Journal of Information Hiding and Multimedia Signal Processing ©2014 ISSN 2073-4212
Ubiquitous International Volume **5**, Number **4**, October **2014**

High Efficiency Reversible Data Hiding for Two-stage Vector Quantization Compressed Images

Dong-Ning Zhao[1], Wei-Xin Xie[1], and Zhe-Ming Lu[1,2]*

[1]ATR National Defense Technology Key Laboratory, College of Information Engineering
Shenzhen University
Shenzhen, 518060, China
zhaodongning1979@gmail.com

[2]School of Aeronautics and Astronautics
Zhejiang University
Hangzhou, 310027, China
*Corresponding Author: zheminglu@zju.edu.cn

Received January, 2014; revised March, 2014

Abstract. *Data hiding is a booming technique aiming to embed secret data into digital media for content authentication, annotation or copyright protection. Reversible data hiding is a special type of data hiding technique that guarantees the host digital media can be losslessly recovered after the secret data are correctly extracted. In the past several years, some reversible data hiding schemes for Vector Quantization (VQ) compressed images have been proposed. Some of them embed data during the VQ encoding process, while others embed data during the VQ index streaming process. Compared with the traditional VQ, Multi-Stage Vector Quantization (MSVQ) is a more attractive data compression scheme with lower complexity and less encoding time due to the reduced codebook size. This paper proposes a novel reversible data hiding scheme based on two-stage VQ. The proposed framework consists of three phases, two-stage VQ codebook generation, path selection and data embedding, and cover image reconstruction and secret data extraction. Our main contribution lies in improving both the payload and transmission efficiency with a smaller codebook in comparison with traditional VQ-based reversible data hiding schemes. Experimental results validate the effectiveness of our scheme and demonstrate that it outperforms several previous methods in terms of payload, efficiency and stego-image quality.*

Keywords: Data hiding, Index coding, Multi-stage vector quantization, Reversible data hiding, Vector quantization

1. Introduction.

With the extensive applications of network technology, information transmission has become more and more convenient via some applications and protocols, such as E-mail, Network File System (NFS), Microsoft Service Network (MSN), Hypertext Transfer Protocol (HTTP), etc. However, this also introduces several information security issues such as copyright protection, owner announcement, covert communication, content authentication, etc. Steganography and cryptography are two kinds of powerful technologies to achieve the information security roles. Cryptography (e.g., DES [1] and RSA [2]) encrypts the content of messages while steganography hides the existence of the message. Data hiding is a powerful modern steganography technique to hide confidential messages in cover media such as images, audio, video, 3D meshes, etc. In fact, data can be hidden not only in audiovisual media but also in text documents [3]. By and large, data hiding schemes can be divided into two classes, namely, irreversible and reversible (or lossless) schemes [4]. The irreversible data hiding techniques can make the secret data imperceptible, but the distortions caused in the host signal are inevitable and irreversible. For some special areas (e.g., military, medical and forensic fields), it is crucial to recover the original host media without any distortion after the extraction of the secret data. This paper focuses on techniques for reversibly hiding information in images. Data hiding algorithms can be implemented in the spatial, transform or compressed domain. Least significant bit (LSB) replacement is typically used in spatial domain methods. Wang *et al.* [3] proposed a reversible hiding scheme to modify the LSB plane of the original image via multiple embedding strategies for achieving high-quality stego images. Chang *et al.* [5] employed a high dynamic programming strategy to find the optimal LSB substitution for data hiding. Mielikainen [6] proposed an improved LSB matching scheme to hide data more effectively in the spatial domain. Discrete Cosine Transform (DCT) and Discrete Wavelet Transform (DWT) are two commonly used transforms in reversible data hiding. Chang *et al.* [7] presented a reversible steganography scheme for hiding data in quantized DCT coefficients in JPEG images. Iwata *et al.* [8] embedded secret data by utilizing the boundaries between zero and non-zero DCT coefficients. Tian [9] proposed a wavelet transform based reversible data hiding method for authentication. With the development of data compression techniques, more and more digital images are stored in JPEG and JPEG2000 compressed formats or transmitted through VQ (Vector Quantization) or BTC (Block Truncation Coding) encoding schemes. Furthermore, in most circumstances, we cannot obtain the original image at all, therefore it is crucial to study and develop reversible data hiding techniques for compressed images. In this context, the reversibility means that the compressed image can be recovered after the extraction of hidden data. That is to say, not the original uncompressed image but its compressed version serves as the cover image. VQ [10] is a popular and efficient technique for signal compression and pattern clustering. Recently, promoted by the research works proposed by Chang *et al.* [11], Lu *et al.* [12] and Chen *et al.* [13], VQ has been widely applied to data hiding. These schemes can be roughly classified into two categories: data hiding during VQ encoding and data hiding during VQ index coding. For the first category, Yang *et al.* [14] proposed a reversible data hiding scheme based on modified fast correlation VQ (MFCVQ). Since the embeddable locations are determined by a pre-defined distance threshold, the hiding payload of the MFCVQ method is unstable and low. To improve the hiding payload, Chang *et al.* [15,16] proposed several reversible data hiding algorithms based on side-match VQ (SMVQ). The SMVQ algorithm can achieve a lower bit rate, however, it brings relatively high computation cost. Recently, several reversible data hiding schemes based on VQ index coding have been proposed. Chang *et al.* [17] presented a reversible hiding method based on a so-called joint neighboring coding (JNC) technique. This method embeds secret data by using the difference values between the current VQ-compressed index and left or upper neighboring indices. In 2009, Wang *et al.* [18] proposed a lossless data hiding scheme based on the improved JNC technique, which achieves higher stego-image quality and higher payload. However, the secret data is exposed in the output bitstream and hence it is not safe enough. To enhance the security, Lu *et al.* [12] proposed a reversible data hiding algorithm based on the VQ-index residual value coding. This algorithm can achieve higher PSNR and higher payload than the algorithms proposed by Yang *et al.* [14] and Chang *et al.* [15]. More recently, a novel path optional lossless data hiding method [19] has been proposed with improvements on payload, stego image quality, transmission efficiency and security. Usually, a reversible data hiding system aims at embedding more data in cover images effectively with low distortions introduced [20-21]. Actually, a good tradeoff must be achieved among these aspects. In this paper, we propose a new reversible data hiding scheme based on two-stage VQ (TSVQ), which enhances the payload and efficiency greatly. Meanwhile, an acceptable stego-image quality can be preserved. The rest of the paper is organized as follows. In Section 2, we first introduce the MSVQ compression technique, and then we introduce the related work, namely, Wang *et al.*'s scheme. Section 3 proposes our lossless data hiding scheme for TSVQ-compressed images. Section 4 provides the experimental results to demonstrate the superiority of our scheme to Wang *et al.*'s scheme and Lu et al.s scheme in terms of payload, efficiency and stego-image quality. Section 5 concludes the whole paper.

2. Related Work.

2.1. Multi-stage VQ (MSVQ). VQ is an attractive block-based encoding method for lossy image compression due to its high compression ratio and simple look-up-table decoding process. In general, VQ can be defined as a mapping from the k-dimensional Euclidean space R_k into a finite subset (codebook) $C = c_i | i = 0, 1, ..., N-1$, where c_i is a codeword and N is the codebook size. The codebook is often generated by the well-known LBG algorithm [22]. The LBG algorithm introduced by Linde, Buzo and Gray in 1980 is a generalization of the Lloyd-Max algorithm to derive a good codebook. It is similar to the k-means method in data clustering. With the codebook in hand, VQ first divides the input image into vectors, and then encodes each vector using its best-matched codeword in C. That is, for each k-dimensional input vector $x = x_1, x_2, ..., x_k$, the encoding rule can be given as follows.

$$d(\mathrm{x}, \mathrm{c}_i) = \min_{0 \le j \le N-1} (d(\mathrm{x}, \mathrm{c}_j)) \tag{1}$$

where c_i is the best-matched codeword and $d(x, c_i)$ denotes the distortion between the input vector and its codeword. In short, the VQ encoding process is to search the codeword with the smallest Euclidean distance for each input vector. Then the input vector is encoded with the index of its best-matched codeword.

