High-Capacity Reversible Data Hiding Method for JPEG Images

Chin-Cheng Chang[1,2], Ran Tang[1], Chia-Chen Lin[3*], Wan-Li Lyu[1]

[1] School of Computer Science and Technology, Anhui University, Hefei, 230601, China.
[2] Department of Information Engineering and Computer Science, Feng Chia University, Taichung, 40724, Taiwan.
[3] Department of Information Management and Computer Science, Providence University, Taichung, 43301, Taiwan.

* Corresponding author. Tel.: +886 04-26328001; email: mhlin3@pu.edu.tw
Manuscript submitted December 10, 2017; accepted January 10, 2018.
10.17706/jsw.13.1.1-17

Abstract: This paper presents a reversible data hiding method for JPEG images by combining (7, 4) the Hamming code and improving Huang *et al.*'s data hiding technique. Huang *et al.*'s original method embeds data into JPEG quantized 8×8 block DCT coefficients by histogram shifting and alternating current (AC) coefficients with values of 1 and -1 to embed secret bits. The proposed method here adopts a hybrid embedding strategy to increase maximum embedding capacity. The experimental results show that the proposed method allows for maximum embedding capability while maintaining acceptable image quality.

Key words: Histogram shifting, Joint photographic experts group (jpeg), reversible data hiding, (7, 4) hamming CODE.

1. Introduction

Networking and multimedia technologies have been rapidly developing in recent years. Everyone on the Internet can upload or download all kinds of multimedia information, such as video, audio, and images. Generally, the Joint Photographic Experts Group (JPEG) format is the most popular digital image format in use on the Internet. People can easily copy or edit these images and transmit them to others. Therefore, the security of information is essential during the transmission phase.

Traditional security transmission methods are divided into cryptography based and steganography based methods. Cryptography turns secret information into an unreadable format using mathematical transformations. However, these operations still may be detected by the eavesdropper. Therefore, a technique named steganography, also known as data hiding, was proposed to hide the secret information in another cover medium. Cover medium can be text files, audio, images, and video. The cover medium is called a stego-medium when it is embedded with secret information.

Most data hiding techniques create some irreversible distortion in the cover image that cannot be recovered even if the secret data has been extracted. However, some images must not be damaged or distorted, such as medical and military images. Therefore, a reversible data hiding (RDH) technique was first proposed by Barton [1] in 2000 to address this concern. To date, there are three main strategies that can be adopted to design an RDH method. The first is lossless compression [2], [3], the second is difference expansion (DE) [4]-[6], and the third is histogram shifting (HS) [7]-[10]. Recently, a new data hiding method based on Hamming code (HC) [11], [12] was also developed. This new lossless compression method uses any redundant space that is produced by the lossless compression process to embed information. The difference expansion-based method was first proposed by Tian in 2003 [4]. Tian's method divided the host image into pixel pairs and the difference of the two pixels' values in the pair is expanded to embed a secret message of one bit. Subsequently, Tian's work has been improved in various approaches.

Histogram shifting was first proposed by Ni et al. in 2004 [7], where the histogram of the host image is utilized for embedding secrets. This method changes each pixel one by one, so the visual quality of the stego-image can be higher than 48 dB. Hamming code was first developed by R.W. Hamming in 1980 [13]. Originally, Hamming code made use of error correction where the receiver can use the code to discover one error and correct it. Mao *et al.* in 2016 [12] proposed a new RDH method by rearranging the order of the Hamming codes. Some recent researchers have combined DE and histogram shifting (HS) to achieve better performance, and some methods were even combined with sorting [15].

Most of the RDH methods from the past few years have proposed spatial images to achieve improved visual quality and embedding capacity. Unfortunately, these methods cannot be used directly on JPEG images. First of all, the more information redundancy the host images have, the more data can be embedded. JPEG images are much less redundant than spatial images. JPEG images are the most commonly used digital images in daily life, and as such, the ability to hide data into JPEG images reversibly may have many applications, such as for secure data systems and image authentication. A histogram pair-based RDH method for JPEG images was proposed by Xuan *et al.* in 2007 [16]. They used the best search strategy to shift the histogram of the quantized DCT coefficient to achieve good performance. In order to verify that the data embedding was imperceptible, only the low frequency and intermediate frequency DCT coefficients were selected to embed the data. Xuan et al.'s method was improved by Sakai *et al* in 2008 [17] by using a new adaptive embedding strategy. In 2012, a method based on Huffman code mapping to embedding secret data into JPEG bit-stream was proposed by Qian and Zhang [21]. Later, a reversible data hiding based on EMD for JPEG image is proposed by Kuo *et al.* in 2012 [22]. After that, a higher capacity reversible data hiding method based on EMD in JPEG images was proposed by Zhang *et al.* in 2013 [20]. Subsequently in 2015, Huang *et al*. [18] further proposed a new HS-based RDH method for JPEG images.

In this paper, we study Huang *et al.'s* HS-based RDH method [18] and propose an improvement. Huang et al.'s method used alternating current (AC) coefficients obtained from DCT coefficients with values of 1 and -1 to hide information. But this method does not use all AC coefficients equal to 1 or -1 to carry secret data, and instead uses a block selection strategy. We improved Huang *et al.'s* HS-based RDH method [18] by combining it with (7, 4) Hamming code. The experimental results show that our RDH method has a higher embedding ability, and also provides good image quality.

The rest of this paper is organized as follows. Huang *et al.'s* HS-based RDH method [18], the (7, 4) Hamming code and Zhang *et al.'s* EMD-based JPEG method [20] are demonstrated in Section 2, respectively. The proposed method is introduced in detail in Section 3. Experimental and comparative results are shown in Section 4. Finally, we summarize our conclusions in Section 5.

2. Related Work

In this section, two HS-based RDH methods are first introduced in Subsection 2.1, then a (7, 4) Hamming code-based RDH scheme is introduced in Subsection 2.2. A brief introduction for JPEG compression will provided in Subsection 2.3. Finally, a high capacity reversible data hiding scheme for JPEG images is introduced in Subsection 2.4.

