International Journal of Innovative
Computing, Information and Control
Volume **9**, Number **9**, September **2013**

ICIC International ©2013 ISSN 1349-4198
pp. 3851–3861

LOSSLESS DATA HIDING BASED ON IMPROVED VQ INDEX JOINT NEIGHBORING CODING

Zhe-Ming Lu, Hua Chen and Fa-Xin Yu

School of Aeronautics and Astronautics
Zhejiang University
No. 38, Zheda Road, Hangzhou 310027, P. R. China
zheminglu@zju.edu.cn

Received September 2012; revised January 2013

Abstract. *Data hiding is a form of steganography to embed secret data into digital media for the purpose of content authentication, annotation and copyright protection. Lossless data hiding is a special kind of data hiding technique that guarantees the digital media can be losslessly restored after the secret data are correctly extracted. In the past several years, several reversible data hiding schemes for Vector Quantization (VQ) compressed images have been proposed. Some embed data during the VQ encoding process such as Side-Match VQ (SMVQ)-based and Modified Fast Correlation VQ (MFCVQ)-based methods; others embed data during the VQ index streaming process such as the Joint Neighboring Coding (JNC)-based scheme. This paper presents an improved JNC (IJNC) process for both lossless data hiding and lossless index compression. The JNC process performs data hiding on each index outside the seed area, while the proposed IJNC process performs data hiding on each 2×2 index block without any seed area. Experimental results show that our scheme outperforms three recently proposed schemes, namely JNC-based, SMVQ-based and MFCVQ-based data hiding methods, in terms of the hiding capacity and the stego image quality.*

Keywords: Data hiding, Lossless data hiding, Vector quantization, Joint neighboring coding, Index compression

1. **Introduction.** The concealment of information within computer files is a common way used in modern steganography [1-3]. Media files are ideal ones for steganographic transmission because of their large size. In this paper, we focus on techniques for hiding information in images. Data hiding schemes can be broadly classified into two categories, i.e., irreversible data hiding and reversible data hiding. For some special applications in military, medical and forensic areas, it is crucial to recover the original host image without any distortion after the extraction of the secret data. Reversible data hiding in images can be performed in the spatial or compressed domain. Barton presented the first spatial-domain reversible data hiding scheme [4] that compresses several bits of the original host image and then hides the compressed data and the secret data in the host image. Celik *et al.* proposed a novel reversible data hiding algorithm [5] based on a generalization of the well-known LSB (Least Significant Bit) modification. Tian proposed a famous reversible data hiding technique [6] based on a so-called difference expansion operation. Thodi and Rodriguez proposed a prediction-error based reversible watermarking scheme [7]. Alattar proposed a reversible watermarking algorithm [8] with very high data hiding capacity for color images. Zhao *et al.* proposed a high capacity novel hiding scheme [9] based on reversible secret sharing.

Since most images are stored in compressed formats such as JPEG and JPEG2000, reversible data hiding techniques in the compressed domain are useful and necessary.

Fridrich *et al.* firstly presented two invertible watermarking methods [10] for digital images in the JPEG format. Xuan *et al.* presented a reversible data embedding method [11] for digital images based on the integer wavelet transform with the companding technique. Bausys and Kriukovas proposed a semi-blind reversible pixel-wise image authentication framework [12] in the frequency domain.

Besides JPEG and JPEG2000 compression methods, Vector Quantization (VQ) is another famous block-based image compression technique that has been widely studied and used due to its characteristics of easy implementation and high efficiency. Yang *et al.* proposed a reversible watermarking scheme [13] based on modified fast correlation VQ (MFCVQ). However, the hiding capacity of the MFCVQ method is unstable and low. To improve the hiding capacity, Chang *et al.* proposed several reversible data hiding algorithms [14-17] based on the side-match VQ (SMVQ) and modified SMVQ. Recently, several reversible data hiding schemes based on VQ index coding have been proposed. Chang *et al.* proposed a reversible data embedding scheme [18] based on the VQ image compression technique which emphasizes that the original VQ compressed codes can be recovered after data extraction. Chang *et al.* presented a reversible information hiding method [19] for VQ-compressed grayscale images by using the so-called joint neighboring coding (JNC) technique. Similarly, Lu *et al.* [20] proposed a reversible data hiding algorithm based on the VQ-index residual value coding. This algorithm can achieve a higher PSNR and a higher data hiding capacity than the algorithms proposed in [13,17]. The main problems of the above algorithms lie in that they require seed areas and either their bit rates are higher than the original VQ method or their PSNR values are lower than the original VQ method.

In this paper, we propose an improved joint neighboring coding (IJNC) algorithm for lossless data hiding with index compression ability. Our contribution lies in the elimination of the restriction of seed areas and the proposal of performing the data hiding and index compression simultaneously based on 2×2 index blocks. Thus, the capacity, one of the most important properties of data hiding schemes, is increased in the proposed scheme. At the same time, the bit rate, one of the most important properties of image compression, is reduced in the proposed scheme. Furthermore, the proposed scheme can preserve the same image quality as the original VQ compressed image.