The basic VQ system has a simple decoding structure but the encoding complexity is high. To release the complexity, multistage VQ is introduced to divide the encoding task into successive stages, where the first stage performs a relatively crude quantization of the input vector using a small codebook. Then, a second stage quantizer operates on the error vector between the original and quantized first stage output. The quantized error vector then provides a second approximation to the original input vector thereby leading to a refined or more accurate representation of the input. A third stage quantizer may then be used to quantize the second stage error to provide a further refinement and so on. Here, Each stage generates its codebook based on the corresponding training vectors in advance. As shown in Fig. 1, VQ$_i$ $(i = 1, 2, ..., p)$ represents the i-th VQ operation. Source $S^{(0)}$ denotes the input image and p is the total number of stages. $D^{(i)}$ $(i = 1, 2, ..., p-1)$ denotes the i-th version encoded image. $S^{(i)}$ $(i = 2, 3, ..., p-1)$ is the residue of $S^{(i)}$ subtracting $D^{(i)}$. $I^{(i)}$ $(i = 1, 2, ..., p)$ is the i-th output index table.

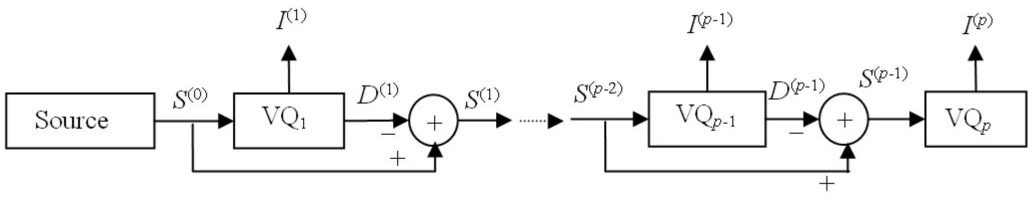

Figure 1. The flow chart of the multi-stage VQ

Specifically, this paper focuses on the two-stage VQ with two codebooks. Each stage generates its codebook based on the corresponding training vectors in advance. In the first stage, we encode the input image by the traditional VQ to generate the first stage index table with the first stage codebook. Then a residue image is produced by calculating the difference between the original image and the encoded version. The residue image is used as the input of the second stage VQ, and thus the second stage index table is generated with the second stage codebook.

2.2. **Path Optional Lossless Data Hiding.** Recently, Wang *et al.* has proposed a scheme named path optional lossless data hiding in [19], which can be divided into three phases, namely, the preprocessing phase, the data hiding phase, and the decoding and extraction phase. There are two main tasks during the preprocessing phase, namely, index table generation and M-sequence generation. For the index table generation task, a sorted VQ codebook is generated by the LBG algorithm, and then the cover image is encoded to be an index table using the obtained codebook. For the second task, the M-sequence is generated according to the selected path and an initial key. Here, M-sequence is a type of pseudorandom sequence, which is used to enhance the security of the embedded secret data. Wang et al.s scheme provides users with two optional paths as shown in Fig. 2. If the number of secret data is huge, Path 2 shown in Fig.2(b) is suggested to reach higher transmission efficiency. When the number of secret data is not

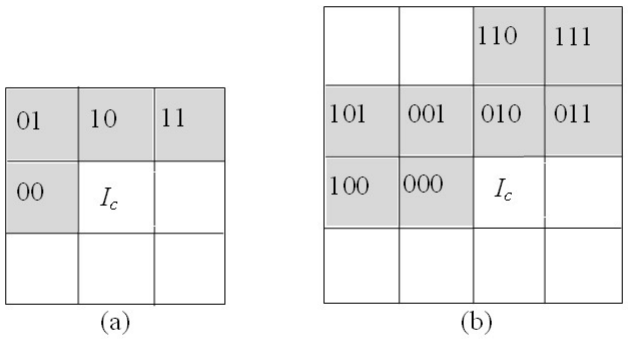

FIGURE 2. Two optional paths in the index table, (a) n=4, (b) n=8.

large, Path 1 shown in Fig.2(a) is proposed to reduce the length of the output bitstream. This task is performed by the M-sequence generator, where the length of the M-sequence is determined according to the selected path. If Path 1 is selected, two bits secret data are hidden in each block, and the number of bits in the output M-sequence is twice as large as the number of blocks in the cover image. Otherwise, if Path 2 is selected, three bits secret data are hidden in each block, and then the number of bits in the output M-sequence is three times as large as the number of blocks in the cover image. Assume we use the parameter n to denote the number of neighboring indices considered for encoding the current index, and n is an integer power of 2 (namely, n=2, 4, 8, ...). I_c stands for the current index to be processed, and $N(I_c)$ is a set of neighboring indices close to I_c. The locations of all indices in $N(I_c)$ are marked by binary codes from 00 to $(n-1)_2$, where $(n-1)_2$ means the binary code of n-1. As shown in Fig. 2, the search order is predetermined, which is composed of $N(I_c)$, the darker area around I_c. For n=4, $N(I_c)$ has four neighboring indices that can be used to encode I_c, as shown in Fig.2(a), and their location bits are"00", "01", "10" and "11", respectively. For n=8, $N(I_c)$ includes eight indices as shown in Fig. 2(b), with their location bits being "000", "001", "010", "011", "100", "101", "110" and "111", respectively. If the secret data size is large, we choose the path n=8. Otherwise, we choose the path n=4.

In the second phase, both the secret data and M-sequence are used for data hiding. The flow chart of Wang *et al.*'s hiding scheme is shown in Fig. 3. Here, b_i denotes the initial location bits of the reference index used to predict I_c, and it is got from the generated M-sequence. b_o denotes the offset bits of the reference index used to predict I_c, and it is got from the given secret data. b_r stands for the actual location bits of the reference index used to predict I_c, and obviously $b_r = b_i + b_o$ and the number of bits in b_i, b_o and b_r are all $\log_2 n$. The index corresponding to b_r is denoted as I_r, which is the reference index used to predict I_c. The difference between I_c and I_r is defined as $E = I_c - I_r$. In addition, we define the parameter $\lceil \log_2 N \rceil$ as the truncation integer that is adopted to divide the range of E, where N is the VQ codebook size and the function $\lceil a \rceil$ returns the least integer not less than a.

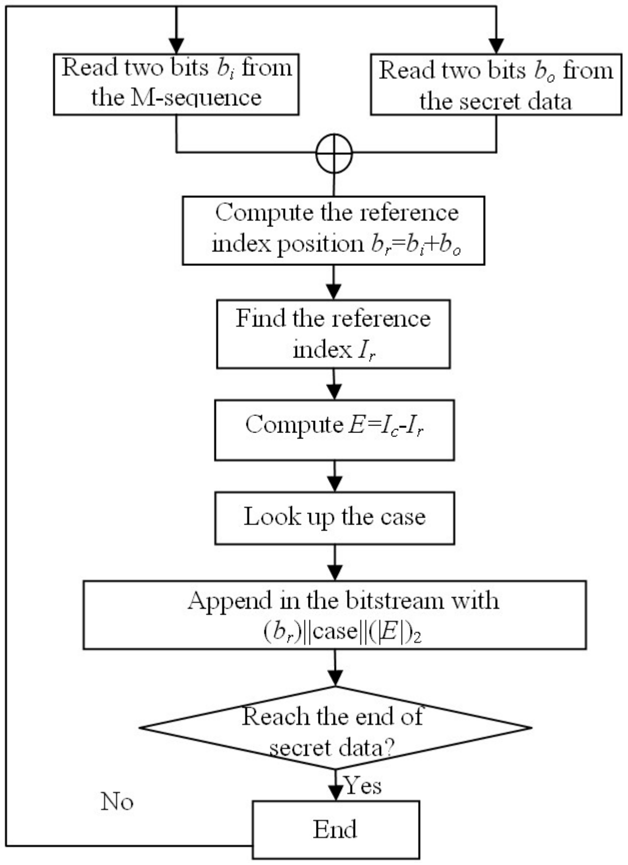
Read two bits b_i from the M-sequence
Read two bits b_o from the secret data
Compute the reference index position b_r=b_i+b_o
Find the reference index I_r
Compute E=I_c-I_r
Look up the case
Append in the bitstream with (b_r)||case||(|E|)$_2$
Reach the end of secret data?
Yes
No
End

FIGURE 3. The flow chart of the path optional lossless data hiding scheme (n=4).