2.1. Two HS-Based RDH Methods

2.1.1. Ni *et al.*'s histogram shifting method

The histogram shifting (HS) method is an RDH algorithm proposed by Ni *et al.* [7]. In their method, the histogram of the image pixel value was counted first. For example, the histogram of the plane image shown in Fig. 1 has the corresponding peak point and zero point marked in the figure. We find the zero point (the gray scale value corresponding to the zero point does not appear in the given image, e.g., h (255) as show in Fig. 1) and the peak point (the gray scale value corresponding to the peak point is the maximum number of pixels in a given image, e.g., h (196) as shown in Fig. 1) in the histogram and take them as a pair.

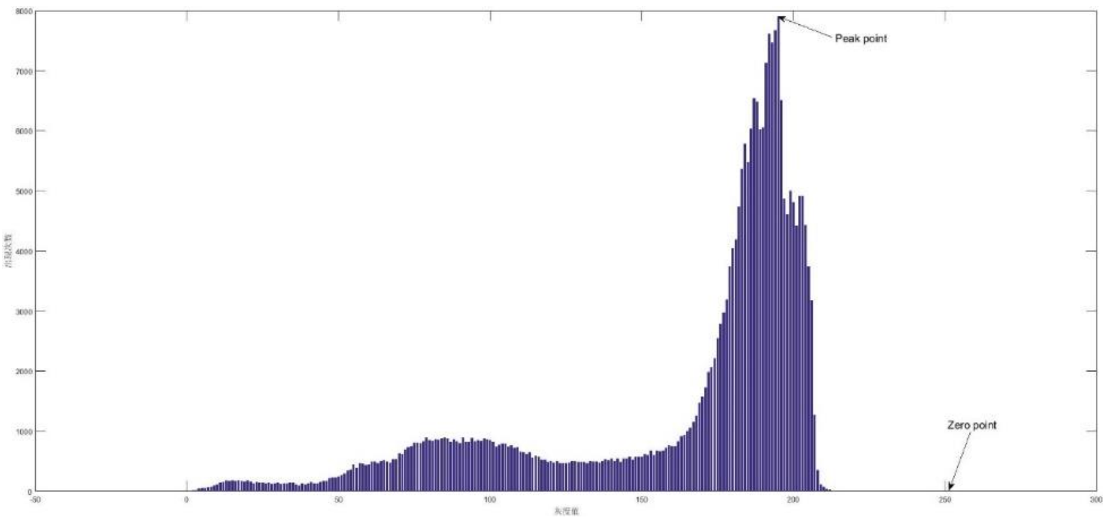


Fig. 1. Histogram of the plane image.

Secondly, the entire image is scanned in an aster scan order. The pixel gray value between 197 (197 included) and 254 (254 included) is incremented by "1." This step is equivalent to shifting all the values in the range [197, 254] of the histogram to the right by one unit, leaving the gray value of 196 empty.

Finally, the entire image is scanned once again in the same order. When the embedding procedure encounters a pixel with a grayscale value of 196, it then checks the to-be-embedded secret sequence. If the corresponding to-be-embedded bit in the sequence is binary "1," then the pixel value is incremented by one. Otherwise, the pixel value remains unchanged.

After that, a stego-image holding the secret data is received. If the receivers receives this stego-image, then they can conduct an extracting procedure to scan the entire image in aster scan order again and find all the pixels with a grayscale value of 196 or 197. If 196 is found, then the secret bit "0" is extracted. If 197 is found, then the secret bit "1" is extracted. Then, the entire image can be scanned in the same order to find all pixels with a grayscale value between 198 and 255 whose pixel values will be reduced by one. The original image can then be recovered without any distortion.

2.1.2. Huang *et al.*'s HS-based RDH method in nonzero AC coefficients in DCT

This HS-based method was proposed by Huang et al. in [18]. In their method, generally speaking, all non-zero AC coefficients in DCT are represented by N = (N_1, N_2, … , N_{m-1}, N_m), where M represents the number of all nonzero AC coefficients in the JPEG image. They concluded that most of the peak points of the AC coefficient histograms are located at points 1 and -1. The embedding algorithm of their method is described as:

$$N_i^{'} = \begin{cases} N_i + sign(N_i) * s & if\ |N_i| = 1 \\ N_i + sign(N_i) & if\ |N_i| > 1 \end{cases} \tag{1}$$

where

$$sign(N_i) = \begin{cases} 1 & if\ N_i > 0 \\ 0 & if\ N_i = 0. \\ -1 & if\ N_i < 0 \end{cases} \tag{2}$$

In (1), $s \in \{0,1\}$, s indicates the secret information bit to be embedded and $N_i^{'}$ denotes the corresponding hidden AC coefficients in the marked JPEG image. In this method, all coefficients are changed at most by one in the embedding process.

The information extraction and image restoration algorithm can be described as follows:

$$s^{'} = \begin{cases} 0 & if\ |N_i| = 1 \\ 1 & if\ |N_i| = 2 \end{cases} \tag{3}$$

$$N_i = \begin{cases} sign(N_i^{'}) & if\ 1 \le |N_i^{'}| \le 2 \\ N_i^{'} - sign(N_i^{'}) & if\ |N_i^{'}| \ge 3 \end{cases} \tag{4}$$

where $s^{'}$ and N_i denotes the extracted secret bit and the restored AC coefficient, respectively.

But Huang et al.'s method has low hiding capacity because it only embeds message bits in AC coefficients with values of 1 and -1.

