Invited Paper

Journal of Information Processing Systems, Vol.6, No.4, December 2010 DOI : 10.3745/JIPS.2010.6.4.435

Hiding Secret Data in an Image Using Codeword Imitation

Zhi-Hui Wang*, Chin-Chen Chang** and Pei-Yu Tsai***

Abstract—This paper proposes a novel reversible data hiding scheme based on a Vector Quantization (VQ) codebook. The proposed scheme uses the principle component analysis (PCA) algorithm to sort the codebook and to find two similar codewords of an image block. According to the secret to be embedded and the difference between those two similar codewords, the original image block is transformed into a difference number table. Finally, this table is compressed by entropy coding and sent to the receiver. The experimental results demonstrate that the proposed scheme can achieve greater hiding capacity, about five bits per index, with an acceptable bit rate. At the receiver end, after the compressed code has been decoded, the image can be recovered to a VQ compressed image.

Keywords—Data Hiding, Steganography, Vector Quantization

1. INTRODUCTION

Besides cryptography, steganography (data hiding) is an important way to transfer a large amount of secret data. In steganography, a data hiding scheme hides secret data in a digital cover medium, such as text, image, audio or video. This technique can avoid attracting the attention of attackers by using the meaningful cover medium rather than showing meaningless encrypted codes to attackers.

In recent years, many researchers have published a large variety of image data hiding techniques in the literatures. These techniques can be classified roughly into three approaches, i.e., the spatial domain, the frequency domain, and the compression domain. In the spatial domain, the cover image is altered directly and undetectably to conceal the secret message. Lee and Chen [18] proposed a steganographic algorithm applied in the spatial domain in which the least significant bit (LSB) of each pixel in the cover image was replaced by secret data. Later, Chang et al. [19] found the optimal LSB substitution for embedding secret data by using a dynamic programming strategy. By applying both run-length encoding and modular computation, Chang et al. [20] designed two efficient data hiding methods for bitmap files and grayscale files. All of the above methods are irreversible data hiding schemes in the spatial domain. The reversible data hiding scheme in this domain can be classified into three categories, i.e., 1) data hiding by difference expansion, 2) data hiding by histogram shifting, and 3) data hiding by prediction error

Manuscript received October 4, 2010; accepted October 5, 2010.
Corresponding Author: Chin-Chen Chang
* Dept. of Software, Dalian University of Technology, Dalian, China (wangzhihui1017@gmail.com)
** Dept. of Information Engineering and Computer Science, Feng Chia University, Taichung, Taiwan (alan3c@gmail.com)
*** Dept. of Computer Science and Information Engineering, Chung Cheng University, Chiayi, Taiwan (petra.peiyu@gmail.com)

expansion. In 2002, the first difference expansion method was proposed by Tian [1, 2]. In 2003, Ni et al. proposed the first histogram shifting-based data hiding scheme [3]. In 2007, Thodi and Rodriguez proposed a data hiding scheme based on prediction error expansion [4]. Based on the above three schemes, many data hiding methods [5-7] have been proposed to enhance hiding capacity or reduce the distortion of the stego-image.

In the frequency domain, the cover image should be preprocessed by discrete cosine transformation (DCT) [8], discrete wavelet transformation (DWT) [9], or discrete Fourier transform (DFT) [10] to get frequency coefficients. After the frequency coefficients are modified slightly to embed the secret data, the stego image can be obtained by inversing the modified frequency coefficients. In 2001, Fridrich et al. [12] proposed an invertible data hiding scheme to modify the quantization table by using a second-order function. In 2002, Chang et al. [11] embedded the secret data into the mid-frequency coefficients. Next, Xuan et al. [13] employed a reversible scheme to embed data into the high-frequency DWT coefficients with histogram modification. In 2007, Chang et al. [14] presented a lossless embedding scheme for JPEG images.

In the compression domain, the secret data are embedded by the alteration of the compression code. Through the compression method, the size of the digital image data can be reduced significantly. Actually, we transmit digital image data in the compression format in our daily life. Many researchers have interest in hiding information in the compression domain. Among compression techniques, such as JPEG, JPEG2000, VQ, and block truncation coding (BTC), VQ is a simple and efficient method that is widely used. Chang et al. [15] proposed a reversible, data hiding algorithm based on the SMVQ. In the same year, Yang et al. [16] proposed an MFCVQ-based, reversible watermarking scheme by using four adjacent blocks to encode the current block. However, the visual quality and hiding capacity of Yang et al.'s scheme were not good enough. In 2009, Chang et al. [17] used the concept of joint neighboring coding (JNC) in the VQ index table for embedding secrets with reversibility. In this paper, a novel data hiding scheme based on the VQ codebook is explored. The proposed method uses a codebook that has been resorted by PCA (principle component analysis) and exploits the distance of two codewords, which are similar to one image block, to embed the secret data. According to the secret to be embedded and the difference between those two codewords, the original image block is transformed into a difference number table that is compressed by entropy coding before sending to the receiver. At the receiver end, after decoding the compressed code, the secret data can be extracted, and the image can be recovered as a VQ compressed image.

The rest of this paper is organized as follows. In Section 2, we briefly review the concept of PCA, entropy coding, and Chang et al.'s data hiding scheme. In Section 3, we present the novel scheme based on the VQ codebook. Our experimental results are discussed in Section 4. In Section 5, we present our conclusions.

2. RELATED WORK

In this section, the principal component analysis (PCA) approach will be introduced first, followed by the entropy coding technique, which is used to produce a series of compression codes. Finally, Chang et al.'s reversible image-hiding method based on the VQ index table is discussed.

2.1 Principal Component Analysis (PCA)

The principle of VQ encoding is mapping the closest codeword in the codebook for each vector. However, a high computation load is required due to the full search of the codebook. The PCA algorithm is a variable reduction procedure. The much smaller numbers of artificial variables can be obtained from the larger data set. Hence, many schemes (Gray and Linde [21], Chang and Chen [22] and Chang et al. [23]) have been proposed to reduce the search time. Among these methods, Chang et al.'s PCA scheme [23] can achieve both high efficiency in encoding procedure and full-search equivalent. In 2010, Wang et al. [24] used a lossless hiding scheme to embed data into the VQ index table by the PCA algorithm.