For simplicity, we take n=4 as the selected path to illustrate the encoding process as shown in Fig. 3. Firstly, read two bits b_i from the M-sequence and two bits bo from the secret data respectively. Secondly, add them together to obtain b_r, and then determine I_r. Thirdly, calculate $E = I_c - I_r$. Here, the coding mode for E is categorized into four cases according to the parameter m. Fourthly, E is encoded as the binary form E' with four cases as follows.

$$E' = \begin{cases} (11)_2||(E)_2^{(m)} & \text{case“11”} \quad 0 \le E \le (2^m - 1) \\ (10)_2||(-E)_2^{(m)} & \text{case“10”} \quad -(2^m - 1) \le E < 0 \\ (01)_2||(E)_2^{(\lceil \log_2 N \rceil)} & \text{case“01”} \quad E > (2^m - 1) \\ (00)_2||(-E)_2^{(\lceil \log_2 N \rceil)} & \text{case“00”} \quad E < -(2^m - 1) \end{cases} \tag{2}$$

Where $A||B$ denotes the concatenation of A and B, and $(E)_2^{(m)}$ stands for the m bits binary code of E. Finally, the output bitstream of the current index can be encoded as “$(b_r)||\text{case}||(|E|)_2$”. Repeating these steps for every index outside the seed area (including the indices in the first row as well as in the first column of the index table), we can generate the final output bitstream. The decoding and extraction process is just the reverse process of data embedding, and readers can refer to Reference [19] for the detailed description.

3. **The Proposed Scheme.** The aim of this paper is to propose an efficient reversible data hiding scheme for two-stage VQ (TSVQ) compressed images. That is, the cover image in this paper is a kind of TSVQ compressed images. According to the two-stage codebook generation algorithm, two stage codebooks are obtained. Thus the cover image can be substituted by two index tables. To achieve a high payload, these two index tables are jointly combined in embedding to generate a more secure bitstream. The proposed scheme not only has feature that the encoding and decoding processes are reversible, but also the index tables and the secret data can be retrieved accurately. The proposed scheme is illustrated in Fig. 4, which can be divided into three stages, namely, the preprocessing stage, the data embedding stage, and the decoding and extraction stage. The proposed scheme can be expressed in detail as follows.

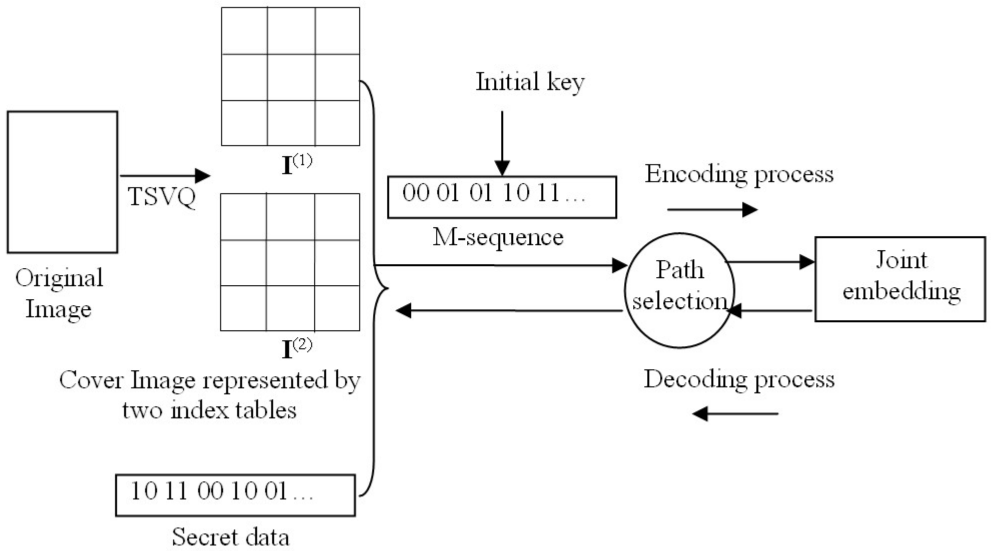

FIGURE 4. The flow chart of the proposed algorithm.

3.1. **Preprocessing Stage.** Before data hiding, several preprocessing steps should be performed. The aim of this step is to obtain the cover image to be embedded with secret data. Firstly, the two-stage codebooks, N_1-sized C_1 and N_2-sized C_2, with 4×4-dimensional codewords, are generated offline based on the two-stage VQ codebook generation algorithm. Secondly, the codewords in each VQ stage are sorted in the ascending order of their mean values in order to make neighboring codewords close to each other.

Thirdly, based on the sorted two-stage VQ codebooks, the $P \times Q$-sized original image is encoded as two $P/4 \times Q/4$-sized index matrices $I^{(s)}$ (s=1 or 2) for the first and second VQ stages respectively. With these two-stage index tables in hand, the secret data can then be hidden by encoding $I^{(s)}$ jointly to achieve a high payload and obtain a more secure bitstream. That is to say, in our paper, the cover image is represented by two index tables. The bitstream of the proposed algorithm consists of three parts including the head information HI, the bitstream CS_1 of the first stage and the bitstream CS_2 of the second stage. CS_1 and CS_2 are generated from the index tables $I^{(1)}$ and $I^{(2)}$ respectively. HI provides the basic information of the bitstream, such as the length of CS_1, the length of CS_2, the length of the secret data, the parameters m and n, and the cover image size.

3.2. **Proposed Data Embedding Process.** In this section, for the sake of convenience, we adopt the parameter set m, n, where the values of m and n have the same meaning as that in Section 2.2. Given the two-stage VQ codebooks, the terms in Section 2 should be changed to fit our scheme. The current indices to be encoded in the index tables $I^{(1)}$ and $I^{(2)}$ are denoted as I_{c1} and I_{c2} respectively. Similarly, the reference indices to predict I_{c1} and I_{c2} in the index tables $I^{(1)}$ and $I^{(2)}$ are denoted as I_{r1} and I_{r2} respectively. Since the initial location bits of the indices used to predict I_{c1} and I_{c2} are the same, the term b_i is not changed and it plays the same role as that in Section 2.2. However, b_o and b_r have to be changed into $b_o^{(s)}$ and $b_r^{(s)}$, respectively, where s=1 or 2. The differences between the current index and the reference index for $I^{(1)}$ and $I^{(2)}$ are denoted as E_1 and E_2, respectively.

Suppose an M-sequence is generated with an initial key in advance. Its length should be not less than twice the length of secret data, since the proposed scheme needs exactly twice the length as long as that of the secret data for encoding. During the embedding stage, the input are two index tables $I^{(1)}$ and $I^{(2)}$, the M-sequence and the secret data, while the output is the stego bitstream. The proposed data embedding process can be illustrated in detail as follows.

Step 1: Parameter selection.

Before embedding, m, n should be appropriately selected in advance. In this paper, we adopt two optional paths, namely, n=8 and n=4. Usually, the larger the n value is, the higher the payload is. If n=4, then the top-most row, the left-most and the right-most columns in the index tables $I^{(1)}$ and $I^{(2)}$ are viewed as the seed area and kept unchanged. If n=8, then the top-most two rows, the left-most and the right-most two columns in the index tables $I^{(1)}$ and $I^{(2)}$ are viewed as the seed area and kept unchanged. Suppose r is defined as the number of bits read from the M-sequence or secret data each time. Obviously, the r value can be computed as

$$r = \log_2 n \tag{3}$$

That is, r is set as 2 for n=4, while r=3 for n=8.

Step 2: Difference computation.

From this step on, we turn to encoding the current indices I_{c1} and I_{c1} in the index tables $I^{(1)}$ and $I^{(2)}$ respectively. Firstly, we get r bits from the M-sequence as the initial location bits b_i of the reference indices in the $I^{(1)}$ and $I^{(2)}$ respectively. Secondly, we get $2r$ bits from the secret data, where the first r bits adopted as the offset bits $b_o^{(1)}$ for $I^{(1)}$ and the second r bits $b_o^{(2)}$ for $I^{(2)}$. Thirdly, we compute $b_r^{(1)} = b_i + b_o^{(1)}$ for $I^{(1)}$ and $b_r^{(2)} = b_i + b_o^{(2)}$ for $I^{(2)}$ respectively. Fourthly, according to $b_r^{(1)}$ and $b_r^{(2)}$, we find the neighboring reference indices I_{r1} and I_{r2} in $I^{(1)}$ and $I^{(2)}$ respectively. Finally, we compute the differences $E_1 = I_{c1} - I_{r1}$ and $E_2 = I_{c2} - I_{r2}$ respectively.