2. Related Works.

Vector Quantization (VQ) is an attractive block-based encoding method for image compression with a high compression ratio in recent years. VQ can be defined as a mapping from the k-dimensional Euclidean space R^k into a finite codebook $C = \{\boldsymbol{c}_i | i = 0, 1, \ldots, N-1\}$, where $\boldsymbol{c}_i$ is a codeword and N is the codebook size. VQ first generates a representative codebook offline based on a number of training vectors using, typically, the well-known LBG algorithm [21]. During the encoding process, each k-dimensional input image block $\boldsymbol{x} = (x_0, x_1, \ldots, x_{k-1})$ is compared with the codewords in the codebook $C = \{\boldsymbol{c}_0, \boldsymbol{c}_1, \ldots, \boldsymbol{c}_{N-1}\}$ to find the best matching codeword $\boldsymbol{c}_i = (c_{i,0}, c_{i,1}, \ldots, c_{i,k-1})$. And then the index i of the best matching codeword is assigned to $\boldsymbol{x}$. After all the input image blocks are encoded, an index table can be obtained. The indices in the index table are transmitted over the channel to the decoder. The decoder has the same codebook as the encoder. For each index i, the decoder merely performs a simple table look-up operation to obtain $\boldsymbol{c}_i$ and then utilizes $\boldsymbol{c}_i$ to reconstruct each input image block $\boldsymbol{x}$.

2.1. MFCVQ-based reversible data hiding algorithm.

The MFCVQ-based reversible data hiding scheme [13] performs the data hiding process during the VQ encoding. In MFCVQ, for the current image block $\boldsymbol{x}$ outside the seed area, it is encoded by using

its four neighboring encoded image blocks $\boldsymbol{c}_u$, $\boldsymbol{c}_l$, $\boldsymbol{c}_v$ and $\boldsymbol{c}_r$. For each input image block $\boldsymbol{x}$, MFCVQ first checks the embeddability of $\boldsymbol{x}$, and then performs the hiding & coding process for the embeddable block while performs the pure coding process for the non-embeddable block. To check the embeddability, the distances between the encoded adjacent blocks and the current bock $\boldsymbol{x}$ are computed respectively as $d_u = d(\boldsymbol{x}, \boldsymbol{c}_u)$, $d_v = d(\boldsymbol{x}, \boldsymbol{c}_v)$, $d_l = d(\boldsymbol{x}, \boldsymbol{c}_l)$ and $d_r = d(\boldsymbol{x}, \boldsymbol{c}_r)$, and the smallest distance $d_{\min}$ is found out of $\{d_u, d_v, d_l, d_r\}$. Then, $d_{\min}$ is compared with the pre-defined distance threshold TH. If $d_{\min} \leq TH$, the current block $\boldsymbol{x}$ is embeddable and the flag bit '0' is sent to the codestream. Otherwise, the current block $\boldsymbol{x}$ is non-embeddable and the flag bit '1' is sent to the codestream. For the embeddable block, the mean vector $\boldsymbol{c}_{mean} = (\boldsymbol{c}_u + \boldsymbol{c}_v + \boldsymbol{c}_l + \boldsymbol{c}_r)/4$ is first calculated, and then the hiding & coding process is performed according to the secret bit. If the secret bit to be embedded is '0', the best-matched codeword $\boldsymbol{c}_{best}$ of $\boldsymbol{c}_{mean}$ is found out of $\{\boldsymbol{c}_u, \boldsymbol{c}_v, \boldsymbol{c}_l, \boldsymbol{c}_r\}$, and the two location bits of $\boldsymbol{c}_{best}$ are sent to the codestream (the two location bits of $\boldsymbol{c}_l, \boldsymbol{c}_u, \boldsymbol{c}_v$ and $\boldsymbol{c}_r$ are '00', '01', '10' and '11' respectively). If the secret bit to be embedded is '1', the best-matched codeword $\boldsymbol{c}_x$ of $\boldsymbol{x}$ is found out of $\{\boldsymbol{c}_u, \boldsymbol{c}_v, \boldsymbol{c}_l, \boldsymbol{c}_r\}$, and the two location bits of $\boldsymbol{c}_x$ are sent to the codestream. Note that if $\boldsymbol{c}_x = \boldsymbol{c}_{best}$, the next best-matched code vector $\boldsymbol{c}'_x$ ($\neq \boldsymbol{c}_{best}$) should be selected. For the non-embeddable block, the normal coding process is performed, i.e., the best-matched codeword $\boldsymbol{c}_{xbest}$ is found from the whole codebook C for the current block $\boldsymbol{x}$, and the index of $\boldsymbol{c}_{xbest}$ is sent to the codestream.

The main drawback of MFCVQ-based hiding method is that the hiding capacity of the MFCVQ method is unstable and low, since the embeddable locations are determined by a pre-defined distance threshold. Furthermore, the PSNR value of the stego image is much less than the original VQ-compressed image.

2.2. **The SMVQ-based reversible data hiding algorithm.** The original side-match VQ based image coding scheme was proposed by Kim [22]. The SMVQ encoder first uses the master codebook to encode the blocks in the first column and the blocks in the first row. Then all other image blocks are encoded from left to right row by row using the correlation of neighboring encoded blocks. Let $\boldsymbol{x}$ denote the current processing vector, and $\boldsymbol{u}$ and $\boldsymbol{l}$ be the codewords of the upper and left neighboring blocks, respectively. The side-match distortion of a codeword $\boldsymbol{c}$ in C is defined as $smd(c) = vd(c) + hd(c)$, where the vertical and horizontal distortions of $\boldsymbol{c}$ are defined as $vd(c) = \sum_{i=0}^{3} (u_{(3,i)} - c_{(0,i)})^2$ and $hd(c) = \sum_{i=0}^{3} (l_{(i,3)} - c_{(i,0)})^2$, where $x_{(i,j)}$ denotes the value of the j^{th} pixel of the i^{th} row in an image block $\boldsymbol{x}$. Then SMVQ encoder sorts the master codebook C according to the side-match distortion of all codewords, and selects the first N_s ($N_s < N$) codewords to form the state codebook C_s of $\boldsymbol{x}$, whose side-match distortions are the smallest. SMVQ encoder next picks the closest codeword of $\boldsymbol{x}$ from the state codebook C_s and transmits the index of this closest codeword to the receiver.