The parameters of the PCA algorithm can be obtained from the large data set. The obtained numbers are called principal components. These principal components are used to reduce multidimensional data sets to data sets fewer dimensions. Thus, the principal components are used to simplify many analyses by decreasing number of dimensions of the original data. The first principal component represents the largest percentage of total variance in the original data sets, and the second principal component represents the second largest percentage, and so on. In our proposed scheme, the PCA algorithm is employed to sort the *n*-dimensional vector according to the principal components that are extracted from all codewords in a codebook C. The sorting technique is described as follows. First, all codewords are normalized according to the mean of the variances, which causes the mean of normalized variances to be equal to 0. Second, we calculate the covariance matrix F with the projected results. Assume the eigenvalues are $\lambda_1,\lambda_2,...,\lambda_n$, with $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_n$ and that the eigenvectors that correspond to $\lambda_1,\lambda_2,...,\lambda_n$ are defined as $V_1,V_2,...,V_n$, respectively. Among all eigenvectors, V_1 is the first principal component, which means V_1 has the largest percentage of total variance based on the ratio $\lambda_1/\sum_{i=1}^{n}\lambda_i$. Eq. (1) shows the value p_i, which is the projected value of V_1, and the ith codeword c_i, where $1 \leq i \leq m$, when the codebook sized by $m \times n$. For example, if the codebook size is 256×16, then 256 projected values can be obtained. Third, all of the codewords are sorted based on the projected values. As a result, a new codebook C' can be obtained via the PCA algorithm.

$$p_i = V_1 \cdot c_i \tag{1}$$

For example, assume a codebook contains ten codewords, as shown in Fig. 1. These codewords are normalized as F:

Index	Codeword
1	(22, 22, 22, 29, 22)
2	(117, 92, 70, 65, 94)
3	(209, 210, 210, 210, 209)
4	(97, 162, 185, 186, 95)
5	(57, 53, 51, 48, 55)
6	(81, 79, 81, 85, 107)
7	(216, 218, 217, 216, 219)
8	(103, 109, 116, 118, 105)
9	(46, 55, 77, 107, 45)
10	(167, 152, 122, 99, 164)

Fig. 1. Example codebook

New index	Old index	Projected values	Codeword
1	1	52.17	(22, 22, 22, 29, 22)
2	5	118.02	(57, 53, 51, 48, 55)
3	9	146.99	(46, 55, 77, 107, 45)
4	6	193.27	(81, 79, 81, 85, 107)
5	2	195.75	(117, 92, 70, 65, 94)
6	8	246.29	(103, 109, 116, 118, 105)
7	10	315.04	(167, 152, 122, 99, 164)
8	4	324.92	(97, 162, 185, 186, 95)
9	3	468.45	(209, 210, 210, 210, 209)
10	7	485.46	(216, 218, 217, 216, 219)

Fig. 2. Sorted codebook

$$F = \begin{pmatrix} 4002.05 & 3805.30 & 3437.65 & 3038.65 & 3943.45 \\ 3805.30 & 4116.56 & 4022.98 & 3701.44 & 3792.20 \\ 3437.65 & 4022.98 & 4174.89 & 4004.67 & 3484.05 \\ 3038.65 & 3701.44 & 4004.67 & 3986.41 & 3111.35 \\ 3943.45 & 3792.20 & 3484.05 & 3111.35 & 4008.45 \end{pmatrix}.$$

Then, the eigenvalues, e_1, e_2, e_3, e_4, e_5, and the corresponding eigenvectors, v_1, v_2, v_3, v_4, v_5, can be computed as follows:

$$\begin{aligned} v_1 &= (0.6872, -0.2210, -0.2739, 0.3663, -0.5193), & e_1 &= 47.4181 \\ v_2 &= (-0.2276, 0.5628, -0.7226, 0.3235, 0.0684), & e_2 &= 1.3639 \\ v_3 &= (0.4381, 0.4674, 0.4598, 0.4289, 0.4407), & e_3 &= 18613.3830 \\ v_4 &= (-0.1245, -0.6448, -0.1977, 0.4764, 0.5501), & e_4 &= 114.0842 \\ v_5 &= (0.5182, -0.0095, -0.3903, -0.5918, 0.4783), & e_5 &= 1512.1108 \end{aligned}$$

Recalling the eigenvalue definition, eigenvalues $\lambda_1, \lambda_2, ..., \lambda_5$ are obtained as e_3, e_5, e_4, e_1, e_2, where $e_3 \geq e_5 \geq e_4 \geq e_1 \geq e_2$, respectively. Hence, v_3 is the first principal component direction, denoted as V_1. We compute every projected value of the codeword in the codebook along with V_1. According to those projected values, all codewords can be sorted. The results are shown in Fig. 2.

2.2 Entropy Coding

Entropy coding [25] is a lossless data compression technique used in JPEG compression. There are two kinds of encoding used in entropy coding, i.e., Differential Pulse Code Modulation (DPCM) and run-length encoding (RLE). To begin with, in the procedure of JPEG entropy encoding, a host image of size $N \times N$ is divided into $N/2^3 \times N/2^3$ blocks with 8×8 quantized DCT coefficients in each block; then, two encoding methods are used to encode each block according to the different kinds of coefficients. After Huffman coding is employed in encoding, a series of compression codes will be obtained.

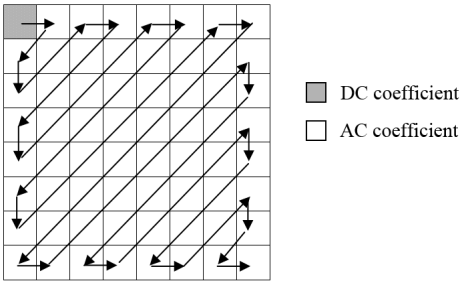


Fig. 3. Two kinds of coefficients and zig-zag scanning

During entropy coding, each block has two kinds of coefficients, i.e., DC and AC. As shown in Fig. 3, in the 8×8 quantized coefficients for each block, the DC coefficient is at the top-left location, and the remaining 63 coefficients are the AC coefficients. DPCM and RLE are used to encode the DC coefficient and the AC coefficients, respectively.