Step 3: E_2 encoding.

Obviously, E_2 takes values in the interval $[-N_2+1, N_2-1]$, and there are four cases to encode E_2 into E_2' according to either Eq. (4) or Eq. (5). If $E_1 \neq 0$, we encode E_2 into

$$E_2{}' = \begin{cases} 11||(E_2)_2^{(m)} & \text{if} \quad 0 \le E_2 \le (2^m-1) \\ 10||(E_2)_2^{(m)} & \text{if} \quad -(2^m-1) \le E_2 < 0 \\ 01||(E_2)_2^{(\lceil \log_2 N_2 \rceil)} & \text{if} \quad E_2 > (2^m-1) \\ 00||(-E_2)_2^{(\lceil \log_2 N_2 \rceil)} & \text{if} \quad E_2 < -(2^m-1) \end{cases} \tag{4}$$

If E_1=0, we encode E_2 as

$$E_2{}' = \begin{cases} (11)_2 & \text{if} \quad 0 \le E_2 \le (2^m-1) \\ (10)_2 & \text{if} \quad -(2^m-1) \le E_2 < 0 \\ (01)_2 & \text{if} \quad E_2 > (2^m-1) \\ (00)_2 & \text{if} \quad E_2 < -(2^m-1) \end{cases} \tag{5}$$

Step 4: E_1 encoding.

In this paper, we consider five cases for E_1 encoding as follows.

Case 1: E_1=0. In this case, E_1 is not encoded in CS_1 while E_2 is encoded in CS_1. In the first stage of VQ, there is high correlation among the neighboring indices. Accordingly, it is reasonable that E_1=0 for most I_{c1} indices. This property can be employed to reduce the length of output bitstream. Here, we use three bits to encode this case as $(100)_2$. If E_2 belongs to its Case 1 or Case 2, three bits are used to represent the case of E1 and m bits are used to represent E_2. Thus, the binary stream is encoded as $100\|(|E_2|)_2$, where $|\bullet|$ denotes the absolute value of its element. If E_2 belongs to its Case 3 or Case 4, three bits are used to represent the case of E_1, and $\lceil \log_2 N_2 \rceil$ bits are used to represent E_2. Thus, the binary stream is encoded as $100\|(|E_2|)_2$. This case can be summarized as Eq. (6), where E_1' represents the encoded version of E_1, $(X)_2$ denotes the binary code of X, and $A\|B$ stands for the concatenation of A and B.

$$E_1{}' = \begin{cases} (100)_2||(E_2)_2^{(m)} & \text{if} \quad 0 \le E_2 \le (2^m-1) \\ (100)_2||(-E_2)_2^{(m)} & \text{if} \quad -(2^m-1) \le E_2 < 0 \\ (100)_2||(E_2)_2^{(\lceil \log_2 N_2 \rceil)} & \text{if} \quad E_2 > (2^m-1) \\ (100)_2||(-E_2)_2^{(\lceil \log_2 N_2 \rceil)} & \text{if} \quad E_2 < -(2^m-1) \end{cases} \tag{6}$$

Cases 2-5: For $E_1 \neq 0$, we can classify E_1 into four cases and encode E_1 as Eq. (7)

$$E_1{}' = \begin{cases} (011)_2||(E_1)_2^{(m)} & \text{if} \quad 0 < E_1 \le (2^m-1) \\ (010)_2||(-E_1)_2^{(m)} & \text{if} \quad -(2^m-1) \le E_1 < 0 \\ (001)_2||(E_1)_2^{(\lceil \log_2 N_1 \rceil)} & \text{if} \quad E_1 > (2^m-1) \\ (000)_2||(-E_1)_2^{(\lceil \log_2 N_1 \rceil)} & \text{if} \quad E_1 < -(2^m-1) \end{cases} \tag{7}$$

Step 5: CS_1 encoding.

To encode CS_1, we just append E_1' to $b_r^{(1)}$. For example, assume $b_r^{(1)} = (01)_2$, E_1=0 and the case of E_2 is $(11)_2$, then the final bitstream for the current index is $01\|000\|(|E_2|)_2$.

Step 6: CS_2 encoding.

To encode CS_2, we just append E2' to $b_r^{(2)}$. For example, assume $b_r^{(2)} = (00)_2$, E_1=0 and the case of E_2 is $(11)_2$, then the final bitstream for the current index is $00\|11$. If $E_1 \neq 0$, and the case of E_2 is $(11)_2$, then the final bitstream for the current index is $00\|11\|(|E_2|)_2$.

Step 7: If all secret data have been embedded or all of the indices outside the seed area have been processed, we append CS_1 and CS_2 to the head information HI to obtain the final bitstream. Otherwise, we go to step 2.

In sum, the final bitstream consists of three parts, namely, the head information of the stream, the first encoded index table (bitstream1, CS_1) and the second encoded index table (bitstream2, CS_2). The head information of the stream embodies the size of the original image, the length of the secret data, m, n, and the lengths of second and third parts sequentially. CS_1 is encoded as the binary form "$b_r^{(1)} \|$case$\|(|E_1|)_2$" or "$b_r^{(1)} \|$case$\|(|E_2|)_2$" depending on whether E_1 equals 0 or not. If $E_1 \neq 0$, CS_1 is encoded as the former, otherwise as the latter. CS_2 is encoded as the binary form "$b_r^{(2)} \|$case$\|(|E_2|)_2$" or "$b_r^{(2)} \|$case" depending on whether E_1 equals 0 or not. If $E_1 \neq 0$, CS_2 is encoded as the former, otherwise as the latter.

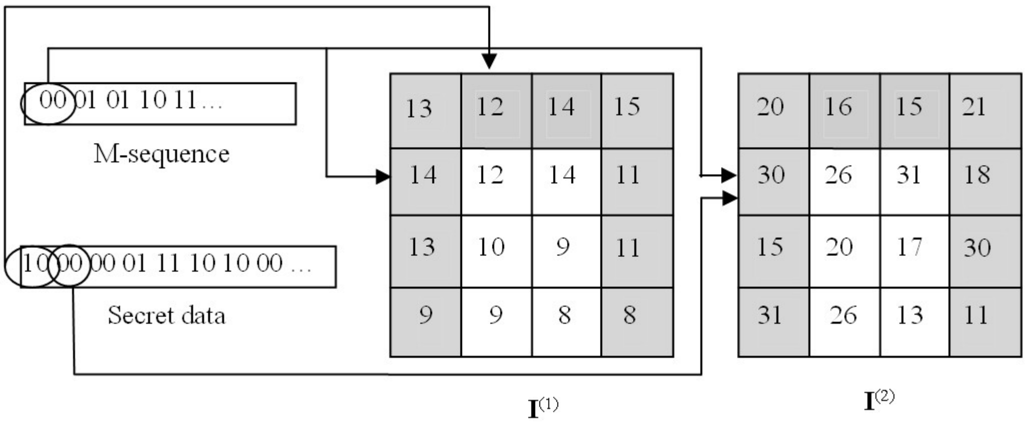

FIGURE 5. The flow chart of the proposed algorithm.

Here, we take a concrete example as shown in Fig. 5 to explain the proposed embedding scheme. Suppose N_1=16, N_2=32, n=4 and m=2. At each time, we read two bits from the M-sequence while obtain four bits from the secret data. In order to deal with the index I_{c1}="12" in the second row and the second column of $I^{(1)}$, the four neighbors $(14)_{00}$, $(13)_{01}$, $(12)_{10}$ and $(14)_{11}$ are taken into account. At the same time, the index I_{c2}="26" in the second row and the second column of $I^{(2)}$ is also processed. "00" is first read from the M-sequence as b_i, and then four bits "1000" are obtained from the secret data, where $b_o^{(1)}$= "10" and $b_o^{(2)}$="00". Then, we compute $b_r^{(1)}$ = "00" + "10" = "10" and $b_r^{(2)}$ = "00" + "00" = "00" and locate the reference indices I_{r1} = "12" and I_{r2}"30" respectively. Thus, we can compute E_1=12-12=0 and $E_2 = 26 - 30 = -4$. Since $E_1 = 0$, according to Eq.(5) and Eq.(6), CS_1 and CS_2 can be encoded as [10$\|$100$\|$00100 and 00$\|$00, respectively.

