A Modification of VQ Index Table for Data Embedding and Lossless Indices Recovery

Zhi-Hui Wang[1,*], Chin-Chen Chang[2,3], Kuo-Nan Chen[3], and Ming-Chu Li[1]

[1] School of Software,

Dalian University of Technology,

Dalian, Liaoning, China

wangzhihui1017@yahoo.cn; li_mingchu@yahoo.com

[2] Department of Information Engineering and Computer Science,

Feng Chia University,

Taichung 407, Taiwan, ROC

alan3c@gmail.com

[3] Department of Computer Science and Information Engineering,

National Chung Cheng University,

Taichung 403, Taiwan, ROC

Kuonan.chen@gmail.com

Received 29 November 2009; Revised 30 December 2009; Accepted 4 January 2010

Abstract. A reversible data hiding scheme means that the original images can be losslessly recovered from the result with secret bits embedded. In this paper, a reversible data hiding scheme based on a vector quantization (VQ) index table is proposed. This paper aims to minimize the size expansion of the embedded result. To achieve this goal, the index appearance frequency histogram was analyzed before the embedding process started. The experiment results showed that the performance of the scheme proposed in this paper outperforms that of in Chang et al.'s scheme proposed in 2009 not only in bit rate, but also in hiding capacity.

Keywords: VQ index table, reversible, size expansion

1 Introduction

The rapid development of computer and networking technologies has changed the way of exchanging messages between each other. The type of messages has been transformed from real objects to digitalized data, and this transformation has the benefit of saving storage space. In addition, the time required to transmit messages has been greatly reduced since the digitalized messages are transmitted via the Internet. Although the new way of transmitting message provides the benefits of high transmission speeds and greatly reduced storage requirements, two serial problems associated with the technologies must be solved. The first problem is the security threat since the Internet is a public environment, and private messages can be intercepted by malicious attackers. The second problem is the utilization of networking bandwidth. The extensive use of high quality digital images could result in network paralysis.

For digital images, many research efforts [1, 2, 3, 4] have focused on hiding information in the original image to solve the security problem. In general, the secret information is encrypted with a private key in advance and embedded in the original images, or the secret information can be embedded in some particular areas of the original images. The receivers (or malicious attackers) cannot extract the embedded secret information unless they have the proper key or know exactly how it must be extracted. We used the famous least significant bit (LSB) substitution method to show how to embed the secret information in the original images. Assume the encrypted secret information is {0, 1, 0} in binary representation, and the to-be-embedded gray-level cover pixel, the original pixel, is 230, i.e., "11100110" in binary representation. The idea of LSB substitution is very simple in that it just replaces some bits of the cover pixel with the secret bits to accomplish the embedding procedure. The to-be-replaced cover bits are low-order cover bits since they create minimal distortion of the original image.

*Correspondence author

By applying the LSB substitution, the embedded result is “11100010” in binary representation (the last three cover bits “110” have been replaced with the three secret bits “010”).

On the other hand, many image compression methods [5, 6, 7] have been proposed to solve the network bandwidth problem. One of the famous image compression methods is vector quantization (VQ) proposed by Gray in 1984 [8]. The idea of the VQ compression is to use one-dimensional vectors to replace multi-dimensional vectors to reduce the file size. The VQ compression method requires that a codebook be trained initially. The codebook is the basis for transforming multi-dimensional vectors to one-dimensional vectors, and it can be produced by the LGB algorithm [9], one of the famous codebook training algorithms which is proposed by Linde, Buzo, and Gray in 1980. The LBG algorithm is introduced roughly as follows. First, a set of representative images sized $M \times N$ is collected and divided into $M \times N / m \times n$ blocks where each block is sized $m \times n$. To train a codebook sized *k*, *k* blocks are selected randomly from the block pool to be the initial centroid blocks. According to the *k* centroid blocks, all the blocks can be classified into *k* groups. Afterwards, *k* centroid blocks, the next generation, can be produced from the *k* groups. The two procedures, finding centroid blocks and classifying blocks into groups, are kept going until the *k* centroid blocks are stable. Finally, a codebook that contains *k* $m \times n$-dimensional vectors (codewords) is created. To compress an image, the to-be-compressed image is also divided into several blocks sized $m \times n$ first. The $m \times n$ pixels in one block can be treated as an $m \times n$-dimensional vector and compared with all codewords in the codebook to determine the closest one. The sequent number, called the index number, of this closest codeword is used to replace the currently processed block in the final result, called the VQ index table. The processes are shown in Figure 1.