Assume there are $N/2^3\times N/2^3$ blocks and that each block contains 8×8 quantized coefficients. $N/2^3\times N/2^3$ DC coefficients are formed into a line. The ith DC coefficient is called dc_i, $1\le i\le N/2^3\times N/2^3$, and so on. Then, the difference $Diff$ between the current DC coefficient and the DC coefficient of the previous block can be computed using Eq. (2), which is called DPCM. Because the first DC coefficient dc_1 doesn't have the previous one, dc_0 is set to 0. By using Eq. (2), most $Diff$ values are smaller than DC values because the consecutive pixels have the feature of similarity. Thus, the $Diff$ values are favorable for DC Huffman coding using the DC Huffman table, as shown in Fig. 4.

$$Diff(i) = dc_i - dc_{i-1}. \tag{2}$$

$$\text{size} = \lceil \log_2(|Diff|+1) \rceil. \tag{3}$$

The color image is composed of three components, i.e., Y, Cb, and Cr, where Y is Luminance, and Cb and Cr are Chrominance, respectively. The DC Huffman code table is composed of Lu-

Luminance bits	Chrominance bits	size	*Diff*
00	00	0	0
010	01	1	-1,1
011	10	2	-3,-2,2,3
100	110	3	-7..-4,4..7
101	1110	4	-15..-8,8..15
110	11110	5	-31..-16,16..31
1110	111110	6	-63..-32,32..63
11110	1111110	7	-127..-64,64..127
111110	11111110	8	-255..-128,128..255
1111110	111111110	9	-511..-256,256..511
11111110	1111111110	10	-1023..-512,512..1023
111111110	11111111110	11	-2047..-1024,1024..2047

Fig. 4. DC Huffman code table

minance bits, Chrominance bits, size and *Diff*. The Luminance bits and Chrominance bits are the unique codes to distinguish Luminance from Chrominance. The definition of size is that it is the smallest length of the *Diff* value, as shown in Eq. (3). If *Diff* = 9, the size is four. The *Diff* value will transform into one's complement that corresponds to variable-length codes using the Huffman table. The Luminance bits or Chrominance bits will precede each *Diff* value, and the output format is (Luminance bits or Chrominance bits, *Diff*). For example, if we assume that *Diff* = 31 is from a Luminance block, we can determine that the size is 5, which corresponds to the Luminance bits that are set to $(110)_2$ in the Huffman table. In this case, the *Diff* is $(11111)_2$, so the data stream has eight bits, which is $(11011111)_2$. In addition, assuming that *Diff* = -6 is from a Chrominance block, the size is three, which corresponds to the Chrominance bits that are set to $(110)_2$. In this case, because the *Diff* is $(001)_2$, there are six bits for the data stream, which are $(110001)_2$.

As to the AC Huffman coding, the remaining 63 coefficients belong to the AC coefficients. Many coefficients are equal to 0 after the quantization step, as shown in the example in Fig. 5(a), which is suitable for RLE with satisfied compression ratio. In RLE, the 63 quantized AC coefficients via zig-zag order scanning form a sequence, as shown in Fig. 3 and Fig. 5(b). Each AC coefficient is represented by two pieces, i.e., <Run, Level length><Level> (<R, L length><L>) in the zig-zag sequence. The first piece contains Run and Level length. The second piece contains Level. Run is the number of consecutive zero-valued AC coefficients before a non-zero AC coefficient. Level is the value of the AC coefficient. Level length is the number of binary bits of the AC coefficient that are used to encode. Let's consider the example shown in Fig. 5(c). Assume the first non-zero value of AC coefficient in the zig-zag sequence is -2; there are no zero values before this value, so the format is represented as $<0,2><-2>$. In addition, assume the non-zero value of the AC coefficient is -3; there are three zero values before this value, so the format is represented as $<3,2><-3>$. The last one is EOB, which is transmitted after the last non-zero value in 64 coefficients. For this example, EOB is put in after the $<3,2><-3>$, meaning that the last non-zero value in the 64 coefficients is -3. In other words, all of the values after this value are equal to 0. Fig. 5(d) shows the binary bits of the run-length format of AC coefficients. Assuming that the run-length format is $<0,2><-2>$, the bits of $<0,2>$ are 01 by mapping the AC Huffman code table, as shown in Fig. 6, and the bits of $<-2>$ with one's complement are represented as 01. Assuming that the run-length format is

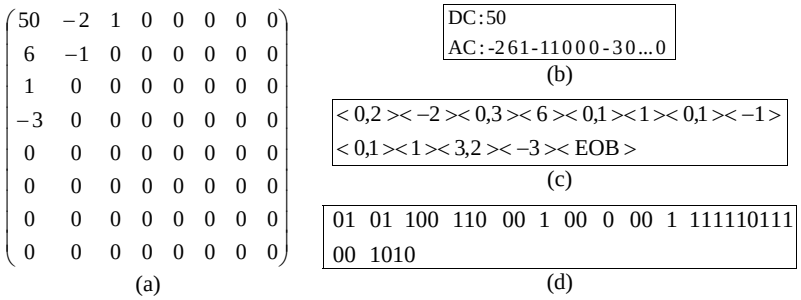


Fig. 5. Example of the AC Huffman coding method based on the DCT block; (a) A block of 8×8 quantized coefficients; (b) the zig-zag sequence of 64 coefficients; (c) the run-length format of AC coefficients; and (d) the data stream of (c) via mapping the AC Huffman code table

<R,L length>	Code Word (in binary)
<0,0>=<EOB>	1010
<0,1>	00
<0,2>	01
<0,3>	100
<1,2>	11011
<2,1>	11100
<3,2>	111110111

Fig. 6. Part of the AC Huffman code table

$<3,2><-3>$, the bits of $<3,2>$ are 111110111, and the bits of $<-3>$ with one's complement are represented as 00. In JPEG, <EOB> is always set to 1010.

From Figs. 5(a)-(d), the output result of the 63 coefficients of AC coefficients is only 34 bits via the AC Huffman coding, which is much less than the number of bits of the original pixels, i.e., $63 \times 8 = 504$ bits. Obviously, the AC Huffman coding reduces the storage requirement significantly. The larger the number of consecutive zero-valued AC coefficients is, the smaller the storage space requirement will be. Normally, there is a large number of the consecutive zero values in the 64 coefficients after the quantization step, so the AC Huffman coding is useful for increasing the ratio of compression.

For the decompression stage, the data stream will be decoded according to the unique codes from the Huffman table.

2.3 Chang et al.'s Data Hiding Scheme

In 2010, Chang et al. published a reversible data hiding scheme based on a VQ index table. To improve the hiding capacity, they reorganized the index sequence of an image's corresponding codebook according to the index appearance frequency in the VQ index table. The procedure includes an embedding procedure and an extracting procedure. The embedding procedure is illustrated as follows:

Step1: Sort codebook (according to the index occurrence frequency in the VQ index table).
Step2: Cluster codebook into *G* groups. The indices in the first group are defined as embeddable indices, while the other indices are defined as un-embeddable indices.
Step3: Transform secret information into base-(*G*-1) equivalence.
Step4: Map each index in the first group with an un-embeddable index, as shown in Fig. 7, where $G = 4$.
Step5: Apply the following embedding rules:
  a. If an index is embeddable, it is replaced with the un-embeddable index value according to the index mapping table.
  b. If an index is un-embeddable, replace it with an embeddable index with some indicator bits appended at the end.