3.3. **Proposed Decoding and Extraction Process.**

Assume that the bitstream received by the decoder does not suffer any attack, then the original TSVQ compressed image can be completely recovered without the aid of the input M-sequence. Here, the input is the received bitstream, which can be divided into three parts, namely, HI, CS_1 and CS_2, while the output is the two TSVQ index tables. The decoding process can be illustrated in detail as follows.

Step 1: Obtain the head information HI from the bitstream to get the cover image size, the length of CS_1, the length of CS_2, and the values of n and m.

Step 2: Input r bits from CS_1 to obtain $b_r^{(1)}$, and input r bits from CS_2 to get $b_r^{(2)}$. Based on $b_r^{(1)}$ and $b_r^{(2)}$, we obtain their corresponding indices I_{r1} and I_{r2} respectively.

Step 3: Obtain the case information from CS_1 and CS_2, and get the difference values E_1 and E_2 from CS_1 and CS_2 based on the case information.

Step 4: Compute the current two-stage indices I_{c1} and I_{c2} as follows.

$$\begin{cases} I_{c1} = I_{r1} + E_1 \\ I_{c2} = I_{r2} + E_2 \end{cases} \tag{8}$$

Step 5: Perform Step 2 to Step 4 over and over again until two index tables are generated.

With regard to extracting the secret data, one input is the received bitstream, which can be divided into three parts, namely, HI, CS_1 and CS_2, and the other input is the key for generating the same M-sequence as used in the embedding process, while the output is the secret data. The same key as used in the embedding process is applied to generate the M-sequence, and the secret data can be obtained as follows.

Step 1: Read the head information HI from the received bitstream.

Step 2: Input r bits of the M-sequence to obtain the initial location bits b_i.

Step 3: Input r bits of CS_1 to get $b_r^{(1)}$, and input r bits of CS_2 to obtain $b_r^{(2)}$. Then retrieve the corresponding indices I_{r1} and I_{r2} respectively. Then, compute the offset bits $b_o^{(1)} = b_r^{(1)} - b_i$ and $b_o^{(2)} = b_r^{(2)} - b_i$. Append $b_o^{(1)}$ and $b_o^{(2)}$ to the extracted secret data.

Step 4: Perform Steps 2 and 3 over and over again until two index tables are generated. In this way, we can obtain the final secret data.

It should be noted that the complexity of the proposed decoding & extraction stage is very low, and the decoding process and the extraction process can be absolutely detached.

4. Experimental Results.

A set of experiments is carried out to evaluate the performance of our scheme. The performance indicators used include peak signal-to-noise ratio (PSNR), payload, bit rate (bit_rate) and transmission efficiency. A higher PSNR indicates a better image quality. The bit_rate means the average number of bits required for representing one pixel. Payload stands for the maximum number of secret bits that can be embedded in the cover image. Transmission efficiency is defined as the ratio of payload to the bitstream length. For $P \times Q$-sized 256 gray-level images, the PSNR between two images can be calculated as

$$\text{PSNR} = 10\log_{10}\frac{255 \times 255}{\text{MSE}} \tag{9}$$

where the mean squared error (MSE) can be calculated as

$$\text{MSE} = \frac{1}{P \times Q}\sum_{i=1}^{P}\sum_{j=1}^{Q}(x_{i,j} - x'_{i,j})^2 \tag{10}$$

where $x_{i,j}$ is the original uncompressed pixel value located at (i, j), and $x'_{i,j}$ is the stego pixel value located at (i, j). In addition, the bit rate is defined as

$$\text{bit\_rate} = \frac{L}{P \times Q} \tag{11}$$

and the efficiency is defined as

$$\text{Efficiency} = \frac{\text{Payload}}{L} \tag{12}$$

where L denoted the bitstream length.

4.1. **Performance of Our Algorithm.** In order to test the average performance of our method, six 512 × 512-sized 256-graylevel images (namely, Lena, F16, Peppers, Baboon, Boat and Goldhill) are adopted as test images. The codebook size is denoted as $N = (N_1, N_2)$, where N_1 and N_2 stand for the codebook sizes of the first and the second stages, respectively. We adopt three codebooks with sizes (16, 32), (32, 64) and (64, 128) respectively. In order to show the reversibility of our scheme, we use PSNR_v and PSNR_e to represent the PSNR values of the image compressed by two-stage VQ without data embedded and the image reconstructed by the proposed scheme, respectively. From Table 1, we can see that the larger the N_1 and N_2 are, the higher PSNR values we can obtain. The PSNR_v increases with the codebook size and PSNR_e is equal to PSNR_v for any test image with any codebook size, which indicates that our scheme can losslessly recover the original index tables from the bitstream.

Table 1. PSNR and payload values of the proposed scheme with different codebook sizes

(N_1, N_2)	(m, n)	Indicators	Lena	F16	Peppers	Baboon	Boat	Goldhill
(16, 32)	(2 or 3, 4)	PSNR	31.06	30.06	30.46	23.78	28.57	30.18
		Payload	64008	64008	64008	64008	64008	64008
	(2 or 3, 8)	PSNR	31.06	30.05	30.46	23.78	28.57	30.18
		Payload	93744	93744	93744	93744	93744	93744
(32, 64)	(2 or 3 or 4, 4)	PSNR	32.26	31.62	31.74	24.73	30.14	31.28
		Payload	64008	64008	64008	64008	64008	64008
	(2 or 3 or 4, 8)	PSNR	32.26	31.62	31.74	24.73	30.14	31.28
		Payload	93744	93744	93744	93744	93744	93744
(64, 128)	(2 or 3 or 4, 4)	PSNR	34.29	32.89	32.87	25.66	31.15	32.29
		Payload	64008	64008	64008	64008	64008	64008
	(2 or 3 or 4, 8)	PSNR	34.29	32.89	32.87	25.66	31.15	32.29
		Payload	93744	93744	93744	93744	93744	93744

Table 2. Bit rate and efficiency values of the proposed scheme with N=(32, 64)

(m, n)	Indicators	Lena	F16	Peppers	Baboon	Boat	Goldhill
(2, 4)	Bit rate	0.9788	0.9551	0.9653	1.0485	0.9962	0.9935
	Efficiency	0.2494	0.2556	0.2529	0.2328	0.245	0.2457
(3, 4)	Bit rate	1.0646	1.0349	1.0408	1.1286	1.0828	1.096
	Efficiency	0.2293	0.2359	0.2345	0.2163	0.2254	0.2227
(4, 4)	Bit rate	1.0573	1.0303	1.0396	1.1206	1.0754	1.088
	Efficiency	0.2309	0.2369	0.2348	0.2178	0.227	0.2243
(2, 8)	Bit rate	1.0927	1.067	1.0786	1.1505	1.1052	1.1075
	Efficiency	0.3272	0.3351	0.3315	0.3108	0.3235	0.3228
(3, 8)	Bit rate	1.1746	1.1513	1.1666	1.227	1.1903	1.2058
	Efficiency	0.3044	0.3106	0.3065	0.2914	0.3004	0.2965
(4, 8)	Bit rate	1.1667	1.1392	1.1604	1.2196	1.182	1.1973
	Efficiency	0.3064	0.3139	0.3081	0.2932	0.3025	0.2986

Besides the codebook size, m and n are two important parameters in the proposed algorithm. Since the codewords are stored in the codebook in the ascending order of their mean values, it is reasonable that the adjacent indices are close to each other because of the high correlation among neighboring image blocks. Consequently, most of the difference values are near zero. Based on this phenomenon, we adopt the parameter m to reduce the length of bitstream. The selection of m is mainly based on the distribution of absolute differences (AD). We take the Lena image as an example to illustrate the distribution of AD in Fig. 6. We can clearly see that the AD values concentrate around zero for any codebook size, especially for the first VQ stage. In our scheme, if $E_1 = 0$, then only E_2 is encoded in the bitstream, so the length of the bitstream is reduced to some extent. In addition, we find that most AD values are smaller than 4 in the first stage, whereas they distribute evenly in the second stage. Hence we adopt m=2 in our scheme. In order to show the effect of different m, n values on performance, Table 1 lists the PSNR, and payload values for different sets of m, n values with the codebook sizes N=(16, 32), $N = (32, 64)$ and $N = (64, 128)$ respectively. Clearly, both m and n have no effects on PSNR values. On the other hand, n affects the payload values greatly while m has no effect on them.