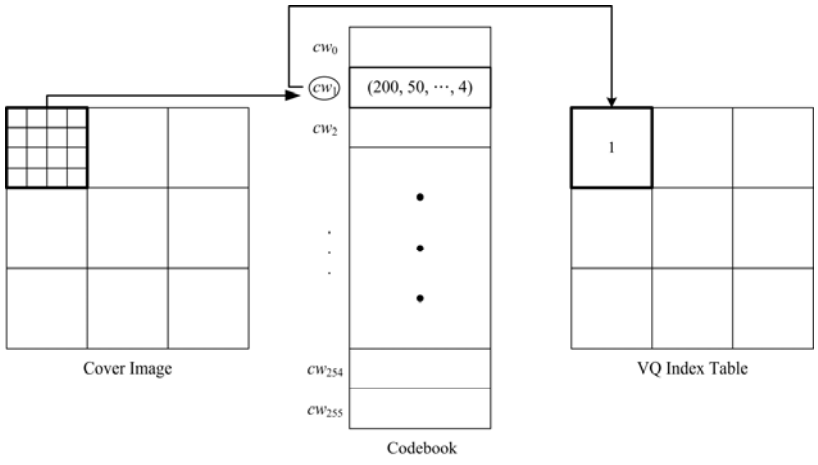


Fig. 1. The processes of the VQ-encoder

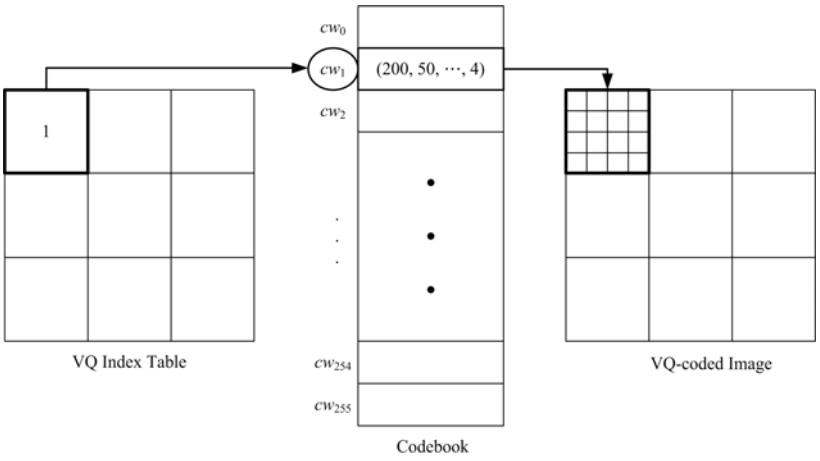


Fig. 2. The processes of the VQ-decoder

Once the receiver gets the VQ index table, the original VQ-coded image can be uncompressed according to the codebook he/she received. The processes of the VQ-decoder are introduced as follows. Each index number can be decoded to recover one $m \times n$ block using the corresponding $m \times n$-dimensional vector in the codebook.

The processes of the VQ-decoder are shown in Figure 2. After all indices in the VQ index table are processed, the VQ-coded image can be produced.

In this paper, we propose an efficient data hiding scheme based on VQ-coded images. The secret information is embedded in the VQ compression code. The receiver can extract the secret information from the final codestream, and the original VQ-coded image can be reversible.

This paper is organized as follows. In Section 2, the paper [10] proposed by Chang et al. in 2009 is reviewed. The details of the proposed scheme are illustrated in Section 3 followed by the experimental results in Section 4. Finally, some conclusions are given in Section 5.

2 Related Works

In this section, we reviewed the scheme proposed by Chang et al. [10] whose performance was compared with our proposed scheme in Section 4. First, the codewords in the codebook are rearranged by setting the first and least index as the two least frequently used indices in the VQ index table. After that, three indices are preserved as indicators and kept them from using in the VQ encoding procedure, the first index F, the last index L, and a randomly chosen index I from the remainder indices by using a pseudo-random number generator with a key K. While the new VQ index table is generated by the sorted VQ codebook, each two indices in the VQ index table are composed as a pair (i_1, i_2) for further analyzing. The embedding rules would fall into Case A if the number of pairs where $i_1 > i_2$ more than the number of pairs where $i_1 < i_2$. Otherwise, the embedding rules would fall into Case B. The embedding rules described in [10] are shown as follows.

```
Input: L, I, a pair of indices (i1, i2), and a secret bit s
Output: a watermarked index pair (j1, j2)

If (Case A)
    {If (i1 > i2)
        {If (s = 1)
            {(j1, j2) = (i1, i1 −i2);}
         Else if (s = 0)
            {(j1, j2) = (i1 −i2, i1);}}
     Else if (i1 = i2)
        {If (s = 1)
            {(j1, j2) = (i1, I);}
         Else if (s = 0)
            {(j1, j2) = (I, i1);}}
     Else if (i1 < i2)
        {If (s = 1)
            {( j1, j2) = (L, i1, i2);}
         Else if (s = 0)
            {(j1, j2) = (L, i2, i1);}}}
Else if (Case B)
    {If (i1 < i2)
        {If (s = 1)
            {( j1, j2) = (i1, i2 −i1);}
         Else if (s = 0)
            {( j1, j2) = (i2 −i1, i1);}}
     Else if (i1 = i2)
        {If (s = 1)
            {(j1, j2) = (i1, I);}
         Else if (s = 0)
            {(j1, j2) = (I, i1);}}
     Else if (i1 > i2)
        {If (s = 1)
            {(j1, j2) = (L, i1, i2);}
         Else if (s = 0)
            {(j1, j2) = (L, i2, i1);}}}
```