2.2. An RDH Scheme Based on the (7, 4) Hamming Code

Hamming codes may be the most popular block codes that can dectect and correct a one-bit error in a block. Furthermore, for a given block length, Hamming codes require minimal redundancy to correct any 1-bit errors [14]. The (7, 4) Hamming code transforms four-bit data (D_1, D_2, D_3, D_4) into seven cover bits by combining the other three corresponding check bits (C_1, C_2, C_3) obtained by (D_1, D_2, D_3, D_4), as shown in (5):

$$\begin{cases} C_1 = D_1 \oplus D_2 \oplus D_4 \\ C_2 = D_1 \oplus D_3 \oplus D_4 \\ C_3 = D_2 \oplus D_3 \oplus D_4 \end{cases} \tag{5}$$

where $\oplus$ means XOR (exclusive or) operation. The normal order of the seven cover bits is shown in Fig. 2. In this RDH scheme, the order of these seven cover bits will be rearranged. The Hamming code finds errors by detecting whether each parity bit and its corresponding data bit constitute an even parity. That is to say, there must be an even number of ones in the parity check bit and its corresponding data bit.

R_1	R_2	R_3	R_4	R_5	R_6	R_7
C_1	C_2	D_1	C_3	D_2	D_3	D_4

Fig. 2. The normal form of a (7, 4) Hamming code.

In 2016, Mao *et al.* [12] proposed an RDH scheme based on the (7, 4) Hamming code. Mat et al.'s (7, 4) Hamming code-based RDH scheme rearranges the seven cover bits from their normal form of R= $\left(C_1, C_2, D_1, C_3, D_2, D_3, D_4\right)$ to R=$\left(D_1, D_2, D_3, D_4, C_1, C_2, C_3\right)$ as shown in Fig. 3, where the first four bits $\left(r_1, r_2, r_3, r_4\right)$ consist of four data bits $\left(D_1, D_2, D_3, D_4\right)$ and the last three bits consist of the parity check bits $\left(C_1, C_2, C_3\right)$.

R1	*R2*	*R3*	*R4*	*R5*	*R6*	*R7*
D1	D2	D3	D4	C1	C2	C3

Fig. 3. The rearranged form of (7, 4) Hamming code in the proposed method.

This method used Y(R) as a new judgment. Y(R) = $\left(y_1, y_2, y_3\right)$ is defined as (6):

$$Y(R) = \begin{cases} y_1 = r_1 \oplus r_2 \oplus r_4 \\ y_2 = r_1 \oplus r_3 \oplus r_4 \\ y_3 = r_2 \oplus r_3 \oplus r_4 \end{cases} \tag{6}$$

For a given stream with seven bits, R=$\left(r_1, r_2, r_3, r_4, r_5, r_6, r_7\right)$; if the calculated three-bit vector $Y(R)$ = $\left(y_1, y_2, y_3\right)$ is equal to (0, 0, 0), then the bit stream R is classified as a perfect stream. A new bit stream R', which is generated by flipping only one bit of the stream R, will generate another three-bit vector $Y(R')$. The location of the flipped bit here is referred to as the error location. The relationship between the error location and the corresponding vector of $Y(R')$ is shown in Table 1.

Table 1. The Relationship between the Error Location and the Vector of $Y(R')$

Error location	$Y(R')$
error free	000
1	110
2	101
3	011
4	111
5	100
6	010
7	001

Table 1 demonstrates that for a perfect stream, R=$\left(r_1, r_2, r_3, r_4, r_5, r_6, r_7\right)$, its error location and the corresponding vector of $Y(R')$ are a one-to-one mapping. Therefore, the receiver can easily find the error location by looking up the table. Hamming codes can be used not only to correct one bit error, but also to hide information. From Table 1, we can see that a perfect stream has eight error location signals where three $(\log_2 8 = 3)$ bits of information with only one bit flipped can be embedded. In addition, the receiver can easily extract the information, and the original image can be recovered without any distortion.

2.3. A Brief Introduction to JPEG Compression

The main steps of the JPEG compression process [19] for grayscale images are given in Fig. 4. Color images can be generally regarded as multiple grayscale images. By applying 2D DCT to the non-overlapping 8×8 blocks of the original image, the spatial domain signal is transformed into a frequency domain signal. Then, the coefficients obtained by DCT are fed to the quantizer and are quantized by the predefined quantization tables. The quantized DCT coefficients are scanned in zig-zag order and the DC coefficients are pre-compressed by differential pulse code modulation (DPCM). The AC coefficients are pre-compressed through running length encoding (RLE). Finally, by applying the Huffman code, the final compressed bit stream will be obtained by processing the symbol string. After enclosing the header file, the final JPEG file is obtained.

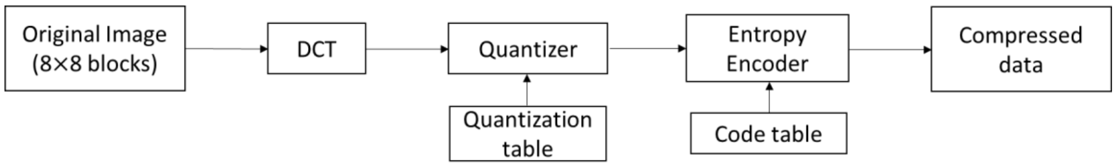


Fig. 4. Flowchart of JPEG compression encoding process.

2.4. A High Capacity Reversible Data Hiding Scheme for JPEG Images

In 2013, a high capacity reversible data hiding method for JPEG images was proposed by Zhang et al. [20]. In their method, they assume one dimensional sequence $\{l_0, l_1, l_2, ..., l_{63}\}$ can be obtained by scanning each DCT coefficients block using zigzag scan. And, this method adopts mid-frequency $\{l_4, l_5, l_6, ..., l_{36}\}$ as embedded region to carry secret bit by using EMD method. For each coefficient-pairs (every two consecutive coefficients) in mid-frequency whose values equals to P will carry secret data, where P is a candidate value. Let l_i and l_j be a pair of coefficients, $l_i^{'}$ and $l_j^{'}$ which are the corresponding coefficients when secret message is embedded. In Zhang *et al.'s* method, P has three cases: P>0, P<0, and P=0. The embedding rules for the three cases are designed as fallows according to the value of P:

(1) When *P*>0, for a pair (l_i, l_j), if $l_i = l_j$=P, which will carry two secret bits. These two bits are converted to a decimal g ($0 \le g \le 3$) first. Then the extraction function f_t is defined as

$$f_t(I, J) = (I + 2 \times J) \bmod 4 \tag{7}$$

g is hidden into $f_t(I, J)$ when $(l_i^{'}, l_j^{'})$ meet the following circumstances: if g= $f_t(l_i, l_j)$, $l_i^{'}$=P and $l_j^{'}$=P; if g= $f_t(l_i+1, l_j)$, $l_i^{'}$=P+1 and $l_j^{'}$=P; if g= $f_t(l_i, l_j+1)$, $l_i^{'}$=P and $l_j^{'}$=P+1; if g= $f_t(l_i+1, l_j+1)$, $l_i^{'}$=P+1 and $l_j^{'}$ =P+1.