With regard to the performance of bit rate and efficiency, we also perform several experiments. First, for $N = (32, 64)$, we test how the m value affects the performance based on three cases with m=2, 3 and 4 respectively. Table 2 shows the bit rate and efficiency values of the proposed scheme for n=4 and n=8 respectively. For the case of m=2, the proposed scheme has the lowest bit rate and the highest efficiency. However, for other m values the bit rates are nearly the same and the efficiencies are nearly the same too. This phenomena can be explained by the reason that most AD values are smaller than 4 in the first stage as shown in Fig. 6. In the following experiments, we test how the value of n and the codebook size affect the bit rate and efficiency. As shown in Fig. 7, the bit rate increases with the codebook size and the value of n. In contrast, the efficiency decreases with the codebook size but increases with n value as shown in Fig. 8.

4.2. **Comparison with Previous Schemes.** According to the above analysis, we take m=2 and N=(32, 64) in comparisons of our scheme with previous algorithms, namely, the path optional lossless data hiding scheme [19] and the VQ-index residual value coding based lossless data hiding scheme [12]. Wang *et al.*s scheme [19] has been described in Subsection 2.2. With regard to Lu *et al.*s scheme [12], its main idea is to encode the cover image with a VQ index table, and then utilize the relationship between the neighboring four indices of the current index and their mean values to hide the secret bit, and finally the index difference is encoded by the proper bit stream to reduce the bit rate. The proposed algorithm adopts the codebook size N=(32, 64), while the two previous algorithms have the codebook size 512.

Table 3 and Table 4 list the comparison results of PSNR, payload, bit rate and efficiency among the proposed, Wang *et al.*'s [19] and Lu *et al.*'s schemes [12]. Here Table 3 gives the results for six test images, while Table 4 gives the average results over 100 test images of size 512×512. From Table 3 and Table 4, we can see that our scheme can obtain larger PSNR values. Also, the payload and bit rate of the proposed algorithm are improved to some extent. Especially, our scheme achieves the payload twice larger than Wang *et al.*'s scheme and nearly four times larger than Lu *et al.*'s scheme. Fig. 9 compares the efficiency values of these three methods. From Fig.9, we can see that our scheme obtains higher efficiency values than Wang *et al.*'s and Lu *et al.*'s schemes for both n=4 and n=8. In a word, the proposed scheme can not only improve the payload with a higher PSNR but also enhance the transmission efficiency.

5. **Conclusion.** A lossless data hiding algorithm based on two-stage VQ is presented in this paper. We perform the embedding process on the two stage index tables jointly to

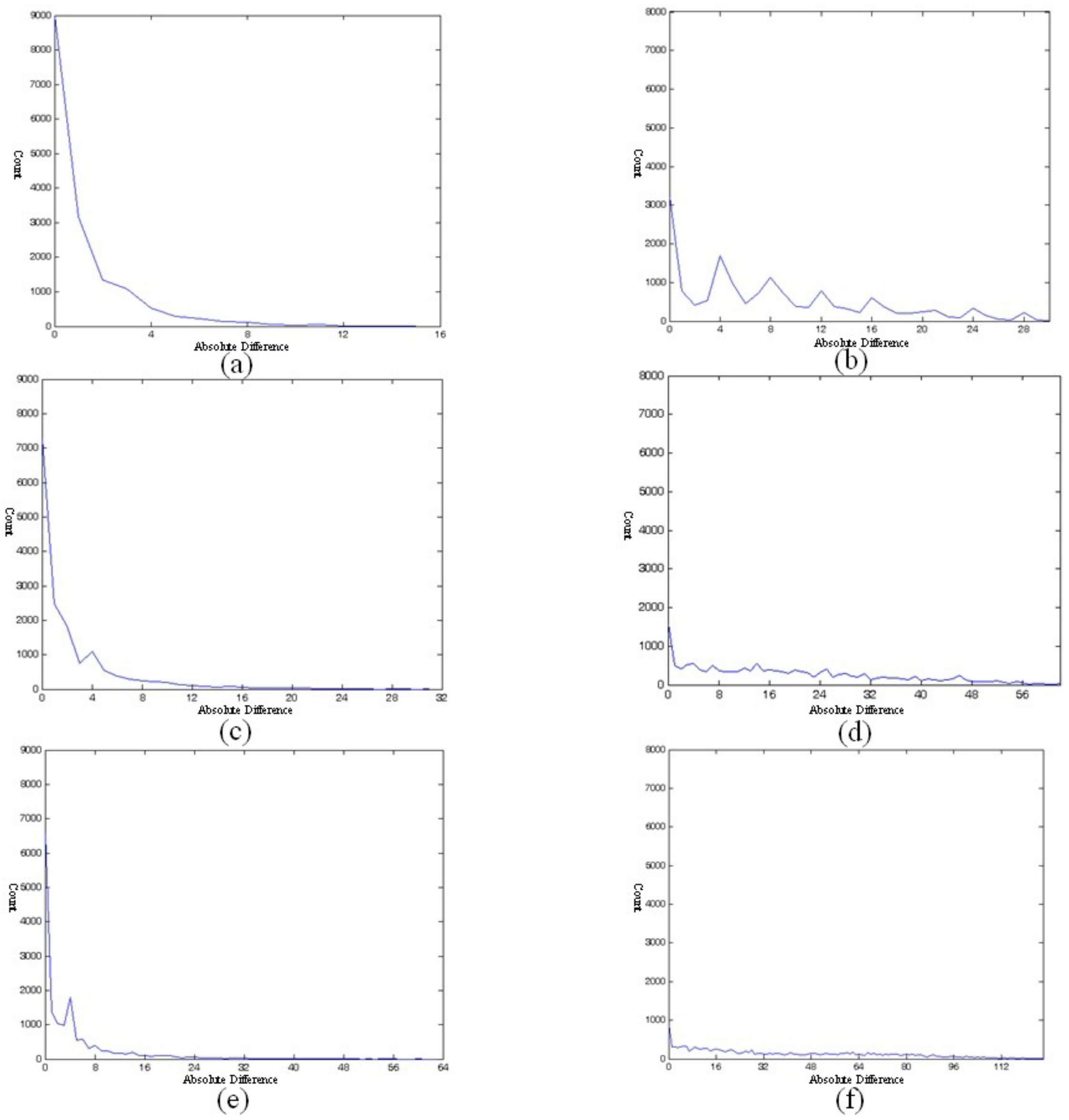

Figure 6. The distribution of AD between Lena images VQ indices for (a) the first stage with N_1=16; (b) the second stage with $N_2 = 32$ (while $N_1 = 16$); (c) the first stage with $N_1 = 32$; (d) the second stage with $N_2 = 64$ (while $N_1 = 32$); (e) the first stage with $N_1 = 64$; (f) the second stage with $N_2 = 128$ (while $N_1 = 64$).

achieve higher payload and efficiency than some previous works. The proposed algorithm enhances the performance in three aspects. (1) Since our scheme can losslessly recover the original index tables and secret data from the bitstream, the process of encoding and decoding is reversible. (2) Given the two index tables, double payload of Wang et al.s scheme and more than four times payload of Lu et al.s scheme can be obtained in our scheme. (3) The transmission efficiency is greatly enhanced with the strategy of embedding secret data in two index tables. The main contribution of this paper lies in two aspects. One is that we develop a high-payload reversible hiding scheme for multistage VQ-compressed images for the first time, which can be further popularized to reversible data hiding in MSVQ-based QuickTime 1.x video. The other is that we improve both the payload and transmission efficiency with a smaller codebook in comparison with traditional VQ-based