The secret bits extracting procedure and the VQ index table recovering procedure are shown in the following.

```
If (Case A)
    {If (j1 ≠ L)
        {If (j1 ≠ I and j2 ≠ I)
            {If (j1 > j2)
                {s = 1;
                 (i1, i2) = (j1, j1 - j2);}
             Else if (j1 < j2)
                {s = 0;
                 (i1, i2) = (j2, j2 – j1);}}
         Else if (j1 = I)
                {s = 0;
                 (i1, i2) = (j2, j2);}
         Else if (j2 = I)
                {s = 1;
                 (i1, i2) = (j1, j1);}}
     Else if (j1 = L)
        {If (j2 < j3)
            {s = 1;
             (i1, i2) = (j2, j3);}
         Else if (j2 > j3)
            {s = 0;
             (i1, i2) = (j3, j2); }}}
Else if (Case B)
    {If (j1 ≠ L)
        {If (j1 ≠ I and j2 ≠ I)
            {If (j1 < j2)
                {s = 1;
                 (i1, i2) = (j1, j1 - j2);}
              Else if (j1 > j2)
                {s = 0;
                 (i1, i2) = (j2, j2 – j1);}}
         Else if (j1 = I)
                {s = 0;
                 (i1, i2) = (j2, j2);}
         Else if (j2 = I)
                {s = 1;
                 (i1, i2) = (j1, j1);}}
     Else if (j1 = L)
        {If (j2 > j3)
            {s = 1;
             (i1, i2) = (j2, j3);}
         Else if(j2 < j3)
            {s = 0;
             (i1, i2) = (j3, j2); }}}
```

3 The Proposed Scheme

In this paper, a reversible data hiding scheme based on a VQ index table is proposed. The index value would be classified into two types, type 1 and type 2, before the embedding processes. Each type 1 index value can carry one secret bit without any extra information added. On the other hand, each type 2 index value can also carry some secret bits (the number of secret bits carried is decided by the size of codebook) by adding indicators in front of them. When the receiver gets the final codestream, codebook, and a lookup table, the secret bits can be exactly extracted, and the original VQ-coded image can be reversed. The two procedures in our proposed scheme, the embedding procedure and the extracting and recovering procedure, are described in Subsections 3.1 and 3.2, respectively.

3.1 The Embedding Procedure

The overview of the embedding procedure is shown in Figure 3.

Without loss of generality, assume the codebook size is n, where $\log_2 n$ is the multiples of two. Then the binary representation of index I in the VQ index table can be represented as $I = b_1 b_2 \ldots b_{\log_2 n}$. Furthermore, the bits in

I are cut in half, and let $F = b_1 b_2 \ldots b_{\log_2 n/2}$ and let $R = b_{(\log_2 n/2)+1} b_{(\log_2 n/2)+2} \ldots b_{\log_2 n}$ for further processing. In our proposed scheme, each index in the VQ index table would be classified into two types according to Equation (1).

$$\begin{cases} \text{If } F > R, \text{ then } I \in \text{type 1, and} \\ \text{if } F \le R, \text{ then } I \in \text{type 2.} \end{cases} \tag{1}$$

In our proposed scheme, the indices that belong to type 1 can carry one secret bit. The embedding rules for type 1 indices that are shown in Equation 2 assume the current processed secret bit is *s* and that the embedded result is *EI*.

$$\begin{cases} \text{If } s = 0, \text{ then } EI = I, \text{ and} \\ \text{if } s = 1, \text{ then } EI = R \,||\, F. \end{cases} \tag{2}$$