(2) When *P*<0, for a pair (l_i, l_j), if $l_i = l_j$=P, which will carry two secret bits, and the corresponding decimal form is g ($0 \le g \le 3$). The extraction function the extraction function f_t is defined as (7). $(l_i^{'}, l_j^{'})$ is determined by the following conditions: if g= $f_t(l_i, l_j)$, $l_i^{'}$=P and $l_j^{'}$=P; if g= $f_t(l_i-1, l_j)$, $l_i^{'}$ =P-1 and $l_j^{'}$=P; if g= $f_t(l_i, l_j-1)$, $l_i^{'}$=P and $l_j^{'}$=P-1; if g= $f_t(l_i-1, l_j-1)$, $l_i^{'}$=P-1 and $l_j^{'}$=P-1.

(3) When *P*=0, for a pair (l_i, l_j), if $l_i = l_j$ = P = 0, will embed three secret bits. The corresponding decimal form *g* lies between 0 and 7, and the extraction function f_s is defined as (8) according to embedding method [5].

$$f_s(I, J) = (I \times a + J \times b) \bmod 8 \tag{8}$$

where *a* and *b* are a pair of weighting coefficients. *g* is hidden into $f_s(I, J)$ according to algorithm [5], a and b are generated by different random seeds. Let *c*= $(g - f_t(l_i, l_j)) \bmod 8$, then $l_i^{'}$ and $l_j^{'}$ can be obtained by *c*.

Secret data can be embedded into the mid-frequency coefficients according to the above rules. Other cases of a pair of coefficients will not be used to carry secret data.

Zhang *et al.'s* method [20] can achieve a very high payload, but their method modifies almost zero coefficients in mid-frequency of each DCT blocks. In other words, Zhang *et al.'s* method is very fragile in terms of security.

3. Proposed High-Capacity RDH Method

In this section, a high-capacity JPEG-based RDH method is presented. Our proposed scheme combines the (7, 4) Hamming code with a HS method, resulting in a higher embedding capacity than Huang *et al.'s* RDH [18]. Flowcharts for the embedding and extraction phases of our proposed method are shown in Fig. 5 and Fig. 7, respectively.

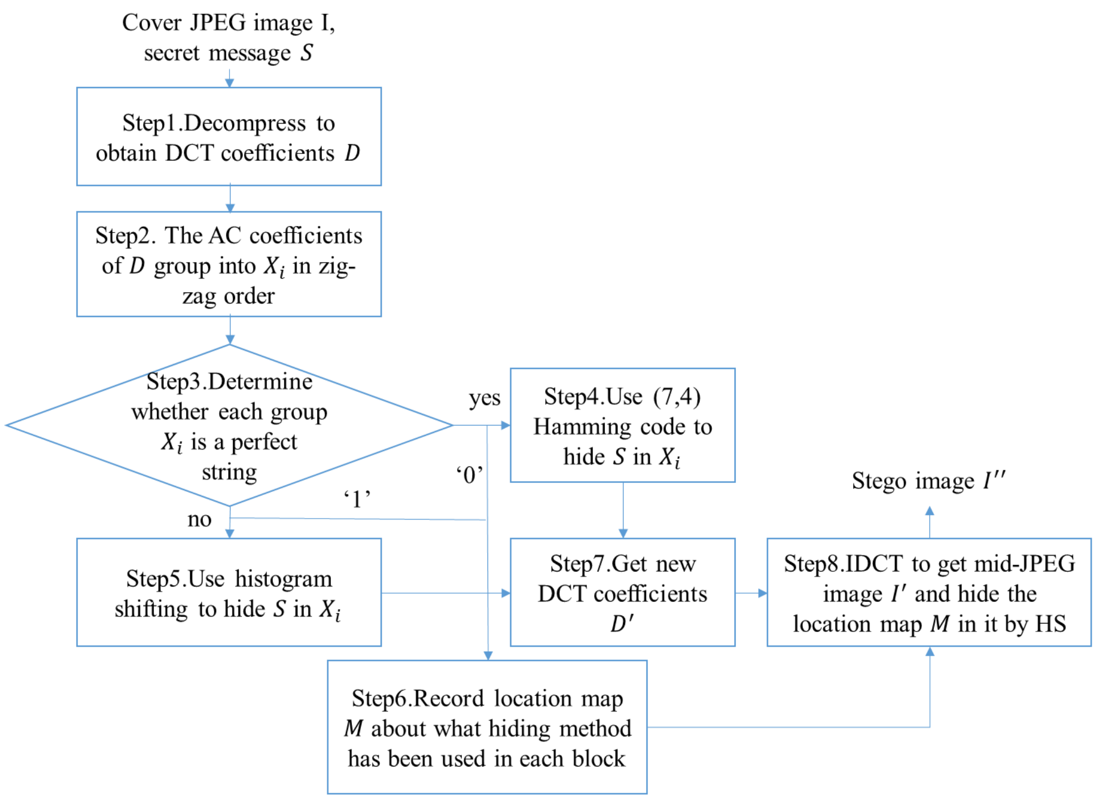

Cover JPEG image I, secret message S
Step1.Decompress to obtain DCT coefficients D
Step2. The AC coefficients of D group into X_i in zig-zag order
Step3.Determine whether each group X_i is a perfect string
yes
'0'
'1'
no
Step4.Use (7,4) Hamming code to hide S in X_i
Stego image I''
Step5.Use histogram shifting to hide S in X_i
Step7.Get new DCT coefficients D'
Step8.IDCT to get mid-JPEG image I' and hide the location map M in it by HS
Step6.Record location map M about what hiding method has been used in each block

Fig. 5. Flowchart of the embedding phase.