Similar to the embedding procedure, the codebook is divided into *G* groups on the receiver end. If the index value belongs to the first group, the secret data and the original index value can

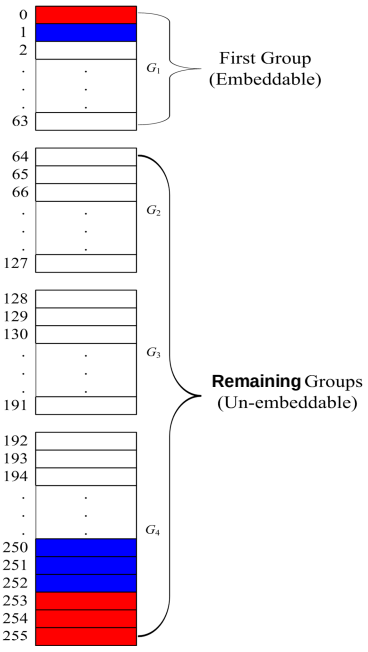


Fig. 7. Index mapping table (G = 4)

be obtained by the indicator bits. If the index value belongs to the remaining groups, the original index value is restored by the corresponding pixel value in the first group, and the secret data can be extracted by using the index mapping table.

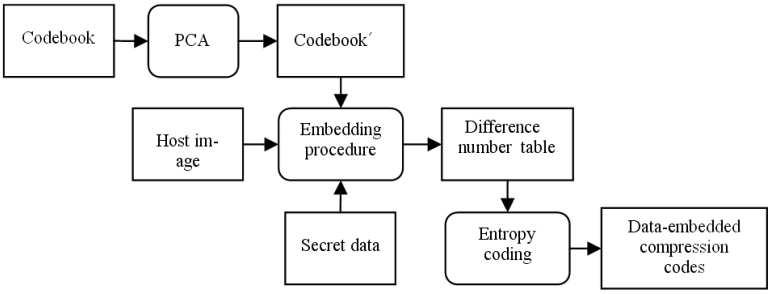
3. PROPOSED SCHEME

In this paper, we propose a codeword-imitated reversible data hiding scheme. Instead of utilizing the index table of an image, the proposed scheme manipulates the item of the codeword. The basic idea is to select an appropriate codeword c_i for an image block B, such that the distortion between c_i and B is as small as possible, and then select another codeword that is the most similar to c_i. Then, based on the differences of the two selected codewords, we can embed a significant quantity of secret data. The overall flowchart of the proposed scheme is shown in Fig. 8.

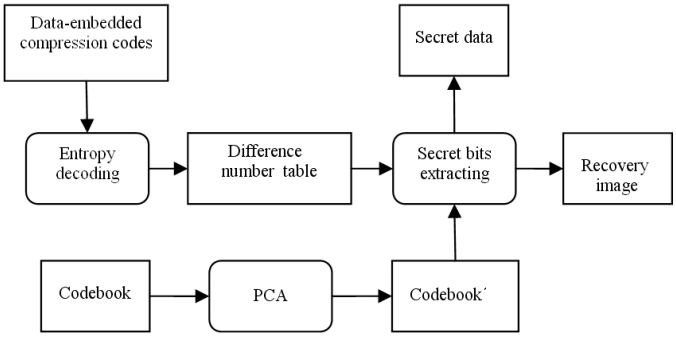
3.1 The Pre-process Phase

The details of the pre-process phase of the proposed scheme are included in the following two steps:

Step1: Codebook C is sorted by PCA and a new codebook C' is formed. Using the PCA algorithm, we can find the projected value for each block of the host image. Assuming that the host image has $N/2^2 \times N/2^2$ 4×4 blocks, the projected value $projval_k$ is defined by Eq. (4):



(a) Data embedding procedure



(b) Data extraction procedure

Fig. 8. Flowchart of the proposed scheme

$$projval_k = B_k \cdot V_1 = \sum_{i=1}^{16} b_i \cdot t_i \tag{4}$$

Here $B_k = [b_1, b_2, ..., b_n]$, $V_1 = [t_1, t_2, ..., t_n]$, and $1 \le k \le N/2^2 \times N/2^2$. B_k and V_1 are the kth block from the host image and the first principal component direction, respectively. The order of the items in B_k is shown in Fig. 9.

Step2: Compare $projval_k$ with all p_i in C' to find the most similar one, which is denoted as *point*. Then, the proposed scheme searches another most-similar codeword *base* for B_k with Euclidean distance code within $[point - r, point + r]$, where r is a threshold that can be adjusted according to the size of the codebook. For example, if the size of the codebook is 256, 512, or 1024, then r can be set to 30, 60, or 120, respectively. To embed secret data, we must establish another codeword *top*. *Top* is determined by *base* and a threshold *T*, as shown in Eq. (5), where $1 \le T \le \lfloor m/2 \rfloor$, and u_1 and u_2 are two indices of *base* and *top*, respectively. *sum* is u_1 with *T* added, while *diff* is u_1 with *T* subtracted. For instance, if T = 50, m = 256 and u_1 = 150, then u_2 is 200. If T = 120, then u_2 is 30. In fact, we use the difference between *base* and *top* to embed secret data. Hence, the embedding capacity is determined mainly by *T*.

$$u_2 = \begin{cases} sum, & \text{if } sum \le \text{codebook size} \\ diff, & \text{otherwise} \end{cases}. \tag{5}$$

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

Fig. 9. A 4×4 block in order of both the host image and the difference number table

3.2 Data Embedding Phase

In the pre-processing procedure, *base* and *top* have been gained in C'. To embed the secret data, the first *n*-2 elements of both *base* and *top* are extracted to calculate their difference. Let *base* and *top* be represented as shown below:

$$base = \{\alpha_i \mid 1 \le i \le n, \quad \alpha_i \in \{0,1,2,...,255\}\}, \tag{6}$$

$$top = \{\beta_i \mid 1 \le i \le n, \quad \beta_i \in \{0,1,2,...,255\}\}. \tag{7}$$

The difference d_i can be described as follows:

$$d_i = \beta_i - \alpha_i, \ 1 \le i \le n-2. \tag{8}$$

Next, our scheme proposes a concept of the difference combination to embed the secret data. The combination approach can be described as:

$$\Gamma = \prod_{i=1}^{n-2} (|d_i| + 1). \tag{9}$$

Here Γ indicates the largest secret number that can be embedded into this block. Therefore, the bit number of the secret data Ψ for a block can be computed as follows:

$$\Psi = \lfloor \log_2 \Gamma \rfloor. \tag{10}$$

Let *S* be the secret bitstream to be embedded in the current block. It can be represented as:

$$S = \{s_k \mid 1 \le k \le \Psi, s_k \in \{0,1\}\}. \tag{11}$$

For example, assume *base* and *top* are [2,2,1,1,0,1,1,3,2,2,1,1,0,2,3,3] and [1,1,2,1,2,2, 1,2,3,5,2,2,1,3,2,4], respectively. The differences $[d_1, d_2, ..., d_{n-2}]$ are calculated as [-1,-1,1,0,2, 1,0,-1,1,3,1,1,1,1]. By Eq. (9), Γ equals 12,288. By using Eq. (10), Ψ = 13, which means that the length of the secret data to be embedded in the block is 13 bits.