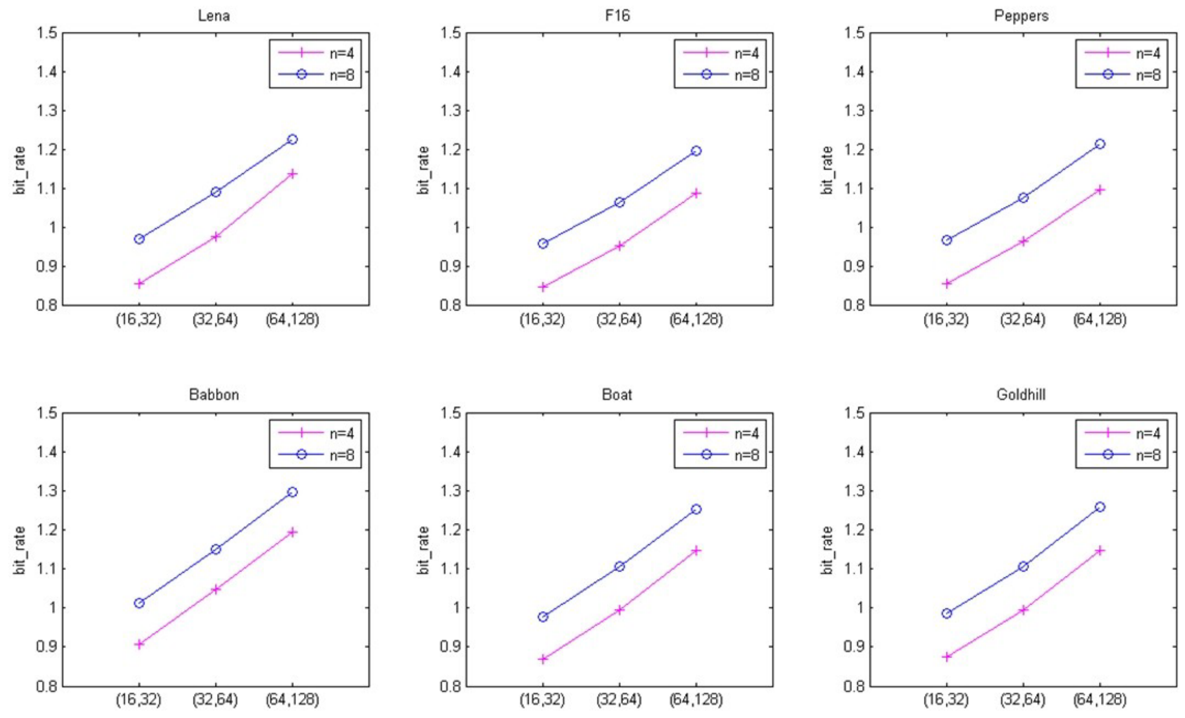

FIGURE 7. The bit rate values under different codebook sizes for various test images.

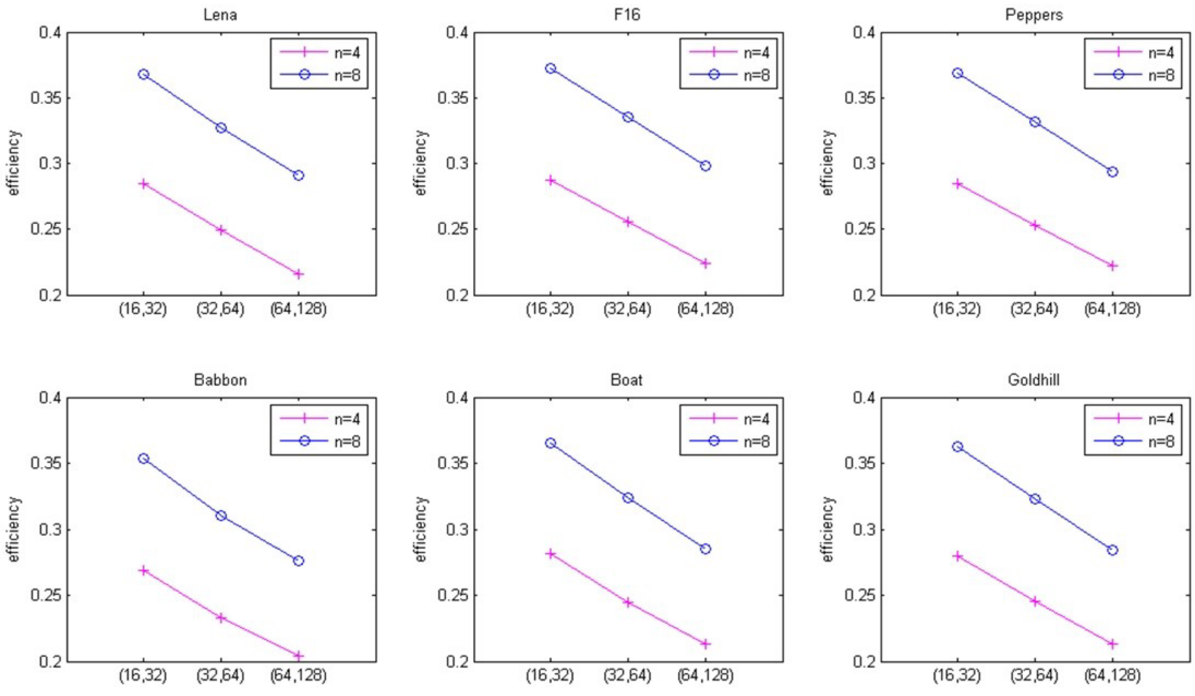

FIGURE 8. The efficiency values under different codebook sizes for various test images.

reversible data hiding schemes. The drawback of the proposed scheme lies in that the bit-rate of the stego bitstream is a bit higher than the bit-rate of the original bitstream. Future work will be concentrated on maintaining the bitstream length unchanged after embedding secret bits.

Table 3. Comparison of the proposed, Wang *et al.*'s and Lu *et al.*'s schemes for six test images

Scheme	N	n	Indicators	Lena	F16	Peppers	Baboon	Boat	Goldhill
Proposed	(32, 64)	4	Payload	64,008	64,008	64,008	64,008	64,008	64,008
			Bit rate	0.979	0.956	0.965	1.049	0.996	0.994
			Efficiency	0.249	0.256	0.253	0.233	0.245	0.246
		8	Payload	93,744	93,744	93,744	93,744	93,744	93,744
			Bit rate	1.093	1.067	1.079	1.151	1.105	1.108
			Efficiency	0.327	0.335	0.332	0.311	0.324	0.323
		-	PSNR_v	32.26	31.62	31.74	24.73	30.14	31.28
			PSNR_e	32.26	31.62	31.74	24.73	30.14	31.28
Wang *et al.*'s	512	4	Payload	32,004	32,004	32,004	32,004	32,004	32,004
			Bit rate	0.641	0.649	0.632	0.691	0.665	0.644
			Efficiency	0.188	0.173	0.181	0.188	0.163	0.173
		8	Payload	47,250	47,250	47,250	47,250	47,250	47,250
			Bit rate	0.706	0.722	0.701	0.751	0.732	0.731
			Efficiency	0.251	0.249	0.250	0.252	0.241	0.248
		-	PSNR_v	31.22	31.61	30.56	23.89	30.04	31.23
			PSNR_e	31.22	31.61	30.56	23.89	30.04	31.23
Lu *et al.*'s	512	-	Payload	14,954	13,588	14,784	15,662	14,785	15,104
			Bit rate	0.569	0.589	0.632	0.691	0.665	0.664
			Efficiency	0.100	0.088	0.089	0.086	0.084	0.086
			PSNR_v	31.22	31.61	30.56	23.89	30.04	31.23
			PSNR_e	31.22	31.61	30.56	23.89	30.04	31.23

Table 4. Comparison of the proposed, Wang *et al.*'s and Lu *et al.*'s schemes based on averaging the performance over 100 test images

Indicators	Proposed $N_1 = 32, N_2 = 64$		Wang *et al.* 's $N = 512$		Lu *et al.* 's N=512
	$n = 4$	$n = 8$	$n = 4$	$n = 8$	
Payload	64,008	93,744	32,004	47,250	14,908
Bit rate	0.990	1.103	0.648	0.726	0.623
Efficiency	0.247	0.324	0.188	0.248	0.091
PSNR_v	30.89		30.74		30.27
PSNR_e	30.89		30.74		30.27