3.1. Embedding Phase

In the embedding phase, a cover JPEG image needs to be decoded to get the quantized DCT coefficients. Then, we check the AC coefficients at every seven bits as a group. If a group is a perfect string as defined in Subsection 2.2, it is embedded with a secret using the (7, 4) Hamming code; otherwise, the HS method introduced in Subsection 2.1.1 is used to embed the secret. Meanwhile, we record the adopted embedding method of each group using a binary location map. After this has been done, we can obtain a new DCT coefficient carrying a secret message. We use these new DCT coefficients to perform inverse DCT operations to obtain a mid-JPEG image. Finally, the location map is embedded in the mid-JPEG image to generate the stego-image. The details of the embedding phase are described below.

Inputs: Cover JPEG image I , secret message S .

Outputs: Stego JPEG image $I^{''}$.

Step 1: Decode cover JPEG image I , as described in Subsection 2.3, to get the quantized DCT coefficient D .

Step 2: The AC coefficients of D are grouped into X_i in zig-zag order. Each group consists of seven consecutive numbers. For example, in Fig. 6(a), 1-7 constitute the first group, 8-14 constitute the second group, ... , and 57-63 constitute the ninth group.

Step 3: For each seven-tuple $X_i = (x_1, x_2, x_3, x_4, x_5, x_6, x_7)$, we denote X_i in two parts, $R = (r_1, r_2, r_3, r_4, r_5, r_6, r_7)$ which consists of the LSB of each element, and $P = (p_1, p_2, p_3, p_4, p_5, p_6, p_7)$ which is composed of the non-LSB parts of each element. We take R into (6) to calculate the vector value of $Y(R)$. If $Y(R)$ is equal to (0, 0, 0), then R is a perfect stream as defined in Subsection 2.2, and the (7, 4) Hamming code is used to hide the data for this seven-tuple X_i , after which we continue to Step 4. Otherwise, R is a non-perfect stream, HS is used to hide the data for this seven-tuple X_i and then we continue to Step 5. For one 8×8 DCT block, we need to make nine judgments.

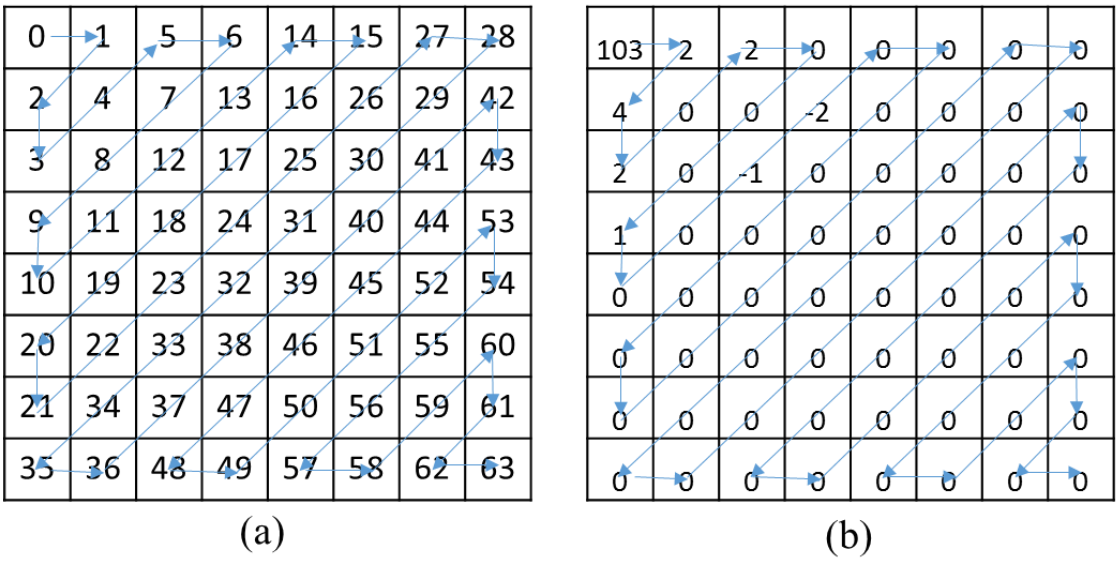


Fig. 6. (a) Zig-zag scan order in DCT. (b) Quantized DCT coefficients in an 8×8 block.

Step 4: For a vector cover seven-tuple $X_i = (x_1, x_2, x_3, x_4, x_5, x_6, x_7)$, whose R and P are defined as Step 3, and secret message S, we first convert S to its corresponding decimal value T (T<8). If T=0, then the stego perfect stream $R^{'} = (r_1^{'}, r_2^{'}, r_3^{'}, r_4^{'}, r_5^{'}, r_6^{'}, r_7^{'})$ is equal to the cover seven-tuple R; otherwise, the stego perfect stream $R^{'} = (r_1^{'}, r_2^{'}, r_3^{'}, r_4^{'}, r_5^{'}, r_6^{'}, r_7^{'})$ is generated by flipping the T^{th} bit of R. Finally, the vector of the stego seven-tuple $X^{'} = (x_1^{'}, x_2^{'}, x_3^{'}, x_4^{'}, x_5^{'}, x_6^{'}, x_7^{'})$ is generated by re-combination of P and $R^{'}$.

