal Processing*, vol. 108, pp. 376–388, 2015.

[10] R. M. Gray, *Vector Quantization*, Readings in Speech Recognition, 1990. (`https://en.wikipedia.org/wiki/Vector_quantization`)

[11] T. Kim, "Side match and overlap match vector quantizers for images," *IEEE Transactions on Image Processing*, vol. 1, no. 2, pp. 170–185, 1992.

[12] C. C. Lin and X. L. Liu, "A reversible data hiding scheme for block truncation compressions based on histogram modification," in *The 6th International Conference on Genetic and Evolutionary Computing*, pp. 157–160, Aug. 2012.

[13] C. C. Lin, X. L. Liu, and S. M. Yuan, "Reversible data hiding for vq-compressed images based on search-order coding and state-codebook mapping," *Information Sciences*, vol. 293, no. 1, pp. 314–326, 2015.

[14] Y. K. Lin, "High capacity reversible data hiding scheme based upon discrete cosine transformation," *Journal of Systems and Software*, vol. 85, no. 10, pp. 2395–2404, 2012.

[15] Y. K. Lin, "A data hiding scheme based upon dct coefficient modification," *Computer Standards & Interfaces*, vol. 36, no. 5, pp. 855–862, 2014.

[16] Y. Linde, A. Buzo, and R. Gray, "An algorithm for vector quantizer design," *IEEE Transactions on Communications*, vol. 28, no. 1, pp. 84–95, 1980.

[17] Z. Pan, X. Ma, X. Deng, and S. Hu, "Low bit-rate information hiding method based on search-order-coding technique," *The Journal of Systems and Software*, vol. 86, no. 11, pp. 2863–2869, 2013.

[18] C. Qin and Y. C. Hu, "Reversible data hiding in vq index table with lossless coding and adaptive switching mechanism," *Signal Processing*, vol. 129, pp. 48–55, 2016.

[19] T. Y. Tu and C. H. Wang, "Reversible data hiding with high payload based on referred frequency for VQ compressed codes index," *Signal Processing*, vol. 108, pp. 278–287, 2015.

[20] M. Vijay and V. VigneshKumar, "Image steganography algorithm based on huffman encoding and transform domain method," in *The 15th International Conference on Advanced Computing*, pp. 517–522, Oct. 2013.

[21] K. Wang, Y. Hu, and Z.M. Lu, "Reversible data hiding for block truncation coding compressed images based on prediction-error expansion," in *The 8th International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, pp. 317–320, Aug. 2012.

[22] X. Wu and W. Sun, "Data hiding in block truncation coding," in *International Conference on Computational Intelligence and Security*, pp. 406–410, Dec. 2010.

[23] C. H. Yang and Y. C. Lin, "Reversible data hiding of a vq index table based on referred counts," *Journal of Visual Communication and Image Representation*, vol. 20, no. 6, pp. 399–407, 2009.

[24] C. H. Yang and Y. C. Lin, "Fractal curves to improve the reversible data embedding for vq-indexes based on locally adaptive coding," *Journal of Visual Communication and Image Representation*, vol. 21, no. 4, pp. 334–342, 2010.

Biography

Chin-Chen Chang received his Ph.D. degree in computer engineering from National Chiao Tung University. His current title is Chair Professor in Department of Information Engineering and Computer Science, Feng Chia University, from February 2005. He is currently a Fellow of IEEE and a Fellow of IEE, UK. And, since his early years of career development, he consecutively won Outstanding Talent in Information Sciences of the R. O. C., AceR Dragon Award of the Ten Most Outstanding Talents, Outstanding Scholar Award of the R. O. C., Outstanding Engineering Professor Award of the R. O. C., Distinguished Research Awards of National Science Council of the R. O. C., Top Fifteen Scholars in Systems and Software Engineering of the Journal of Systems and Software, and so on. On numerous occasions, he was invited to serve as Visiting Professor, Chair Professor, Honorary Professor, Honorary Director, Honorary Chairman, Distinguished Alumnus, Distinguished Researcher, Research Fellow by universities and research institutes. His current research interests include database design, computer cryptography, image compression, and data structures.

Jun-Yong Chen received his B.S. degree in 2015, in Department of Computer Science and Information Engineering from Chung Hua University, Hsinchu, Taiwan. He is currently studying in Feng Chia University. His current research interests include image processing and data hiding.

Yan-Hong Chen was born in September 1980 and received his master degree in computer applications from Northeast Dianli University, China in March 2005. He is currently a lecturer at School of Information, Zhejiang University of Finance and Economics. His current research interests include data hiding, image retrieval, evolutionary algorithms etc.

Yanjun Liu received her Ph.D. degree in 2010, in School of Computer Science and Technology from University of Science and Technology of China (USTC), Hefei, China. She has been an assistant professor serving in Anhui University in China since 2010. She currently serves as a senior research fellow in Feng Chia University in Taiwan. Her specialties include E-Business security and electronic imaging techniques.