Chang *et al.* [17] first employed the SMVQ technique in reversible data hiding field. They also view the blocks in the uppermost row and the leftmost column that are encoded by the standard VQ as the seed area. For each block $\boldsymbol{x}$ outside the seed area, a state codebook is generated based on its upper and left encoded blocks. The best matching codeword $\boldsymbol{c}_{\min}$ of the current block $\boldsymbol{x}$ is found from the state codebook. Then an approximate codeword is obtained by

$$\boldsymbol{c}_b = \left\lfloor \frac{2\boldsymbol{c}_{\min} + \boldsymbol{c}_a}{3} \right\rfloor \tag{1}$$

where $\boldsymbol{c}_a$ is the closest codeword to $\boldsymbol{c}_{\min}$ in the state codebook, $\lfloor \boldsymbol{v} \rfloor$ stands for the operation of obtaining a different but closest vector with integer components to $\boldsymbol{v}$. To hide a secret bit '0', the codeword $\boldsymbol{c}_{\min}$ is used to encode the current block. To hide a secret bit '1', the approximate codeword $\boldsymbol{c}_b$ is used to encode the current block. In the extraction phase, the receiver compares each received block with the corresponding codeword from the state codebook. If the Euclidean distance is equal to 0, the secret bit is 0, and the codeword remains unchanged; otherwise, the secret bit 1 is extracted and the codeword $\boldsymbol{c}_b$ is replaced with $\boldsymbol{c}_{\min}$.

The main drawback of SMVQ-based hiding method is that the PSNR value of the stego image is much less than the original VQ-compressed image.

2.3. **JNC-based reversible data hiding algorithm.** Recently, Chang *et al.* [19] presented a new reversible data hiding algorithm based on the joint neighboring coding (JNC) technique to hide data in the VQ index table. This method embeds secret data by using the difference values between the current VQ-compressed index and left or upper neighboring indices. Given a $P \times Q$-sized host gray-level image I and the codebook C containing N 16-dimensional codewords $\boldsymbol{c}_i$, $i = 0, 1, \ldots, N-1$. Before the JNC coding process, several preprocessing steps should be performed. First, compute the sum of each codeword $\boldsymbol{c}_i$ by $s_i = \sum_{j=0}^{15} c_{ij}$. Second, sort the codebook C according to s_i in the ascending order: $s_0 \leq s_1 \leq \cdots \leq s_{N-1}$ in order to make neighboring codewords close to each other. Finally, based on the sorted codebook, encode the $P \times Q$-sized image into a $(P/4) \times (Q/4)$-sized index matrix $\mathbf{P} = \{p_{i,j} | i = 0, 1, \ldots, P/4-1;\ j = 0, 1, \ldots, Q/4-1\}$ that is often called index table.

With the index table $\mathbf{P}$ in hand, the secret data are then embedded by encoding the VQ indices outside the seed area, i.e., $p_{i,j}$, $i \neq 0$ and $j \neq 0$. This process is performed by encoding the difference values d's between the left neighboring index $p_{i,j-1}$ or the upper neighboring index $p_{i-1,j}$ and the index $p_{i,j}$. The embedding strategy is based on the secret bit. If the secret bit is '1', then the difference value $d = p_{i,j-1} - p_{i,j}$ is computed. If the secret bit is '0', then the difference value $d = p_{i-1,j} - p_{i,j}$ is computed. Based on the dynamic range of d, Chang *et al.* considered four cases as follows [19]:

Case 1: $0 < d < 2^m - 1$. The index $p_{i,j}$ is encoded into the binary form $11||d_2$ or $01||d_2$ to represent embedding the secret bit 1 or 0, respectively. Here, $||$ denotes the concatenation operation, and m should be properly selected based on the distribution of index differences.

Case 2: $-(2^m - 1) < d < 0$. The index $p_{i,j}$ is encoded into the binary form $10||d_2$ or $00||d_2$ to represent embedding the secret bit '1' or '0', respectively.

Case 3: $d = 0$. The index $p_{i,j}$ is encoded into the binary form $11||d_2$ or $01||d_2$ to represent embedding the secret bit '1' or '0', respectively. Here, d_2 is represented with m digits '0'.

Case 4: $|d| > 2^m - 1$. The index $p_{i,j}$ is encoded into the binary form $10||00...0$ (m digits '0') $||(p_{i,j})_2$, or $00||00...0$ (m digits '0') $||(p_{i,j})_2$ to represent embedding the secret bit '1' or '0', respectively. Here, $(p_{i,j})_2$ is the binary representation of $p_{i,j}$ with $\log_2 N$ bits.

The main drawback of JNC-based hiding method is that the bit rate of the stego image is a bit larger than the original VQ-compressed image.