For example: Shown in Fig. 6(b) is an 8×8 block of quantized DCT coefficients. We found the first seven-tuple of the AC coefficient in Fig. 6(b) in zig-zag scan order as $X = (2,4,2,0,2,0,0)_{10}$ = $\left(010,100,010,000,010,000,000\right)_2$, and the LSB of each element constitutes $R = \left(0,0,0,0,0,0,0\right)_2$. We take R into (2) and find that R is a perfect string. The corresponding P to R is $P = \left(01,10,01,00,01,00,00\right)_2$. Assuming the secret data is $S = \left(1,1,0\right)_2 = 6_{10}$, $R^{'} = \left(0,0,0,0,0,1,0\right)_2$ can be easily generated by flipping the 6th bit of R. Then, the vector of the seven-tuple stego bits $X^{'} = \left(010,100,010,000,010,001,000\right)_2$ = $\left(2,4,2,0,2,1,0\right)_{10}$ can be obtained by combining P and $R^{'}$.

Step 5: For a vector of cover seven-tuple $X_i = (x_1, x_2, x_3, x_4, x_5, x_6, x_7)$ and binary secret bit streams $S = (s_1, s_2, ..., s_i), s_i \in \{0,1\}$, we first read x sequentially. If $x_i (1 \le i \le 7)$ is a nonzero number, then we calculate $x_i^{'}$ using (1) and (2). There, a binary secret bit is corresponds to s in (1); otherwise, x remains unchanged, and $x_i^{'}$ is equal to x, which is zero. Then, we can get the stego seven-tuple $X^{'} = (x_1^{'}, x_2^{'}, x_3^{'}, x_4^{'}, x_5^{'}, x_6^{'}, x_7^{'})$ to carry the secret message.

For example: Shown in Fig. 6(b) is an 8×8 block of quantized DCT coefficients. We found the second seven-tuple of AC coefficients in Fig. 6(b) in a zig-zag scan order as X = (0, 1, 0, 0, -1, -2, 0), and the LSB of each element is a non-perfect string. We take it as a cover seven-tuple, and assume the secret bits are $S = \left(1,0\right)_2$. Reading X sequentially, sequentially, x_1, x_3, x_4, x_7 are 0 with no need for change; then, according to (1), $\left|x_2\right| = 1$ and $s_1 = 1$, so $x_2^{'} = x_2 + sign(x_2) * s_1 = 1 + (1) * 1 = 2$; $\left|x_5\right| = 1$ and $s_2 = 0$, so $x_2^{'} = x_2 + sign(x_2) * s_2 = 1 + (1) * 0 = 1$; $\left|x_6\right| > 1$, so $x_6^{'} = x_6 + sign(x_6) = -2 + (-1) = -3$. After this is complete, we can obtain the seven-tuple stego bits $X^{'} = \left(0,2,0,0,-1,-3,0\right)_{10}$.

Step 6: We used”0” to represent choosing the (7, 4) Hamming code for hiding. “1” means that HS method was used for hiding the secret message. Therefore, one 8×8 DCT block will generate a 9-bit location map (the whole JPEG image was formed on a large location map M according to the raster scan order of DCT blocks). This location map M is used to record the embed method of each block.

Step 7: Repeat Step 3 to Step 6 until all DCT blocks of D are processed. Then a new DCT coefficient $D^{'}$ carrying the secret message will be obtained.

Step 8: After the above step is completed, we can get a new DCT coefficient $D^{'}$ to carry secret data and a location map M to record which hiding method has been used in each block. Then, a new JPEG image $I^{'}$ can be generated by an inverse DCT operation using the new DCT coefficient $D^{'}$. This new JPEG image is named the mid-JPEG image. Then, the location map carrying the embed method will be embedded in this mid-JPEG image using the HS method.

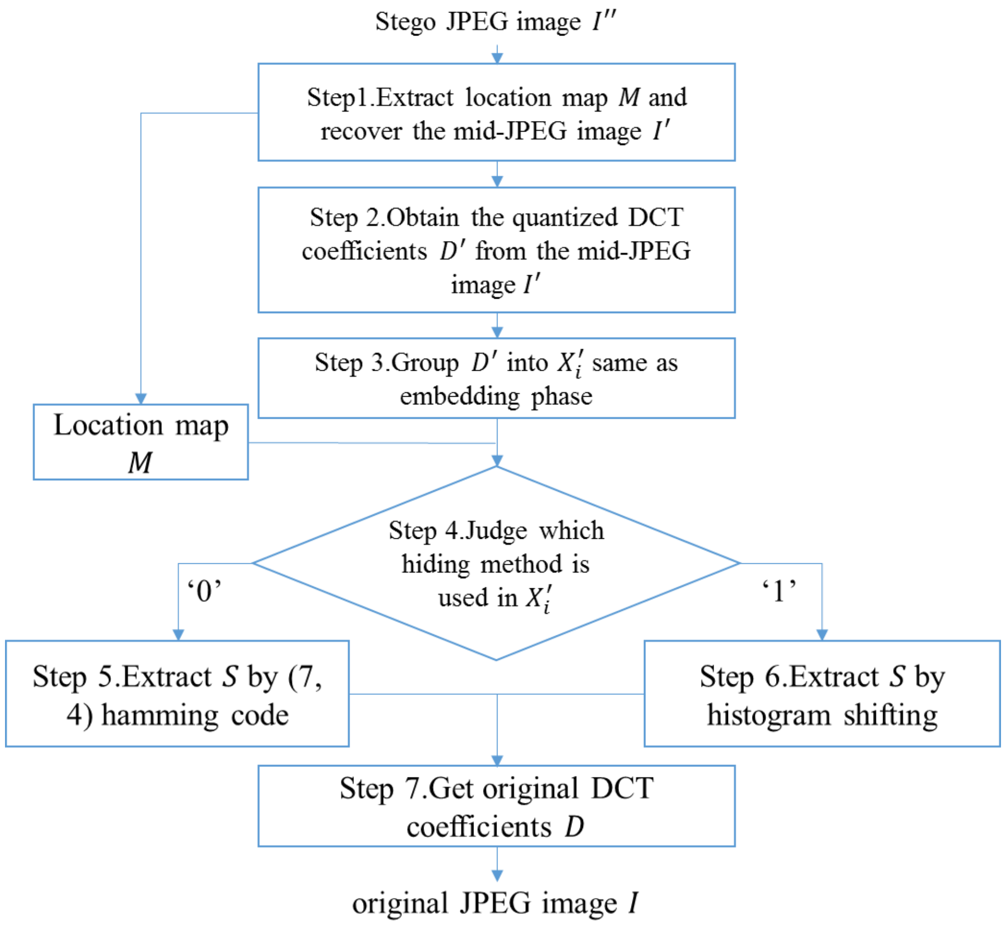
We first generate the mid-JPEG image I’s histogram $H(i)$. In the histogram $H(i)$, we find the maximum point $h(a)(a \in [0,255])$ and the minimum zero point $h(b)(b \in [0,255])$. If $h(b)$ > 0, we record the all coordinates $\left(x,\ y\right)$ of those pixels and the grayscale value b as overhead record information. Then we set