Next, generate a difference number table to embed the secret data. Steps (12-1) through (12-4) show the embedding procedure, where the initial value of *l* is 0 and $0 \le l \le n-3$. Let E_l be the decimal number of *S*. According to E_l, we can calculate the numbers of the table in sequence and repeat those four steps until *l* is illegal according to its definition.

0	1	1	1
3	0	0	0
0	1	0	1
0	1	9	6

Fig. 10. Example of a block of the difference number table

$$q_l=\left\lfloor \mathrm{E}_l / \prod_{i=1}^{n-2-l}\left(|d_i|+1\right)\right\rfloor, \tag{12-1}$$

$$g_l=\mathrm{E}_l \bmod \left(\prod_{i=1}^{n-2-l}\left(|d_i|+1\right)\right), \tag{12-2}$$

$$\mathrm{E}_{l+1}=g_l, \tag{12-3}$$

$$l=l+1. \tag{12-4}$$

In order to recover the image, the last two elements of the block are used to embed *base* index, as shown in Fig. 9. For a codebook sized m, we redefine *m*, as shown in Eq. (13), where w_1 and w_2 are two closest integers. Eq. (14) describes the converted format of the *base* index. By Eq. (15), the embedding information $[q_1,q_2,...,q_{n-3},g_{n-3},z_1,z_2]$ is transformed into $[x_1,x_2,...,x_n]$.

$$m=w_1\times w_2. \tag{13}$$

$$z_i=\begin{cases}\lfloor u_1/w_2\rfloor, & \text{if } i=1\\ u_1 \bmod w_2, & \text{if } i=2\end{cases}. \tag{14}$$

$$x_l=\begin{cases}q_l, & \text{if } l<n-2\\ g_{n-3}, & \text{if } l=n-2\\ z_1, & \text{if } l=n-1\\ z_2, & \text{if } l=n\end{cases}. \tag{15}$$

Remember that we assumed that the random secret bits $[s_1,s_2,...,s_{13}]$ are generated as [1,0,1,1,1,0,1,0,0,1,1,0,1]. Then, the decimal number E_1 = 5965. By Eqs. (12-1) through (12-4), part of the embedding information can be obtained as [0,1,1,1,3,0,0,0,0,1,0,1,0,1]. In addition, due to the fact that $u_1=150$, and $w_1=w_2=16$ when the size of the codebook is 256, we have $z_1=9$ and $z_2=6$ by Eq. (14). Using Eq. (15) to transform the embedding information, x_l can be obtained and embedded in the difference number table in sequence, as shown in Fig. 10. Finally, entropy coding is exploited to encode the difference number table and *T*.

3.3 Data Extraction and Image Restoration

When the receiver gets the data-embedded compression codes, entropy decoding is used to decode the compression code to get the difference number table and *T*. The sorted codebook C'

is used to extract the secret data and recover the image. Let $[\bar{x}_1,\bar{x}_2,...,\bar{x}_n]$ be the extracting data from the difference number table, where *n* is the element number of a block. Then, $\bar{u}_1$ is calculated by Eq. (16):

$$\bar{u}_1=\bar{x}_{n-1}\times w_2+\bar{x}_n . \tag{16}$$

Here $\bar{u}_1$ is the new *base* index in C', and the new *top* index, $\bar{u}_2$, also can be computed as a function of *T*. The next step is similar to Eq. (8), the new differences between the new *base* and the new *top* are redefined as $\bar{d}_i, 1\le i\le n$. Apply the same methods as shown in Eqs. (9) and (10), the length of the extracted secret data for the block can be obtained and redefined as $\bar{\Psi}$. In addition, we can extract the secret data according to the following equation:

$$\bar{\mathrm{E}}_1=\sum_{i=1}^{n-3}\left[\bar{x}_i\times\prod_{j=1}^{n-2-i}\left(\left|\bar{d}_j\right|+1\right)\right]+\bar{x}_{n-2} . \tag{17}$$

The Eq. (17) shows the decimal number of the extracted secret data as $\bar{\mathrm{E}}_1$. Since the length of the extracted secret data is $\bar{\Psi}$, $\bar{\mathrm{E}}_1$ is transformed into $\bar{\Psi}$ bits in binary format. So, the original $\bar{\Psi}$ secret bits can be extracted. For example, we can extract the extracting data [0,1,1,1,3,0,0,0,0,1,0,1,0,1,9,6] according to Fig. 10. First, we use the last two elements, 9 and 6, to compute $\bar{u}_1$. By Eq. (16), $\bar{u}_1$ = 150. Also, if *T*=50, then $\bar{u}_2$ is 200. Second, the same methods can be used to get $\bar{\Psi}$ = 13. Third, according to Eq. (17), $\bar{\mathrm{E}}_1$ equals 5965. Finally, the original string of secret bits [1,0,1,1,1,0,1,0,0,1,1,0,1] that comes from 5965 is obtained.

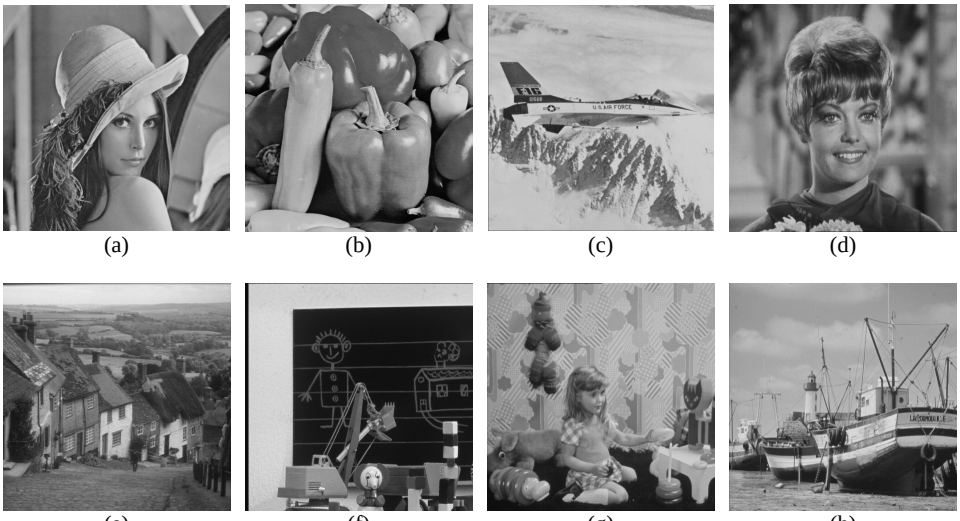
In the recovery stage, the image is recovered as soon as the secret data are extracted. In order to recover image, the index of the closest codeword in C was embedded in the difference number table. In other words, the extracting data are composed of the secret data and the index of the closest codeword. Hence, when the secret data are extracted, the image is recovered using $\bar{u}_1$ of each block with the codebook.