3. **Proposed Algorithm.** From Section 2, we can see that MFCVQ, SMVQ and JNC-based data hiding methods do not embed secret bits in the seed area. Second, they perform the data hiding process in embeddable blocks one by one. In order to embed more secret bits, the first strategy that we bethink of is to cancel the seed area and make all blocks embeddable. In order to reduce the coding bit rate, the second strategy that we adopt

is to perform the index coding process on each 2×2 index block rather than one by one. Based on these two strategies, we propose a novel data hiding scheme based on improved joint neighboring coding. The main idea of our scheme is to first vector quantize the cover image, obtaining an index table, and then divide the index table into non-overlapping 2×2 index blocks. Finally, we encode the indices in each 2×2 index block according to four input secret bits and the dynamic range evaluated by the difference between the maximal index value and the minimal index value in the index block. The proposed algorithm consists of three stages, i.e., the VQ encoding stage, the data hiding & index coding stage and the data extraction and decoding stage, which can be illustrated as follows:

3.1. **The VQ encoding stage.** This stage is a preprocessing stage before data hiding and index coding. The aim of this stage is to obtain the index table for later use. Given a $P \times Q$-sized 256 grayscale host image I and the codebook C with N 16-dimensional codewords $\boldsymbol{c}_i$, $i = 0, 1, \ldots, 511$ (we adopt $k = 4 \times 4$ and $N = 512$ in this paper), this stage can be expressed as follows:

Step 1: For each codeword $\boldsymbol{c}_i$, compute its sum value by $s_i = \sum_{j=0}^{15} c_{ij}$, $i = 0, 1, \ldots, 511$.

Step 2: Sort the codebook C according to s_i in the ascending order, $s_0 \leq s_1 \leq \cdots \leq s_{511}$, obtaining the sorted codebook C_s.

Step 3: The original cover image I is divided into non-overlapping 4×4-sized blocks $\boldsymbol{b}_{i,j}$, $i = 0, 1, \ldots, P/4 - 1$, $j = 0, 1, \ldots, Q/4 - 1$.

Step 4: Encode each image block $\boldsymbol{b}_{i,j}$ with the sorted codebook C_s based on the fast nearest neighbor search algorithm [23] or [24], obtaining the index of its best matched codeword, $p_{i,j}$. All these indices compose the index table $\mathbf{P} = \{p_{i,j} | i = 0, 1, \ldots, P/4 - 1; j = 0, 1, \ldots, Q/4 - 1\}$.

3.2. **The data hiding and index coding stage.** With the index table $\mathbf{P}$ in hand, now we can introduce our reversible data hiding and index coding algorithm. Before embedding, we perform the permutation operation on the secret bit sequence in order to enhance the security. Our embedding method is performed not index by index but on each 2×2 index block in order to make full use of the correlation between neighboring indices, and thus the encoding bit rate can be reduced. Assume the index table $\mathbf{P}$ is segmented into non-overlapping 2×2 index blocks, $\boldsymbol{q}_{h,l}$, $h = 0, 1, \ldots, P/8 - 1$; $l = 0, 1, \ldots, Q/8 - 1$. Here, $\boldsymbol{q}_{h,l} = \{p_{2h,2l}, p_{2h}, p_{2l+1}, p_{2h+1,2l}, p_{2h+1,2l+1}\}$, and we adopt {'00', '01', '10', '11'} to denote their locations accordingly. For each 2×2 index block $\boldsymbol{q}_{h,l}$, assume the corresponding 4 secret bits to be embedded are $\{w_1, w_2, w_3, w_4\}$, the data hiding and index coding process can be illustrated detailedly as follows:

Step 1: Find the maximal and minimal indices $p_{\max}$ and $p_{\min}$ in $\boldsymbol{q}_{h,l}$, and record their location bits $w_{\max} \in \{$'00', '01', '10', '11'$\}$ and $w_{\min} \in \{$'00', '01', '10', '11'$\}$ respectively.

Step 2: Calculate the dynamic range $r = p_{\max} - p_{\min}$. For $N = 512$, based on the dynamic range r, consider eight cases as shown in Table 1. Obviously, it requires 3 bits to denote each case. Append these 3 bits (they are called range bits w_{ran} in this paper) to the codestream. (If $N = 256$, we may consider four cases, and thus each case requires 2 bits).

Step 3: Viewing the first two secret bits w_1 and w_2 as the location bits, find the reference index p_{ref} in $\boldsymbol{q}_{m,n}$. (e.g., if the first two secret bits is '01', then $p_{ref} = p_{2h,2l+1}$). Append the 2 secret bits $w_{ref} = w_1 || w_2$ (they are called reference location bits) followed by the 9 bits $(p_{ref})_2$ (since the codebook size $N = 512$) to the codestream. Here, $||$ denotes the concatenation operation, and $(\ )_2$ denotes the operation to get the binary description of a number.

Step 4: According to the third secret bit w_3, select the comparison index p_{com} out of $\{p_{\max}, p_{\min}\}$. If $w_3 = 1$, then set $p_{com} = p_{\max}$ and $w_{com} = w_{\max}$; Otherwise, if $w_3 = 0$,

TABLE 1. Eight cases considered for coding the index differences ($N = 512$)

Case	Dynamic Range $r = p_{\max} - p_{\min}$	Range Bits w_{ran}	m (number of bits required to code each difference)
Case 1	$0 \le r < 2$	'000'	1
Case 2	$2 \le r < 4$	'001'	2
Case 3	$4 \le r < 8$	'010'	3
Case 4	$8 \le r < 16$	'011'	4
Case 5	$16 \le r < 32$	'100'	5
Case 6	$32 \le r < 64$	'101'	6
Case 7	$64 \le r < 128$	'110'	7
Case 8	$r \ge 128$	'111'	9

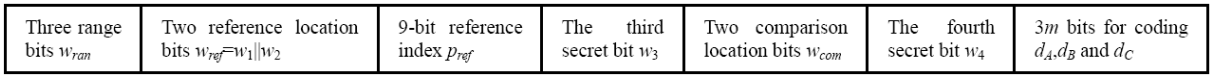

FIGURE 1. The bit string structure for encoding each index block ($N = 512$)

then set $p_{com} = p_{\min}$ and $w_{com} = w_{\min}$. Here we should consider two cases. In Case 1, the reference index and the comparison index are located at the same position, i.e., $p_{ref} = p_{com}$ and $w_{ref} = w_{com}$. Denote the remainder 3 indices in $\boldsymbol{q}_{h,l}$ as p_A, p_B and p_C respectively (in the raster-scan order). In Case 2, the reference index and the comparison index are located at different positions, i.e., $w_{ref} \neq w_{com}$. Except for p_{com}, define $p_A = p_{ref}$ and denote the remainder 2 indices in $\boldsymbol{q}_{h,l}$ as p_B and p_C respectively (in the raster-scan order). Append the third secret bit w_3 followed by the 2 comparison location bits w_{com} to the codestream.