$h(b) = 0$. Without a loss of generality, we assume $a < b$. Move all parts of histogram $H(i)$ into the range of $(a,\ b)$ to the right by one unit. This means that all the pixels whose grayscale values are $i \in (a,\ b)$ are increased by 1. Then we scan the image, and if there is a pixel whose grayscale value is equal to a, we check the to-be-embedded location map M's bit. If the bit equals "1," then the pixel grayscale value is changed to a+1. If the bit is "0," then the pixel value is retained as a. Finally, we can get the stego JPEG image I''.

After all steps are finished, a stego JPEG image I'' carrying location map M can be obtained. This stego JPEG image I'' also carries secret message S.

3.2. Extraction and Restoration

The extraction and restoration phases are shown in Fig. 7. Secret message S is extracted from stego JPEG image I''.



Stego JPEG image I''
Step1.Extract location map M and recover the mid-JPEG image I'
Step 2.Obtain the quantized DCT coefficients D' from the mid-JPEG image I'
Step 3.Group D' into X_i' same as embedding phase
Location map M
Step 4.Judge which hiding method is used in X_i'
'0'
'1'
Step 5.Extract S by (7, 4) hamming code
Step 6.Extract S by histogram shifting
Step 7.Get original DCT coefficients D
original JPEG image I

Fig. 7. Flowchart of the extraction and restoration phase.

Inputs: Stego JPEG image I''.

Outputs: Secret message S and original JPEG image I.

Step 1: We first generate stego-JPEG image I'''s histogram $H(i)$. Assume the grayscale value of the maximum point and the minimum points are a and b, respectively. Without loss of generality, assume $a < b$. We scan the stego image I'' in the same scan order as in embedding procedure. When we meet a pixel whose grayscale value is a +1, a location map bit "1" can be extracted. When we meet a pixel whose value is

a, a bit "0" can be extracted. Then we scan image I'' again. For whole pixels whose grayscale values satisfy $i \in (a,\ b]$, pixel value i is subtracted by 1. If there is overhead record information found in the extracted data, then we set the pixel grayscale value (whose coordinate $(x,\ y)$ is saved in the overhead) as b.

After that, we can get location map M and recover mid-JPEG image I'.

Step 2: The quantized DCT coefficients D' carrying the secret message can be generated from mid-JPEG image I'.

Step 3: The AC coefficients of D' are grouped into $X' = (x_1', x_2', x_3', x_4', x_5', x_6', x_7')$ in zig-zag order, and each group consists of seven consecutive numbers, just like the embedding phase.

Step 4: Via location map M, we know which hiding method has been used for X_i' If the corresponding location of X_i' in location map M is "0," that means the (7, 4) Hamming code is used for X_i'. If so, then go to Step 5. Otherwise, histogram shifting is used for X_i', and Step 6 is next.

Step 5: For a vector of stego seven-tuple $X' = (x_1', x_2', x_3', x_4', x_5', x_6', x_7')$ we denote X' in two parts, one of which is $R' = (r_1', r_2', r_3', r_4', r_5', r_6', r_7')$, which consists of the LSB of each element, and the other part is $P = (p_1, p_2, p_3, p_4, p_5, p_6, p_7)$, which is composed of non-LSB parts of each element. Then we calculate the value of $Y(R')$ using (6). If $Y(R') = (0,0,0)$, then the secret data $S = (0,0,0)_2$, and the cover perfect stream R is equal to R'; otherwise, we look up with Table 1 for $Y(R')$ to obtain the error location T, then convert T to 3-bits binary form T', T' is the secret data. The cover perfect stream R can be obtained by flipping the T^{th} bit of R'. Finally, reconstruct the cover seven-tuple $X_i = (x_1, x_2, x_3, x_4, x_5, x_6, x_7)$ by P and R, and adjust their corresponding elements for re-combination.

For example: When we get the stego seven-tuple $X' = (2,4,2,0,2,1,0)_{10}$, we divide X' = $X' = (2,4,2,0,2,1,0)_{10} = (010,100,010,000,010,001,000)_2$ into two parts, $P = (01,10,01,00,01,00,00)_2$ and $R' = (0,\ 0,\ 0,\ 0,\ 0,\ 1,\ 0)_2$. Then get $Y(R')$ = (0, 1, 0) by (6) because $y_1 = r_5 \oplus r_1 \oplus r_2 \oplus r_4 = 0 \oplus 0 \oplus 0 \oplus 0 = 0$; $y_2 = r_6 \oplus r_1 \oplus r_3 \oplus r_4 = 1 \oplus 0 \oplus 0 \oplus 0 = 1$; $y_3 = r_7 \oplus r_2 \oplus r_3 \oplus r_4 = 0 \oplus 0 \oplus 0 \oplus 0 = 0$. As the value of $Y(R') = (0,1,0) \neq (0,0,0)$, we can look up Table 1 and find that the error location is 6. Therefore, secret data $S = 6_{10} = (1,1,0)_2$, and then flipping the 6^{th} bit of R' gets $R = (0,0,0,0,0,0,0)_2$. At last, we reconstruct the cover seven-tuple $X = (010,100,010,000,010,000,000)_2 = (2,4,2,0,2,0,0)_{10}$ by combining the corresponding elements P and R.