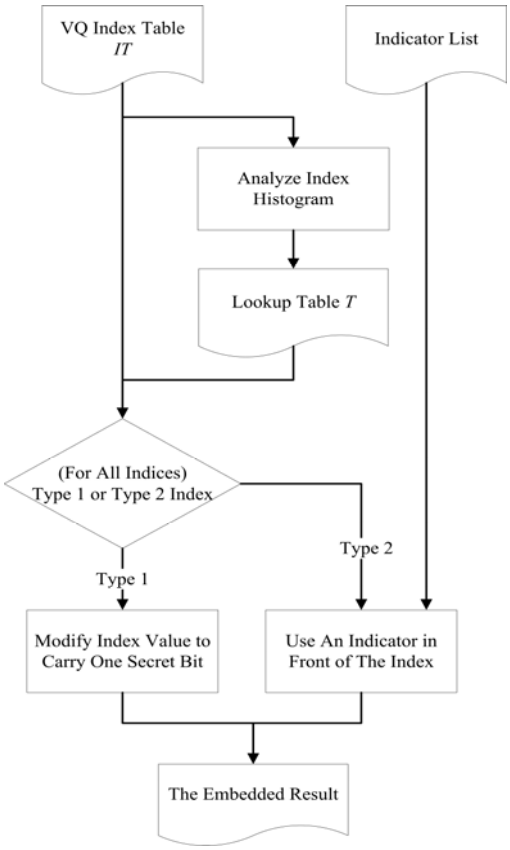


Fig. 3. The overview of the embedding procedure

On the other hand, each type 2 index can carry $(\log_2 n)/2$ bits by using an indicator D_j, where $j = 0, 1, \ldots, 2^{(\log_2 n)/2} - 1$, in front of it. An index is defined as an indicator that has the property of $F = R$. These $(\log_2 n)/2$ indicators are sorted and given sequence numbers to create an indicator list. For embedding $(\log_2 n)/2$ secret bits in a type 2 index, the indicator in the indicator list that has a sequence number that is equivalent to the secret bits is chosen.

As we introduced above, we know that the best case is to hide secret bits in type 1 indices (without size expansion). To increase the number of type 1 indices that we can embed, the index appearance histogram of the VQ index table was analyzed in order to generate a lookup table *T*. A one-to-one mapping is made between the high appearance frequency indices and all the possible type 1 indices ($F < R$) in the lookup table. Note that the way to judge whether an index belongs to type 1 or 2 changes. Now, only the high appearance indices appeared in the lookup table *T* are treated as type 1 indices, and the others are type 2 indices. Finally, the embedding procedure is concluded as follows:

Input: the VQ index table *IT*
Output: the secret bits embedded codestream
Step 1: Analyze the appearance frequency histogram of *IT* and create the lookup table *T*.
Step 2: Collect and sort the indices that have the property of $F = R$ to be the indicator list.

Step 3: Set a current processed index as a type 1 index if it can be found in *T*, and apply Equation 2 to embed a secret bit in it. Otherwise, set the current processed index as a type 2 index, and $(\log_2 n)/2$ secret bits can be embedded by using an indicator in front of it whose sequence number is equivalent to the to-be-embedded secret bits in the indicator list.

Step 4: Output the final codestream after all indices in *IT* are processed.

Example 1

Assuming that the size of codebook is 16, a segment of the VQ index table *IT* is shown in Figure 4; the index appearance histogram of *IT* is shown in Figure 5; the lookup table is shown in Figure 6; and the string of secret bits *s* is 0, 1, 0, 1, 1, 1, 0, 1, 0, 1.

6	10	7
8	3	6
9	6	8

Fig. 4. A segment of the VQ index table *T* in Example 1

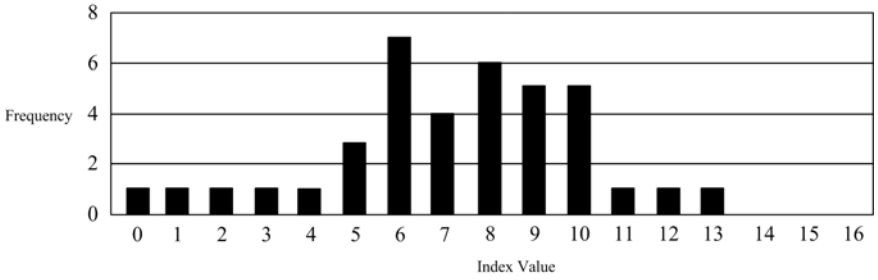


Fig. 5. The index appearance frequency histogram of *IT*

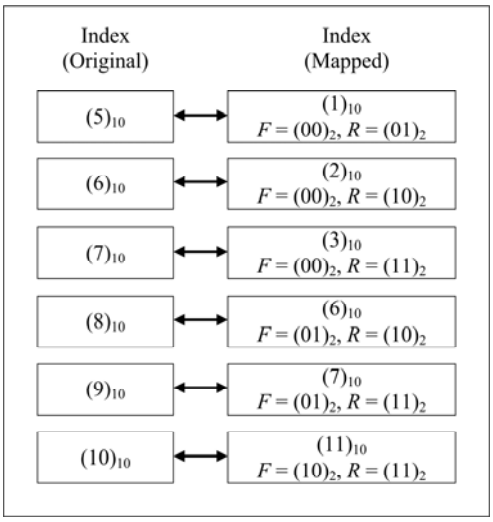


Fig. 6. The lookup table generated according to the index appearance frequency histogram in Example 1