4. EXPERIMENTAL RESULTS

In our experiments, eight grayscale images were used as test images, as shown in Fig. 11. The secret bit data to be embedded were randomly generated. The size of the test image was 512×512. The test images were divided into 16384 non-overlapping 4×4 blocks in the codebook encoding procedure. Codebooks sized 256, 512, and 1024 were used in our experiments. To increase the data capacity of the new codebooks C'_i, where i is the codebook size, a special threshold value was established to modulate the distance of any two codewords. The Java environment was used in this experiment.

The image quality was evaluated using the peak signal-to-noise ratio (PSNR) between the recovered image and the original image. The PSNR formula is stated as follows:

$$\mathrm{PSNR}=10\times\log_{10}\frac{255\times255}{(\frac{1}{\mathrm{H}\times\mathrm{W}})\sum_{i=1}^{\mathrm{H}}\sum_{j=1}^{\mathrm{W}}(x_{ij}-\hat{x}_{ij})^2} . \tag{18}$$



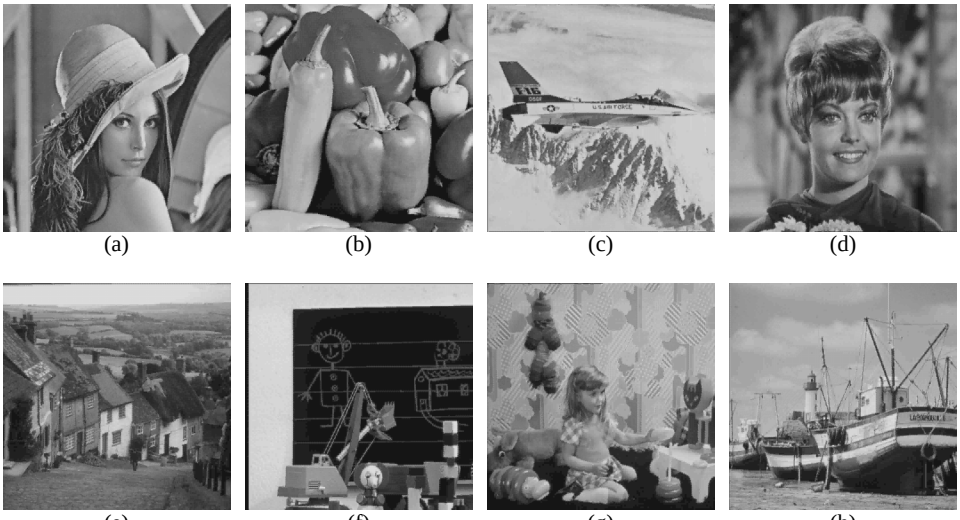
(a) (b) (c) (d)

(e) (f) (g) (h)

Fig. 11. The host images: (a) Lena; (b) Peppers; (c) Jet; (d) Zelda; (e) GoldHill; (f) Toys; (g) Girl; (h) Boat

Here H and W are the height and width of the image, respectively, and x_{ij} and $\hat{x}_{ij}$ are the pixel values of the coordinates (i, j) of the original image and the recovered image, respectively. In the recovery stage, every recovered image for codebook 256 is shown in Fig. 12.

Tables 1-3 show the results of the embedding capacity and compression rate with different threshold T in C'_{256}, C'_{512}, and C'_{1024}, respectively. The compression rate is the ratio of the storage space of the compressed information to the original image. In the embedding stage, the



(a) (b) (c) (d)

(e) (f) (g) (h)

Fig. 12. The recovered images using a codebook with 256 codewords: (a) Lena; (b) Peppers; (c) Jet; (d) Zelda; (e) GoldHill; (f) Toys; (g) Girl; (h) Boat

number of embedded data is decided by the distance between the closest codeword with the original pixel value and the next codeword in codebook C', respectively. In order to increase the embedding capacity, the dis-tance between the two codewords must be larger. Hence, we set T as the distance between the indices of the two codewords. For example, assume the index of the closest codeword is 25 and T is set to 10; the secret data will be embedded in the difference between indices 25 and 35 of the codewords in the codebook. If the result, which is the index value of the closest codeword plus T is greater than the codebook size, subtract T from the index. In other words, the next codeword results from counting T indices successively either after the closest codeword or before the closest codeword. The larger T represents the greater distance between the two codewords. The reason is that the order of the sorted codewords obtained by the PCA algorithm means the variation of every codeword. On the other hand, the larger the value of T is, the greater the compression rate will be.

Normally, the difference is more obvious when the distance is greater. In the entropy encoding procedure, the higher cost is associated with more obvious differences. Thus, in Tables 1-3,

Table 1. Results of the embedding capacity (bits) and compression rate with different thresholds (T) for C'_{256}

Images	Codebook sizes=256									
	T=1			T=50			T=100			T=128	
	Capacity (bits)	Rate		Capacity (bits)	Rate		Capacity (bits)	Rate		Capacity (bits)	Rate
Lena	769,654	0.69		1,110,834	0.81		1,252,309	0.82		1,336,640	0.84
Peppers	766,200	0.68		1,121,658	0.81		1,264,040	0.82		1,339,335	0.83
Jet	730,403	0.68		1,163,924	0.82		1,313,995	0.84		1,378,163	0.85
Zelda	763,103	0.68		1,092,625	0.80		1,240,907	0.82		1,328,848	0.83
GoldHill	759,414	0.68		1,098,370	0.81		1,261,544	0.82		1,347,293	0.84
Toys	679,981	0.62		1,265,824	0.81		1,400,008	0.83		1,442,097	0.84
Girl	767,018	0.68		1,128,801	0.81		1,261,597	0.82		1,339,641	0.83
Boat	799,421	0.70		1,144,511	0.81		1,261,165	0.83		1,336,530	0.83
Averages	754,399	0.68		1,140,818	0.81		1,281,946	0.83		1,356,068	0.84