Step 5: According to the secret bits w_3 and w_4, calculate the differences among p_{com}, p_A, p_B and p_C. If $w_4 = 0$ and $w_3 = 0$, then $d_A = p_A - p_{com}$, $d_B = p_B - p_{com}$ and $d_C = p_C - p_{com}$. If $w_4 = 0$ and $w_3 = 1$, then $d_A = p_{com} - p_A$, $d_B = p_{com} - p_B$ and $d_C = p_{com} - p_C$. If $w_4 = 1$ and $w_3 = 0$, then $d_A = p_{com} - p_A$, $d_B = p_{com} - p_B$, $d_C = p_{com} - p_C$. If $w_4 = 1$ and $w_3 = 1$, then $d_A = p_A - p_{com}$, $d_B = p_B - p_{com}$, $d_C = p_C - p_{com}$. We easily find that w_4 can indicate the sign of the differences, i.e., if $w_4 = 0$, the sign is positive; otherwise, the sign is negative.

Step 6: According to the dynamic range r given in Table 1, look up the number of bits (it is denoted by m) required to encode each difference. If $w_4 = 0$, $3m$ bits are used to denote $(d_A)_2$, $(d_B)_2$ and $(d_C)_2$. Otherwise, if $w_4 = 1$, $3m$ bits are used to denote $(-d_A)_2$, $(-d_B)_2$ and $(-d_C)_2$. Append the fourth secret bit w_4 followed by the $3m$ bits of d_A, d_B and d_C to the codestream.

Perform Step 1 to Step 6 for each 2×2 index block $\boldsymbol{q}_{h,l}$, we can hide four bits in each index block and output a bit string whose length is $18 + 3m$ as given in Figure 1. Here, m is different from block to block. If most of the index blocks are with $m < 6$, then the bit rate can be reduced compared to the original index table that requires 36 bits to denote four indices in each index block ($N = 512$). An index coding and data hiding example for a typical 2×2 index block is shown in Figure 2.

3.3. **The decoding and extracting stage.** Our data hiding scheme is reversible because we can recover the original index table after data extraction, and thus the original VQ compressed image can be losslessly recovered. The input is the IJNC codestream, our purpose is to extract the secret bit sequence and recover the original VQ compressed

image. Assume $N = 512$, the detailed decoding and extracting process can be described as follows.

Step 1: Read the next 3 bits into w_{ran} from the codestream, find the corresponding case in Table 1, and look up the value m in Table 1.

Step 2: Read the next 11 bits from the codestream. Get the first 2 bits w_1 and w_2 as the reference location bits w_{ref}, append them to the output secret bit sequence. Transform the last 9 bits into the reference index p_{ref}, and put this index in the current index block according to the location bits w_{ref}.

Step 3: Read the next 3 bits from the codestream. Get the first bit w_3, append it to the output secret bit sequence. Get the last two bits as the comparison location bits w_{com}.

Step 4: Read the next $1+3m$ bits from the codestream. Get the first bit w_4, append it to the output secret bit sequence. Transform the following three m bits into three positive difference values d_{A+}, d_{B+} and d_{C+}. If $w_4 = 0$, $d_A = d_{A+}$, $d_B = d_{B+}$ and $d_C = d_{C+}$; Otherwise, $d_A = -d_{A+}$, $d_B = -d_{B+}$ and $d_C = -d_{C+}$.

Step 5: Except for the reference index, recover the other three indices in the current 2×2 index block. Here, we should consider two cases:

Case 1: $w_{ref} = w_{com}$. In this case, the reference index is just the comparison index, and thus $p_{com} = p_{ref}$. If $w_4 = 0$ and $w_3 = 0$, then $p_A = d_A + p_{com}$, $p_B = d_B + p_{com}$ and $p_C = d_C + p_{com}$. If $w_4 = 0$ and $w_3 = 1$, then $p_A = p_{com} - d_A$, $p_B = p_{com} - d_B$ and $p_C = p_{com} - d_C$. If $w_4 = 1$ and $w_3 = 0$, then $p_A = p_{com} - d_A$, $p_B = p_{com} - d_B$ and $p_C = p_{com} - d_C$. If $w_4 = 1$ and $w_3 = 1$, then $p_A = d_A + p_{com}$, $p_B = d_B + p_{com}$ and $p_C = d_C + p_{com}$. Put p_A, p_B and p_C at the remainder 3 locations in the raster-scan order.