If we process the segment VQ index table from left to right and top to bottom, the to-be-processed index sequence would be {6, 10, 7, 8, 3, 6, 9, 6, 8}. The first index, 6, belongs to type 1 and would be replaced with the mapped value, 2, found in the lookup up table that is shown in Figure 6. According to Equation (2), the embedded result would be equal to 2 if the corresponding secret bit is 0, and the embedded result would be equal to $(1000)_2 = (8)_{10}$ (i.e., $R \parallel F$) for carrying secret bit 1. Here, the corresponding secret bit for this index is 0, so, the embedded result is equal to 2. Next, for the second index, $(10)_{10}$, the mapped decimal value $(11)_{10}$ can be found in the lookup table. It means that this index also belongs to type 1 and would be replaced with the decimal value of 11 ($F = (10)_2$ and $R = (11)_2$) first. Since its corresponding secret bit to be carried equals 1, the embedded result is equal to decimal value 14 ($R \parallel F$) according to Equation (2). Similarly, the embedded results for the third

index $(7)_{10}$ and the fourth index $(8)_{10}$ are equal to $(3)_{10}$ and $(9)_{10}$ for carrying their own corresponding secret bits 0 and 1, respectively. The fifth index, $(3)_{10}$, is defined as type 2 index since it cannot be found in lookup table T. It means that it can carry the next two secret bits, $(11)_2$, by using the indicator $(15)_{10}$ in front of it according to the indicator list shown in Figure 7. The next four indices to be processed, $(6)_{10}$, $(9)_{10}$, $(6)_{10}$, and $(8)_{10}$ all can be found in the lookup table and defined as type 1 indices. The processes for them are similar to the process discussed above. Finally, the secret bits embedding results are shown in Figure 8.

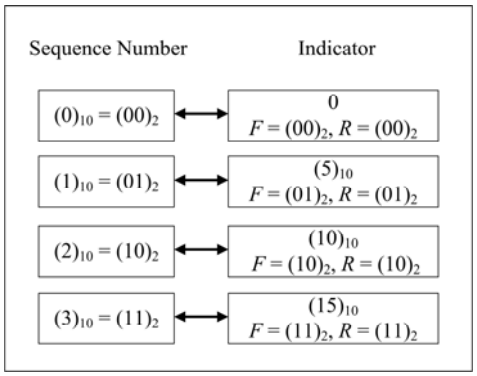


Fig. 7. The indicator list in Example 1

2	14	3
9	15 || 3	2
13	2	9

Fig. 8. The embedding results for Example 1

3.2 The Extracting and Recovering Procedure

The overview of the extracting and recovering procedure is shown in Figure 9, below:

The receiver can extract the secret bits from the embedded results and the original VQ index table can be recovered by the following processes. Note that the codebook size is n.

Input: the embedded result and the lookup table T

Output: the secret bits and the original VQ index table

Step 1: Decode the embedded result $\log_2 n$ by $\log_2 n$ bits. Name these current processed $\log_2 n$ bits as X.

Step 2: If X has the property that $F = R$, then it means that X is an indicator. The secret bits can be extracted according to the sequence number X in the indicator list, and the original VQ index can be recovered by the next $\log_2 n$ bits in the embedded result.

Step 3: If X has the property that $F < R$, the secret bit 0 can be extracted and the corresponding original index value can be recovered according to lookup table T by treating X as the mapped index.

Step 4: If X has the property that $F > R$, the secret bit 1 can be extracted, and the original index value can be recovered according to lookup table T where the mapped index is equal to $R \,||\, F$.

Step 5: After all bits in the embedded result are processed, all secret bits can be extracted, and the original VQ index table can be reversed.