Table 2. Results of the embedding capacity (bits) and compression rate with different thresholds (T) for C'_{512}

Images	Codebook sizes=512									
	T=1			T=100			T=200			T=256	
	Capacity (bits)	Rate		Capacity (bits)	Rate		Capacity (bits)	Rate		Capacity (bits)	Rate
Lena	728,369	0.68		1,120,092	0.82		1,323,337	0.84		1,404,776	0.86
Peppers	723,735	0.67		1,130,238	0.81		1,327,322	0.84		1,407,312	0.86
Jet	646,687	0.64		1,155,904	0.83		1,345,171	0.85		1,418,648	0.87
Zelda	735,453	0.68		1,110,547	0.81		1,321,187	0.84		1,403,091	0.85
GoldHill	723,142	0.67		1,110,634	0.82		1,325,272	0.84		1,412,183	0.85
Toys	659,711	0.62		1,230,899	0.81		1,395,721	0.83		1,463,948	0.85
Girl	741,165	0.68		1,139,646	0.82		1,324,802	0.84		1,405,423	0.85
Boat	773,366	0.70		1,149,009	0.82		1,320,356	0.84		1,402,067	0.85
Averages	716,454	0.67		1,143,371	0.82		1,335,396	0.84		1,414,681	0.85

Table 3. Results of the embedding capacity (bits) and compression rate with different thresholds (T) for C'_{1024}

Images	Codebook sizes=1024									
	T=1			T=100			T=200			T=512	
	Capacity (bits)	Rate		Capacity (bits)	Rate		Capacity (bits)	Rate		Capacity (bits)	Rate
Lena	680,915	0.66		952,161	0.80		1,123,335	0.83		1,421,500	0.87
Peppers	684,388	0.66		974,098	0.80		1,134,584	0.82		1,421,634	0.86
Jet	633,243	0.64		1,026,532	0.83		1,183,752	0.84		1,441,675	0.88
Zelda	686,951	0.66		928,373	0.78		1,104,011	0.82		1,420,171	0.87
GoldHill	699,174	0.67		944,971	0.79		1,117,393	0.83		1,433,999	0.87
Toys	670,035	0.63		1,047,391	0.80		1,229,706	0.82		1,466,530	0.85
Girl	691,399	0.67		1,003,314	0.82		1,129,514	0.83		1,421,244	0.87
Boat	733,216	0.69		1,035,478	0.82		1,138,901	0.83		1,414,695	0.87
Averages	684,915	0.66		989,040	0.80		1,145,150	0.83		1,430,181	0.87

the values of embedding capacity and compressed rate of $T = 100$ are greater than those of $T = 1$. Also, the ranges of T are different depending on the size of the codebook. The maximum T can be defined as half of the size of the codebook.

In Chang et al.'s method, the embedding procedure is based on a VQ index table. The bit rate with VQ compression equals 0.5 when the codebook size is 256. In contrast, the embedding procedure in the proposed method, which is based on a difference number table that uses the distance between two codewords in the codebook, is different from that used in Chang et al.'s method. The size of the difference number table is 512×512, which is dissimilar to the VQ index table. Therefore, we compared the embedding capacity and PSNR for the two methods using identical compression codes, and the results are shown in Tables 4 and 5 for different codebook sizes. In the same compression codes, the embedding capacity for each host image of the proposed method is nearly 3 bpp, which is obviously more than Chang et al.'s method. In fact, the embedding capacity by replacing the value with an index from the VQ index table is restricted.

In our scheme, we use the product of the distance of the elements from the two codewords to determine the number of embedded secret data. The product is regarded as the combination of the secret bits. For example, assuming that the product equals 1024, ten secret bits can be em-

Table 4. Results of the embedding capacity (bits) and the PSNRs of the recovered images compared with Chang et al.'s method for codebook 256

Images	Codebook sizes=256				
	Chang et al.'s method			Proposed method	
	Capacity(bits)	PSNR		Capacity(bits)	PSNR
Lena	177,344	31.37		769,654	31.31
Peppers	229,469	30.73		766,200	30.68
Jet	393,852	30.58		730,403	30.53
Zelda	199,290	33.38		763,103	33.31
GoldHill	184,319	30.21		759,414	30.2
Toys	402,901	29.92		679,981	29.86
Averages	264,529	31.03		744,793	30.98

Table 5. Results of the embedding capacity (bits) and the PSNRs of the recovered images compared with Chang et al.'s method for codebook 512

Images	Codebook sizes=512				
	Chang et al.'s method		Proposed method		
	Capacity(bits)	PSNR	Capacity(bits)	PSNR	
Lena	143,070	32.25	728,369	32.22	
Peppers	190,470	31.41	723,735	31.37	
Jet	298,592	31.58	646,687	31.55	
Zelda	182,109	34.19	735,453	34.17	
GoldHill	156,415	30.79	723,142	30.79	
Toys	376,149	30.16	659,711	31.13	
Averages	224,468	31.73	702,850	31.87	

Table 6. Results of the PSNRs of the recovered images with different codebook sizes

Images	Codebook sizes=256	Codebook sizes=512	Codebook sizes=1024	
	PSNR	PSNR	PSNR	
Lena	31.31	32.22	33.21	
Peppers	30.68	31.37	32.13	
Jet	30.53	31.55	32.23	
Zelda	33.31	34.17	34.9	
GoldHill	30.2	30.79	31.44	
Toys	29.86	31.13	32.49	
Girl	30.81	31.74	32.28	
Boat	29.33	30.14	30.88	
Averages	30.76	31.64	32.44	

bedded in the 4×4 block. In reality, the average distance of the elements is more than 4, which means that more than two secret bits can be embedded for each pixel. The higher product values indicate greater combinations, so a significantly greater quantity of secret bits can be embedded.

Table 6 compares the PSNRs of the recovered images with different codebook sizes. The larger codebook size has the better image quality. In our scheme, satisfactory levels of image quality can be maintained invariably, even though the embedding capacity is increased, a powerful benefit for a data hiding scheme. The embedding capacity can be as high as 5 bpp at the maximum value of *T*, while maintaining satisfactory quality.

5. CONCLUSIONS

This paper proposes a novel, reversible data hiding scheme for the VQ compression technique in which a series of binary secret bits are embedded into each 4×4 block. In order to increase the embedding capacity, some new strategies were used. The first strategy was to utilize the product of the differences between two similar codewords to embed data. This strategy is more powerful than using single difference to embed data. The second strategy employs an adjustable threshold in the sorted codebook to gain more embedding capacity. By the method that the secret bits and the indices of the sorted codebook in the difference number table, satisfactory quality can be maintained even though the embedding capacity has been increased significantly. For the receiver, the storage space can be economized by the lossless compression technique. Com-

pared to Chang et al.'s method, our scheme has a significantly larger embedding capacity and can recover images with satisfactory quality.