Case 2: $w_{ref} \neq w_{com}$. In this case, p_A is the difference between p_{com} and p_{ref}. If $w_4 = 0$ and $w_3 = 0$, then $p_{com} = d_A - p_{ref}$. If $w_4 = 0$ and $w_3 = 1$, then $p_{com} = p_{ref} + d_A$. If $w_4 = 1$ and $w_3 = 0$, then $p_{com} = p_{ref} + d_A$. If $w_4 = 1$ and $w_3 = 1$, then $p_{com} = d_A - p_{ref}$. Put p_{com} in the current 2×2 index block according to the location bits w_{com}. Now turn to recover left two indices. If $w_4 = 0$ and $w_3 = 0$, then $p_B = d_B + p_{com}$ and $p_C = d_C + p_{com}$. If $w_4 = 0$ and $w_3 = 1$, then $p_B = p_{com} - d_B$ and $p_C = p_{com} - d_C$. If $w_4 = 1$ and $w_3 = 0$,

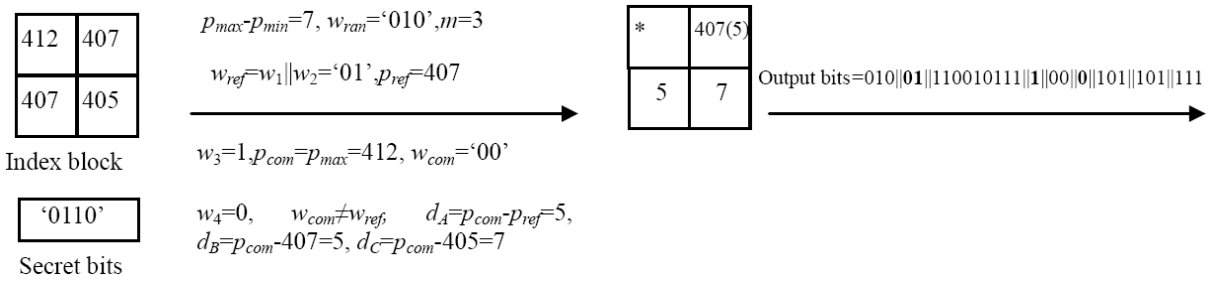

FIGURE 2. A detailed example to hide 4 bits and encode an index block

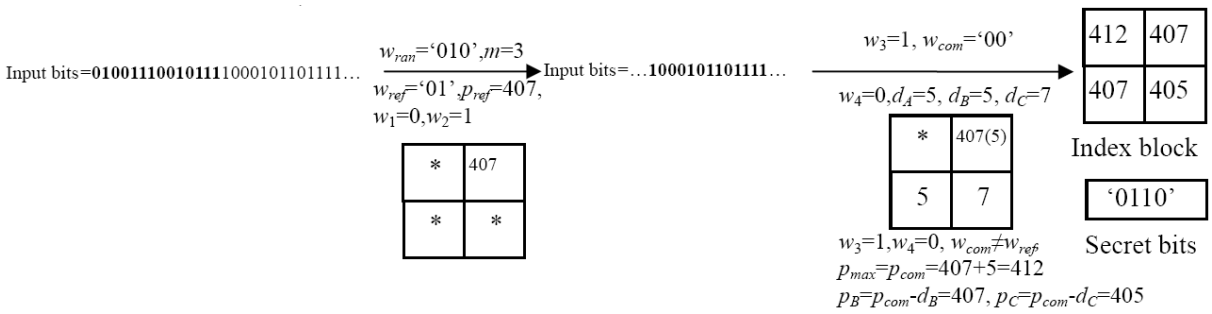

FIGURE 3. A detailed example to extract secret data and decode an index block

then $p_B = p_{com} - d_B$ and $p_C = p_{com} - d_C$. If $w_4 = 1$ and $w_3 = 1$, then $p_B = d_B + p_{com}$ and $p_C = d_C + p_{com}$. Put p_B and p_C at the remainder 2 locations in the raster-scan order.

Repeatedly perform Step 1 to Step 5 for all 2×2 index blocks, we can recover the original VQ index table and output the whole secret bit sequence. Figure 3 gives an example to show the decoding and data extraction process for an index block. According to the restored VQ indices and the sorted codebook, we can reconstruct the original VQ-compressed image. In this way, the whole reversible process is realized.

4. **Experimental Results.** To evaluate the proposed scheme, we use six test images, Lena, Peppers, Mandrill, Boat, Goldhill and Jet F16, of the same size 512×512 with 256 gray scales. The 512-sized master codebook C is generated by the LBG algorithm based on two training images, Lena and Peppers, where the dimension of each codeword is 4×4. Comparisons among our algorithm, the MFCVQ-based algorithm [13], the SMVQ-based algorithm [17] and the JNC-based algorithm [19] are performed. For SMVQ, we set the state codebook size to be 128. For MFCVQ, we set the threshold to be 324 (in this paper we use the squared Euclidean distance, if the normal Euclidean distance is used then the threshold is 18). For JNC, we adopt $m = 6$ as given in [19].

In order to show the distribution of 2×2 index blocks for different images, Table 2 lists the number of index blocks belong to each case of Table 1. Because the total number of bits required to encode an index block based on our algorithm is $18 + 3m$ while the number of bits required to encode four indices based on the basic VQ ($N = 512$) is 36, in order to reduce the bit rate, we expect to have $18 + 3m \leq 36$, i.e., $m \leq 6$, for most index blocks. From Table 2, we can see that, for all images except for the Mandrill image, most index blocks belong to the first 6 cases, thus our scheme can reduce the bit rate as shown in Table 3. The Mandrill image is very complex with high detail and it is outside of the training set, which make the variation of indices in most index blocks be large, and thus most index blocks belong to Case 7 and the bit rate is somewhat larger than that of the normal VQ.