6. **Conclusions.** A lossless data hiding algorithm based on two-stage VQ is presented in this paper. We perform the embedding process on the two stage index tables jointly to achieve higher payload and efficiency than some previous works. The proposed algorithm enhances the performance in three aspects. (1) Since our scheme can losslessly recover the original index tables and secret data from the bitstream, the process of encoding and decoding is reversible. (2) Given the two index tables, double payload of Wang et al.s scheme and more than four times payload of Lu et al.s scheme can be obtained in our scheme. (3) The transmission efficiency is greatly enhanced with the strategy of embedding secret data in two index tables. The main contribution of this paper lies in two aspects. One is that we develop a high-payload reversible hiding scheme for multistage VQ-compressed images for the first time, which can be further popularized to reversible data hiding in MSVQ-based QuickTime 1.x video. The other is that we improve both the payload and transmission efficiency with a smaller codebook in comparison with traditional VQ-based

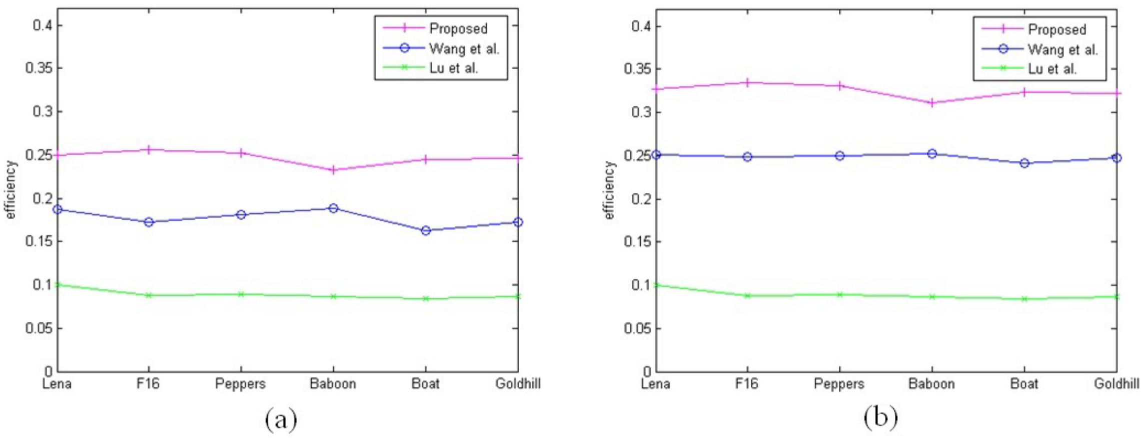

FIGURE 9. Comparisons of efficiency values among our scheme, Wang *et al.*'s scheme and Lu *et al.*'s scheme with (a) n=4 and (b) n=8.

reversible data hiding schemes. The drawback of the proposed scheme lies in that the bit-rate of the stego bitstream is a bit higher than the bit-rate of the original bitstream. Future work will be concentrated on maintaining the bitstream length unchanged after embedding secret bits.

Acknowledgment. This work was partially supported by the National Natural Science Foundation of China under grant 61171150 and 61103174, and also supported by Zhejiang Provincial Natural Science Foundation of China under grant R1110006, the Special Fund on Strategic New Industry Development of Shenzhen under grant JCYJ20120817163934173, JCYJ20130329105534856, ZDSY20120613125016389, JCYJ20120613113535357 and the Natural Science Foundation of SZU under grant 201215.

REFERENCES

[1] R. Davis, The data encryption standard in perspective, *IEEE Communications Society Magazine*, vol. 16, no. 6, pp. 5-9, 1978.
[2] R. L. Rivest, A. Shamir, and L. Adleman, A method for obtaining digital signatures and public-key cryptosystems, *Communications of the ACM*, vol. 21, no. 2, pp. 120-126, 1978.
[3] Z. H. Wang, C. C. Chang, C. C. Lin, and M. C. Li, A reversible information hiding scheme using left-right and up-down Chinese character representation, *Journal of Systems and Software*, vol. 82, no. 8, pp. 1362-1369, 2009.
[4] T. D. Kieu, and C. C. Chang, A high stego-image quality steganographic scheme with reversibility and high payload using multiple embedding strategy, *Journal of Systems and Software*, vol. 82, no. 10, pp. 1743-1752, 2009.
[5] C. C. Chang, J. Y. Hsiao, and C. S. Chan, Finding optimal least-significant-bit substitution in image hiding by dynamic programming strategy, *Pattern Recognition*, vol. 36, no. 7, pp. 1583-1595, 2003.
[6] J. Mielikainen, LSB matching revisited, *IEEE Signal Processing Letters*, vol. 13, no. 5, pp. 285-287, 2006.
[7] C. C. Chang, C. C. Lin, C. S. Tseng, W. L. Tai, Reversible hiding in DCT-based compressed images, *Information Sciences*, vol. 177, no. 13, pp. 2768-2786, 2007.
[8] M. Iwata, K. Miyake, and A. Shiozaki, Digital steganography utilizing features of JPEG images, *IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences*, vol. E87A, no. 4, pp. 929-936, 2004.
[9] J. Tian, Wavelet-based reversible watermarking for authentication, Proc. of SPIE, vol. 4675, pp. 679-690, 2002.
[10] R. M. Gray, Vector quantization, *IEEE ASSP Magazine*, vol. 1, pp. 4-29, 1984.
[11] C. C. Chang, Y. C. Chou, and Y. P. Hsieh, Search-order coding method with indicator-elimination property, *Journal of Systems and Software*, vol. 82, no. 3, pp. 516-525, 2009.

[12] Z. M. Lu, J. X. Wang, and B. B. Liu, An improved lossless data hiding scheme based on image VQ-index residual value coding, *Journal of Systems and Software*, vol. 82, no. 6, pp. 1016-1024, 2009.

[13] W. J. Chen, and W. T. Huang, VQ indexes compression and information hiding using hybrid lossless index coding, *Digital Signal Processing*, vol. 19, no. 3, pp. 433-443, 2009.

[14] B. Yang, Z. M. Lu, and S. H. Sun, Reversible watermarking in the VQ-compressed domain, *Proc. of 5th IASTED International Conference on Visualization, Imaging, and Image Processing*, pp. 298-303, 2005.

[15] C. C. Chang, W. L. Tai, and C. C. Lin, A reversible data hiding scheme based on side match vector quantization, *IEEE Trans. Circuits and Systems for Video Technology*, vol. 16, no. 10, pp. 1301-1308, 2006.

[16] C. C. Chang, and C. Y. Lin, Reversible steganographic method using SMVQ approach based on declustering, *Information Sciences*, vol. 177, no. 8, pp. 1796-1805, 2007.

[17] C. C. Chang, T. D. Kieu, and W. C. Wu, A lossless data embedding technique by joint neighboring coding, *Pattern Recognition*, vol. 42, no. 7, pp. 1597-1603, 2009.

[18] D. H. Chu, Z. M. Lu, and J. X. Wang, A high capacity reversible information hiding algorithm based on difference coding of VQ indices, *ICIC Express Letters, Part B: Applications*, vol. 3, no. 4, pp. 701-706, 2012.

[19] J. X. Wang, and Z. M. Lu, A path optional lossless data hiding scheme based on VQ joint neighboring coding, *Information Sciences*, vol. 179, no. 19, pp. 3332-3348, 2009.

[20] C. C. Chang, W. C. Wu, and Y. C. Hu, Lossless recovery of a VQ index table with embedded secret data, *Journal of Visual Communication and Image Representation*, vol. 18, no. 3, pp. 207-216, 2007.

[21] C. C. Chang, C. Y. Lin, and Y. H. Fan, Lossless data hiding for color images based on block truncation coding, *Pattern Recognition*, vol. 41, no. 7, pp. 2347-2357, 2008.

[22] Y. Linde, A. Buzo, and R. M. Gray, An algorithm for vector quantizer design, *IEEE Trans. Communications*, vol. 28, no. 1, pp. 84-95, 1980.