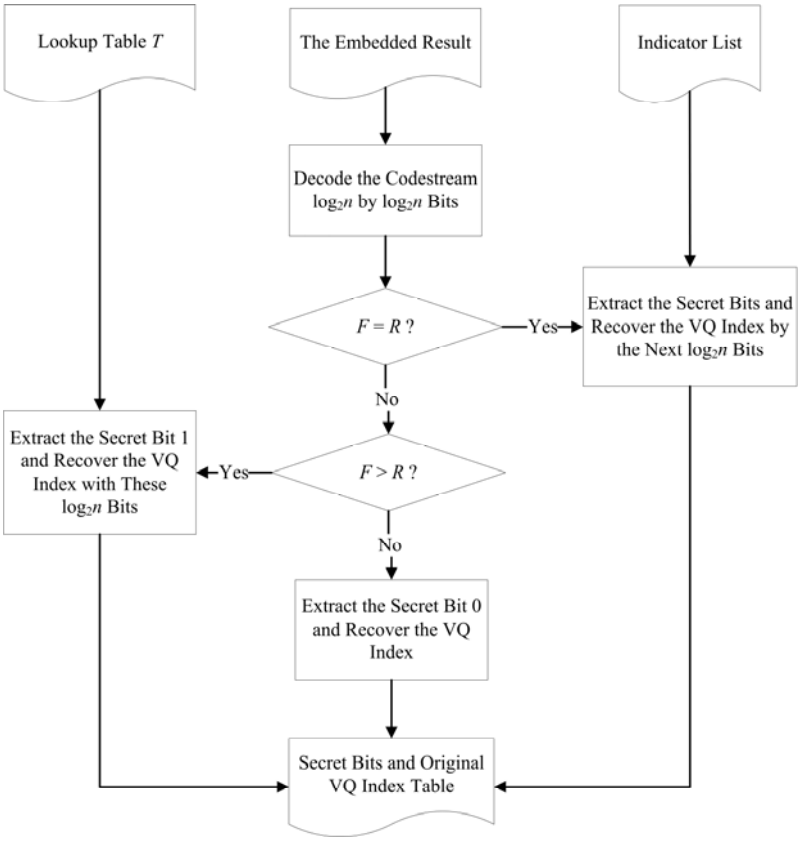


Fig. 9. Overview of the extracting and recovering procedure

Example 2

Followed by Example 1, lookup table T is shown in Figure 6, the indicator list is shown in Figure 7, and the embedded result is shown in Figure 8. Note that the indicator list can be created by the receiver. For the first four bits in the embedded result, its decimal value is 2 ($F = (00)_2$, $R = (10)_2$). Since it has the property of $F < R$, the secret bit 1 can be extracted, and the index value $(6)_{10}$ can be recovered according to lookup table T. The decimal value of the next four bits in the embedded result is equal to 14 ($F = (11)_2$, $R = (10)_2$). Since the currently processed bits have the property of $F > R$, the secret bit 1 can be extracted, and the mapped index value $(11)_{10}$ can be recovered by applying $R \| F$. Afterwards, the original index value $(10)_{10}$ can be reversed according to lookup table T. The original index values, $(7)_{10}$, and $(8)_{10}$, can be reversed by similar processing. For the next four bits in the embedding result, $(15)_{10}$, is an indicator since it has the property of $F = R$. The secret bits $(11)_2$ can extracted by checking the sequence number of this indicator in the indicator list, and the original index value $(3)_{10}$ can be reversed by the next four bits in the embedding result. Finally, after all bits in the embedded result are processed, the secret bit stream s is created by concatenating all the extracted secret bits, i.e., s = {0, 1, 0, 1, 1, 1, 0, 1, 0, 1}, and the original VQ index table is shown in Figure 10.

6	10	7
8	3	6
9	6	8

Fig. 10. The recovered VQ index table in Example 1

4 Experimental Results

Six test images, Jet, Pepper, Lena, Zelda, GoldHill, and Toys, all sized 512×512, were used in our experiment to show the performance of our proposed scheme, and they are displayed in Figure 11. The test images were compressed by using the VQ compression scheme to create the VQ index table. Two codebook sizes, 256 and 1024, were used in the experiment to show how the size of the codebook influences the bit rate and the data hiding

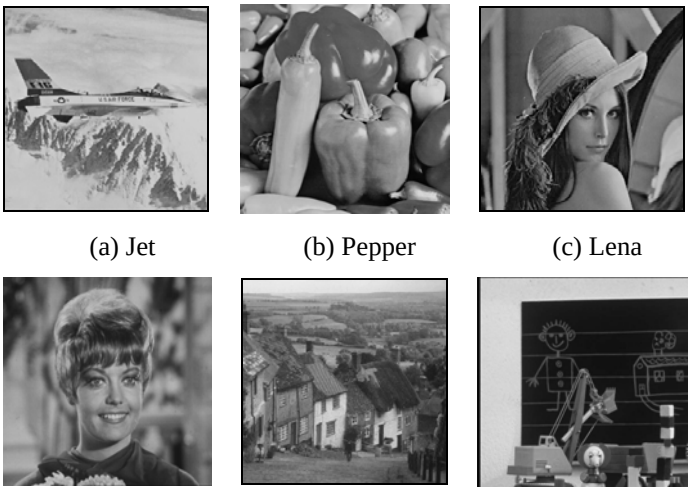
capacity in our proposed scheme. The VQ-coded images created by using the size 256 codebook are shown in Figure 12, and the VQ-coded images by using the size 1024 codebook are shown in Figure 13. Note that the evaluation tool for image quality we used was the peak signal-to-noise ratio (PSNR), and it is defined as follows:

$$\text{PSNR} = 10 \times \log_{10} \frac{255^2}{\text{MSE}},$$

where MSE = $(1/M \times N) \sum_{i=1}^{M \times N} (X_i - X'_i)^2$; M and N are the width and length of the image, respectively; and X_i and X'_i represent the pixel values of the original image and the VQ-coded image, respectively. In addition, the bit rate is defined as follows:

$$\text{bit rate} = \frac{C}{M \times N},$$

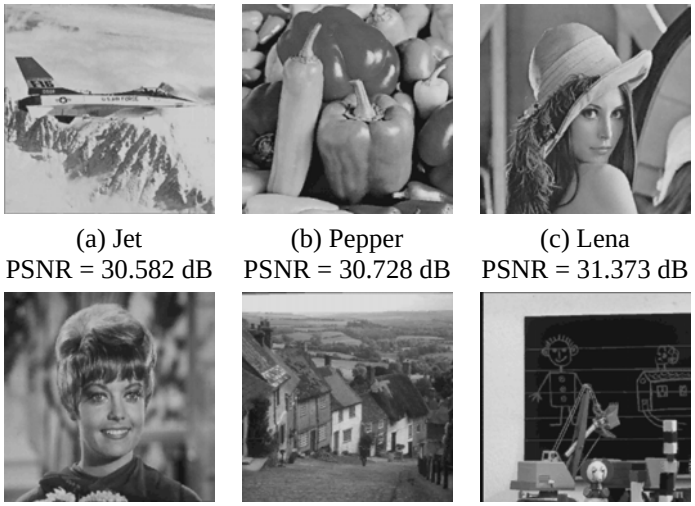
where C is the number of bits in the final codestream, and M×N is the number of pixels in the VQ-coded image. Note that the unit of bit rate is bits per pixel (bpp).



(a) Jet (b) Pepper (c) Lena

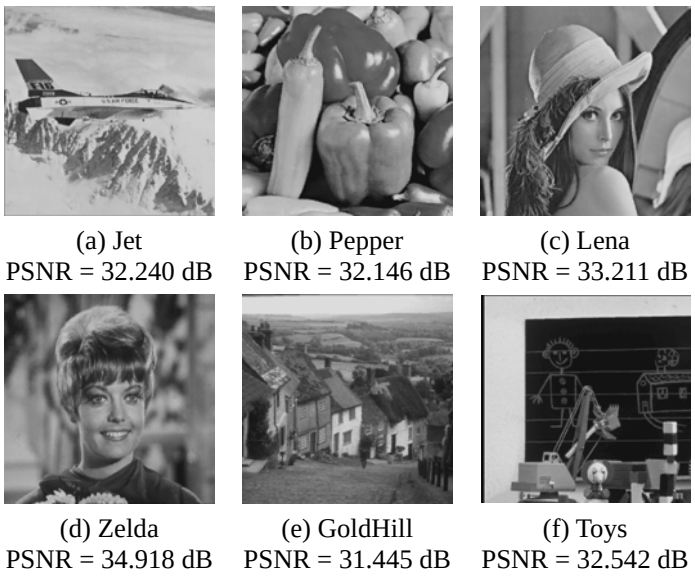
(d) Zelda (e) GoldHill (f) Toys

Fig. 11. The six test images



(a) Jet PSNR = 30.582 dB
(b) Pepper PSNR = 30.728 dB
(c) Lena PSNR = 31.373 dB
(d) Zelda PSNR = 33.375 dB
(e) GoldHill PSNR = 30.212 dB
(f) Toys PSNR = 29.920 dB

Fig. 12. The VQ images compressed by using the size 256 codebook



(a) Jet PSNR = 32.240 dB
(b) Pepper PSNR = 32.146 dB
(c) Lena PSNR = 33.211 dB
(d) Zelda PSNR = 34.918 dB
(e) GoldHill PSNR = 31.445 dB
(f) Toys PSNR = 32.542 dB

Fig. 13. The VQ images compressed by the size 1024 codebook

Table 1 shows the capacity, bit rate, and the number of type 1 indices of the six VQ-coded test images by applying our proposed scheme with the size 256 and 1024 codebooks, respectively. It can be observed from table 1 that if the user wants to have good performance in bit rate, he/she can use the codebook whose size is comparatively small. On the other hand, if the user wants to have higher image quality of VQ-coded image and hiding capacity, the codebook whose size is comparatively high can be used.