To show the superiority of our proposed algorithm, we compare it with the MFCVQ-based algorithm [13], the SMVQ-based algorithm [17] and the JNC-based algorithm [19]. Four aspects of performance are adopted in the experiments to evaluate a data hiding scheme, i.e., the capacity representing the maximum number of secret bits that can be hidden, the peak signal to noise ratio (PSNR) representing the quality of the stego-image, the bit rate representing the performance of the compression efficiency whose unit is bits per pixel (bpp), and the embedding efficiency indicating the number of embedded secret data when a bit of the binary code stream has been transmitted. Obviously, the embedding efficiency can be calculated as Capacity/(Bit rate $\times P \times Q$). Because the VQ codebook is generated from the Lena and Peppers images, so their PSNRs based on any schemes are much higher than those of other test images outside the training set. Because the Mandrill

Table 2. The number of 2×2 index blocks belong to each case in Table 1

Image	Case 1 ($m = 1$)	Case 2 ($m = 2$)	Case 3 ($m = 3$)	Case 4 ($m = 4$)	Case 5 ($m = 5$)	Case 6 ($m = 6$)	Case 7 ($m = 7$)	Case 8 ($m = 9$)
Lena	162	51	362	782	890	738	622	489
Peppers	211	147	376	756	928	599	505	574
Mandrill	2	2	55	216	597	1216	1502	506
Boat	283	70	217	698	887	843	638	460
Glodhill	227	42	140	474	1010	1189	812	202
Jet_F16	678	286	638	512	402	438	591	551

TABLE 3. Comparisons of the proposed IJNC-based, MFCVQ-based, SMVQ-based and JNC-based algorithms

Algorithm	Performance	Lena	Peppers	Mandrill	Boat	Goldhill	Jet_F16
VQ	PSNR (dB)	30.860	30.428	22.070	28.184	29.618	28.014
($N = 512$)	Bit Rate (bpp)	0.563	0.563	0.563	0.563	0.563	0.563
	PSNR (dB)	30.860	30.428	22.070	28.184	29.618	28.014
Proposed IJNC	Capacity (Bits)	16384	16384	16384	16384	16384	16384
($N = 512$)	Bit Rate (bpp)	0.534	0.528	0.583	0.532	0.537	0.498
	Efficiency	0.117	0.118	0.107	0.117	0.116	0.126
	PSNR (dB)	30.860	30.428	22.070	28.184	29.618	28.014
JNC	Capacity (Bits)	16129	16129	16129	16129	16129	16129
($N = 512$, $m = 6$)	Bit Rate (bpp)	0.574	0.573	0.606	0.567	0.567	0.572
	Efficiency	0.107	0.107	0.102	0.109	0.109	0.108
	PSNR (dB)	28.443	28.139	21.412	26.148	27.823	26.079
MFCVQ-based	Capacity (Bits)	7435	6875	957	7107	2304	8175
($N = 512$, $TH = 324$)	Bit Rate (bpp)	0.425	0.440	0.599	0.434	0.563	0.406
	Efficiency	0.067	0.059	0.006	0.062	0.016	0.076
	PSNR (dB)	28.192	27.947	21.042	25.881	27.422	25.832
SMVQ-based	Capacity (Bits)	16129	16129	16129	16129	16129	16129
($N = 512$, $N_s = 128$)	Bit Rate (bpp)	0.44	0.44	0.44	0.44	0.44	0.44
	Efficiency	0.140	0.140	0.140	0.140	0.140	0.140

image is a high-detail image outside the training set, its encoding quality is far worse than any other image. As shown in Table 3, the PSNRs of stego images in our scheme and the JNC-based method are exactly equal to those of the original VQ compressed images, the reason is that the proposed IJNC-based scheme and the JNC-based scheme reversibly hide data during the coding of the VQ index table and no distortions are introduced in the index coding and data hiding process. The MFCVQ-based and SMVQ-based methods reversibly hide data during the VQ encoding and no operations are performed on the consequent VQ index coding process, and their reversibility means that they can recover the original MFCVQ-encoded image and the SMVQ-encoded image respectively. However, the MFCVQ and SMVQ based encoding methods obtain worse image quality by around 2dB, although they obtain relatively lower bit rates.

With respect to the embedding capacity, the proposed scheme achieves the highest embedding capacity. Compared with the SMVQ-based scheme and the JNC-based scheme, our scheme embeds information in all indices without the seed area, so we can embed 255 more bits in the index table. Our proposed scheme achieves significantly better embedding capacity in comparison with the MFCVQ-based method. The embedding capacity of the MFCVQ-based method is quite low because this method just embeds the secret data into smooth image blocks outside the seed area.

In regard to the bit rate, from Table 3, we can see that the MFCVQ and SMVQ based methods obtain relatively lower bit rates while our scheme and the JNC-based scheme achieve relatively higher bit rates. This is because the MFCVQ scheme only uses 3 bits to encode one embeddable VQ-compressed index, while the SMVQ only uses 7 bits ($N_s = 128$) to denote each index outside the seed area. Anyhow, compared with the ordinary VQ, our scheme can not only hide information but also compress indices, because our scheme can get lower bit rates by around 0.02bpp for most images except the Mandrill image. Compared with the JNC-based scheme, our scheme can achieve lower bit rates for all test images by around 0.04bpp.

In terms of embedding efficiency, the SMVQ-based scheme has the highest efficiency values, and the MFCVQ-based method has the lowest efficiency values. This means that the SMVQ can transmit the most number of embedded secret data when a bit of the binary code stream has been transmitted, while the MFCVQ-based method can transmit the least number of embedded secret data. Our scheme has the second highest efficiency values that are higher than that of the JNC-based scheme by around 0.01.