REFERENCES

[1] J. Tian, "Wavelet-based reversible watermarking for authentication," Proceedings of SPIE Photonics West, Security and Watermarking of Multimedia Contents IV, Vol.4675, January, 2002, pp.679-690.

[2] J. Tian, "Reversible Watermarking by Difference Expansion," Proc. of Workshop on Multimedia and Security: Authentication, Secrecy, and Steganalysis, December, 2002, pp.19-22.

[3] Z. Ni, Y. Q. Shi, N. Ansari, and W. Su, "Reversible Data Hiding," In Proc. of International Symposium on Circuits and Systems, Bangkok, Thailand, Vol.2, May, 2003, pp.912-915.

[4] D. M. Thodi and J. J. Rodríguez, "Expansion Embedding Techniques for Reversible Watermarking," IEEE Trans. Image Process, Vol.16, No.3, 2007, pp.721-730.

[5] P. Tsai, Y. C. Hu and H. L. Yeh, "Reversible image hiding scheme using predictive coding and histogram shifting," Signal Processing, Vol.89, Issue 6, June, 2009, pp.1129-1143.

[6] H. C. Wu, C. C. Lee, C. S. Tsai, Y. P. Chu, H. R. Chen, "A high capacity reversible data hiding scheme with edge prediction and difference expansion," Journal of Systems and Software, In Press, Corrected Proof, Available online 11 July, 2009.

[7] H. W. Tseng and C. P. Hsieh, "Prediction-based reversible data hiding.," Information Sciences, Vol.179, No.14, 2009, pp.2460-2469.

[8] N. Ahmed, T. Natarajan and K. R. Rao, "Discrete cosine transform," IEEE Transactions on Computers, Vol.C-23, 1974, pp.90-93.

[9] O. Rioul and P. Duhamel, "Fast algorithms for wavelet transforms," IEEE Transaction on Information Theory, Vol.38, No.2, 1992, pp.569-586.

[10] E. O. Brigham, "The fast Fourier transform," Englewood Cliffs, NJ: Prentice-Hall, 1974.

[11] C. C. Chang, T. S. Chen and L. Z. Chung, "A steganographic method based upon JPEG and quantization table modification," Information Sciences, Vol.141, No.1-2, 2002, pp.123-138.

[12] W. C. Du and W. J. Hsu, "Adaptive data hiding based on VQ compressed images," IEE Proceedings on Vision, Image, and Signal Processing, Vol.150, No.4, 2003, pp.233-238.

[13] 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, 2009, pp.3332-3348.

[14] 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, 2009, pp.1597-1603.

[15] C. C. Chang, C. Y. Lin and Y. Z. Wang, "New image steganographic methods using run-length approach," Information Sciences, Vol.176, No.22, 2006, pp.3393-3408.

[16] B. Yang, Z. M. Lu and S. H. Sun, "Reversible watermarking in the VQ-compressed domain," in: Proceedings of the Fifth International Conference on Visualization, Imaging, and Image Processing, Benidorm, Spain, 2005, pp.289-303.

[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, 2009, pp.1597-1603.

[18] Y. K. Lee and L. H. Chen, "High capacity image steganographic model," IEE Proceedings-Vision, Image, and Signal Processing, Vol.147, No.3, 2000, pp.288-294.

[19] 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, 2003, pp.1583-1595.

[20] C. C. Chang, C. C. Lin, C. S. Tseng and W. L. Tai, "Reversible hiding in DCT-based compressed images," Information Sciences, Vol.177, No.13, 2007, pp.2768-2786.

[21] R. M. Gray and Y. Linde, "Vector quantization and predictive quantizers for Gauss-Markov sources," IEEE Transactions on Communications, Vol.30, No.2, 1982, pp.381-389.

[22] C. C. Chang and T. S. Chen, "A new tree-structured quantization with closest-coupled multipath searching method," Optical Engineering, Vol.36, No.6, 1997, pp.1713-1720.

[23] C. C. Chang, D. C. Lin and T. S. Chen, "An improved VQ codebook search algorithm using principal

component analysis," Journal of Visual Communication and Image Representation, Vol.8, No.1, 1977, pp.27-127.
[24] Z. H. Wang, C. C. Chang, K. N. Chen and M. C. Li, "An encoding method for both image compression and data lossless information hiding," The Journal of Systems and Software, Vol.83, No.11, 2010, pp.2073-2082.
[25] T. S. Chen, C. C. Chang, K. F. Hwang, Digital Image Processing, FLAG, ROC, 2006.

Zhi-Hui Wang

She received the BS degree in software engineering in 2004 from the North Eastern University, Shenyang, China, and the MS degree in software engineering in 2007 from the Dalian University of Technology, Dalian, China. She is currently pursuing her PhD degree in computer software and theory from the Dalian University of Technology, Dalian, China. Her research interests include data hiding, and image processing.

Chin-Chen Chang

He received his BS degree in applied mathematics in 1977 and the MS degree in computer and decision sciences in 1979, both from the National Tsing Hua University, Hsinchu, Taiwan. He received his Ph.D in computer engineering in 1982 from the National Chiao Tung University, Hsinchu, Taiwan. During the academic years of 1980-1983, he was on the faculty of the Department of Computer Engineering at the National Chiao Tung University. From 1983-1989, he was on the faculty of the Institute of Applied Mathematics, National Chung Hsing University, Taichung, Taiwan. From August 1989 to July 1992, he was the head of, and a professor in, the Institute of Computer Science and Information Engineering at the National Chung Cheng University, Chiayi, Taiwan. From August 1992 to July 1995, he was the dean of the college of Engineering at the same university. From August 1995 to October 1997, he was the provost at the National Chung Cheng University. From September 1996 to October 1997, Dr. Chang was the Acting President at the National Chung Cheng University. From July 1998 to June 2000, he was the director of Advisory Office of the Ministry of Education of the R.O.C. From 2002 to 2005, he was a Chair Professor of National Chung Cheng University. Since February 2005, he has been a Chair Professor of Feng Chia University. In addition, he has served as a consultant to several research institutes and government departments. His current research interests include database design, computer cryptography, image compression and data structures.

Pei-Yu Tsai

She received BS degree in Information Management from Shih Chien University in 2008. Currently, she is a graduate student in the department of Computer Science and Information Engineering of National Chung Cheng University, Taiwan. Her research interests include image data hiding, watermarking, and image processing.