Table 1. Hiding capacity and bit rates of our proposed scheme (Codebook size = 256)

Images \ Codebook size		256	1024
Jet	Capacity	19804	18612
	Bit rate (bpp)	0.535	0.646
	No. of type 1 indices	15244	15827
Pepper	Capacity	21079	21908
	Bit rate (bpp)	0.548	0.678
	No. of type 1 indices	14819	15003
Lena	Capacity	21766	24204
	Bit rate (bpp)	0.555	0.700
	No. of type 1 indices	14590	14429
Zelda	Capacity	19096	19884
	Bit rate (bpp)	0.528	0.658
	No. of type 1 indices	15480	15509
Goldhill	Capacity	22720	23692
	Bit rate (bpp)	0.564	0.695
	No. of type 1 indices	14272	14557
Toys	Capacity	18439	18756
	Bit rate (bpp)	0.521	0.648
	No. of type 1 indices	15699	15791

Table 2 shows the comparisons between [10] and our proposed scheme. It is obvious that, irrespective of the values of hiding capacity or bit rate, the scheme proposed in this paper has a great improvement compared with [10].

Table 2. Comparison of [10] with our proposed scheme (codebook size = 256)

Scheme / Images	Chang et al.'s Scheme		The Proposed Scheme	
	Capacity	Bit rate (bpp)	Capacity	Bit rate (bpp)
Airplane	8192	0.574	19804	0.535
Pepper	8192	0.579	21079	0.548
Lena	8192	0.592	21766	0.555
Zelda	8192	0.599	19096	0.528
Goldhill	8192	0.597	22720	0.564
Toys	8192	0.552	18439	0.521
Averages	8192	0.582	20484	0.553

5 Conclusions

In this paper, a reversible data hiding scheme based on VQ index tables was proposed. The main purpose of this paper was to improve the bit rate with satisfactory hiding capacity. To minimize the size expansion of the final codestream, the index appearance frequency histogram was analyzed before embedding the secret bits to the VQ index table. To show the performance of our proposed scheme, we made a serious comparison with the scheme [10] proposed by Chang et al. The experimental results show that the scheme proposed in this paper outperforms [10] not only in bit rate, but also in hiding capacity. In the future, we will aim to improve the hiding capacity based on the satisfied bit rate proposed in this paper.

References

[1] C. C. Chang, T. D. Kieu, W. C. Wu, "A Lossless Data Embedding Technique by Joint Neighboring Coding," *Pattern Recognition*, Vol. 42, pp. 1597-1603, 2009.

[2] C. C. Chang, C. Y. Lin, Y. H. Fan, "Lossless Data Hiding for Color Images Based on Block Truncation Coding," *Pattern Recognition*, Vol. 41, No. 7, pp. 2347-2357, 2008.

[3] C. C. Chang, P. Y. Lin, J. S. Yeh, "Preserving Robustness and Removability for Digital Watermarks Using Subsampling and Difference Correlation," *Information Sciences*, Vol. 179, No. 13, pp. 2283-2293, 2009.

[4] C. F. Lee, K. N. Chen, C. C. Chang, "A New Data Hiding Strategy with Restricted Region Protection," *Imaging Science Journal*, Vol. 57, No. 5, pp. 235-249, 2009.

[5] A. B. Hussein, "A Novel Lossless Data Compression Scheme based on the Error Correcting Hamming Codes," *Computers & Mathematics with Applications*, Vol. 56, No. 1, pp. 143-150, 2008.

[6] A. Kingston and F. Autrusseau, "Lossless Image Compression via Predictive Coding of Discrete Radon Projections," *Signal Processing: Image Communication*, Vol. 23, No. 4, pp. 313-324, 2008.

[7] Y. Ma, H. Derksen, W. Hong, J. Wright, "Segmentation of Multivariate Mixed Data via Lossy Coding and Compression," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 29, No. 9, pp. 1546-1562, 2007.

[8] R. M. Gray, "Vector Quantization," *IEEE Transactions on Acoustics, Speech and Signal Processing*, pp. 4-29, 1984.

[9] Y. Linde, A. Buzo, R. M. Gray, "An Algorithm for Vector Quantizer Design," *IEEE Transactions on Communications*, Vol. 28, pp. 84-95, 1980.

[10] Z. H. Wang, K. N. Chen, C. C. Chang, M.C. Li, "Hiding Information in VQ Index Tables with Reversibility," *Proceedings of the Second International Workshop on Computer Science and Engineering*, Qingdao, China, pp. 1-6, 2009.