With regard to the embedding and extracting speed, because our scheme and the JNC-based scheme are based on the index table, if we ignore the VQ encoding process, our scheme and the JNC-based scheme are much faster than the MFCVQ-based and SMVQ-based schemes. Taking the above five attributes into comprehensive consideration, the proposed scheme is a more effective method for its high capacity, high PSNR, high embedding efficiency and low bit rate.

5. **Conclusions.** In this paper, we present an improved joint neighboring coding based reversible data hiding algorithm. The main features of our scheme are as follows: (1) Our scheme can obtain the same PSNR values as the JNC-based scheme. (2) Our scheme does not need the seed area, so it can achieve higher data hiding capacity compared with other algorithms that require the seed area. (3) Our scheme is based on 2×2 index blocks, which can make full use of the correlation between indices; thus it can reduce the bit rate compared with the ordinary VQ. (4) Our proposed scheme is separated into the VQ encoding process for generating the index table and the data hiding process to embed secret data into the output codestream. Obviously, the proposed algorithm realizes the desired "separation of the encoder and the embedder" requirement stated in [13] which facilitates the individual processing of the encoder and the watermark embedder and the controlling of the corresponding performance.

Acknowledgement. This work was partially supported by the National Natural Scientific Foundation of China (Grants No. 61003255 and 61171150). The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation.

REFERENCES

[1] H. Luo, F.-X. Yu, S.-C. Chu and Z.-M. Lu, Hiding multiple watermarks in transparencies of visual cryptography, *International Journal of Innovative Computing, Information and Control*, vol.5, no.7, pp.1875-1881, 2009.

[2] H.-X. Wang, Z.-M. Lu, Y.-N. Li and S.-H. Sun, A compressed domain multipurpose video watermarking algorithm using vector quantization, *International Journal of Innovative Computing, Information and Control*, vol.5, no.5, pp.1441-1450, 2009.

[3] Z.-M. Lu and X.-W. Liao, Counterfeiting attacks on two robust watermarking schemes, *International Journal of Innovative Computing, Information and Control*, vol.2, no.4, pp.841-848, 2006.

[4] J. M. Barton, Method and apparatus for embedding authentication information within digital data, *United States Patent, 5 646 997*, 1997.

[5] M. U. Celik, G. Sharma, A. M. Tekalp and E. Saber, Reversible data hiding, *Proc. of the IEEE Int. Conf. Image Processing*, Rochester, New York, USA, vol.2, pp.157-160, 2002.

[6] J. Tian, Reversible data embedding using a difference expansion, *IEEE Transactions on Circuits and Systems for Video Technology*, vol.13, no.8, pp.890-896, 2003.

[7] D. M. Thodi and J. J. Rodriguez, Prediction-error based reversible watermarking, *Proc. of the IEEE Int. Conf. Image Processing*, Singapore, vol.3, pp.1549-1552, 2004.

[8] A. M. Alattar, Reversible watermark using difference expansion of quads, *Proc. of the IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, Montreal, Quebec, Canada, vol.3, pp.377-380, 2004.

[9] Z. Zhao, K.-Y. Chau and Z.-M. Lu, High capacity data hiding in reversible secret sharing, *International Journal of Innovative Computing, Information and Control*, vol.7, no.11, pp.6411-6422, 2011.

[10] J. Fridrich, M. Goljan and R. Du, Invertible authentication watermark for JPEG images, *Proc. of the IEEE Int. Conf. Information Technology: Coding and Computing*, Las Vegas, NV, USA, pp.223-227, 2001.

[11] G. R. Xuan, C. Y. Yang, Y. Z. Zhen, Y. Q. Shi and Z. C. Ni, Reversible data hiding using integer wavelet transform and companding technique, *Proc. of the 3rd Int. Workshop on Digital Watermarking*, Seoul, Korea, pp.115-124, 2004.

[12] R. Bausys and A. Kriukovas, Reversible watermarking scheme for image authentication in frequency domain, *Proc. of the 48th Int. Symposium ELMAR-2006 Focused on Multimedia Signal Processing and Communications*, Zadar, Croatia, pp.53-56, 2006.

[13] B. Yang, Z. M. Lu and S. H. Sun, Reversible watermarking in the VQ-compressed domain, *Proc. of the 5th Int. Conf. Visualization, Imaging, and Image Processing*, Benidorm, Spain, pp.298-303, 2005.

[14] C. C. Chang and C. Y. Lin, Reversible steganography for VQ-compressed images using side matching and relocation, *IEEE Transactions on Information Forensics and Security*, vol.1, no.4, pp.493-501, 2006.

[15] C. C. Chang and T. C. Lu, Reversible index-domain information hiding scheme based on side-match vector quantization, *Journal of Systems and Software*, vol.79, no.8, pp.1120-1129, 2006.

[16] C. C. Chang, W. L. Tai and M. H. Lin, A reversible data hiding scheme with modified side match vector quantization, *Proc. of the 19th IEEE Int. Conf. Advanced Information Networking and Applications*, Taipei, Taiwan, vol.1, pp.947-952, 2005.

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

[18] 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.

[19] 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.

[20] 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.

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

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

[23] J. S. Pan, Z. M. Lu and S. H. Sun, An efficient encoding algorithm for vector quantization based on subvector technique, *IEEE Transactions on Image Processing*, vol.12, no.3, pp.265-270, 2003.

[24] Z.-M. Lu, S.-C. Chu and K.-C. Huang, Equal-average equal-variance equal-norm nearest neighbor codeword search algorithm based on ordered Hadamard transform, *International Journal of Innovative Computing, Information and Control*, vol.1, no.1, pp.35-41, 2005.