Step 6: For a vector of stego seven-tuple $X' = (x_1', x_2', x_3', x_4', x_5', x_6', x_7')$. Read X' sequentially, according to (3), if $|x_i'| = 1$, a secret bit 0 can be extracted; if $|x_i'| = 2$, a secret bit 1 can be extracted. After this, secret data can be completely extracted. Then read X' sequentially again, and according to (4), x_i' can be completely restored to the original x_i and we can obtain the cover seven-tuple $X_i = (x_1, x_2, x_3, x_4, x_5, x_6, x_7)$.

For example: When we get the stego seven-tuple $X' = (0,2,0,0,-1,-3,0)$, we first do a sequential scan

$X^{'}$ to find $x_2^{'} = 2$ and $x_5^{'} = -1$, which satisfy the requirements of (3), which provides secret data $S = (1,0)$. Then, we sequentially scan $X^{'}$ again according to (4), and we can calculate that $x_2 = sign(x_2^{'}) = 1$; $x_5 = sign(x_5^{'}) = -1$; $x_6 = x_6^{'} - sign(x_6^{'}) = -3 - sign(-3) = -2$, and x_1, x_3, x_4, x_7 are also 0. Finally, the cover seven-tuple $X = (0,1,0,0,-1,-2,0)$ is obtained.

Step 7: Repeat Step 4 to Step 6 until all $X_i^{'}$ of DCT coefficients $D^{'}$ are processed. Then we can get the original DCT coefficient D. Through D, we obtain the original JPEG image I by an inverse DCT procedure.

When all the steps are finished, original JPEG image I can be restored without any distortion and the whole secret message S can be extracted.

4. Experimental Results

In this section, several experiments were performed to demonstrate the performance of our proposed method. Eight grayscale images of size 512×512, including Lena, Peppers, Plane, Baboon, Blonde, Boat, Elaine, and Lake were used as test images as shown in Fig. 8. The 2016a version of the MATLAB programming language running on an Intel(R) Core(TM) i7-4790 was used to implement each method. We used the MATLAB function randint() to generate pseudo random numbers as secret messages. Two metrics were used to demonstrate the performance of the proposed hiding method: the peak signal-to-noise ratio (PSNR) of the stego JPEG images and actual data embedding capacity (pure payload). PSNR is calculated between the original JPEG image and the stego JPEG image and is used as a measure to evaluate the visual quality of the stego JPEG image. Pure payload is used to evaluate the maximum data embedding capacity of the JPEG image.

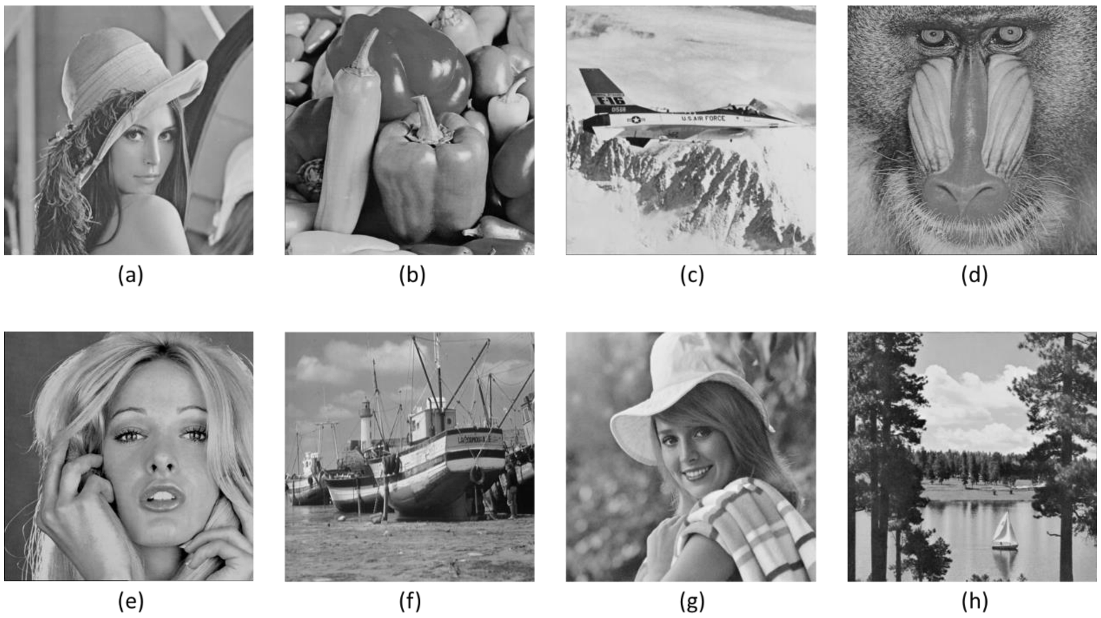


Fig. 8. Eight test images: (a) Lena, (b) Peppers, (c) Plane, (d) Baboon, (e) Blonde, (f) Boat, (g) Elaine, (h) Lake.

Some statistical results for these eight images are shown in Table 2. All the images are uncompressed images (.tiff). We compressed them with different quality factors (i.e., QF = 70, 80, 90, and 100). Then, the ratio of the perfect string (PS) to a non-perfect string (NPS) was gathered for each quantized DCT block (described in Section 3.2) as shown in Table 2. Experiment results demonstrated that there are many PS in the quantized DCT coefficients. The number of PS decreases with an increase of quality factors. Table 2 shows that the highest average number of PS can reach two-thirds of the whole image when the quality factor is 80. But even when the quality factor is 100, there are still a considerable number of perfect strings for hiding information. One PS can be used to carry three bits of data if we use the (7, 4) Hamming code. The more PS there is, the more information that can be embedded.

Table 2. Average PS and NPS Quantity and Proportional Analysis

QF	PS	NPS	PS/NPS
70	24867	11997	2.0729
80	21980	14885	1.4767
90	14974	21890	0.6841
100	4632	32232	0.1437

In our experiments, secret information bits are generated by randomization. All JPEG images are generated from the original image through a standard JPEG compression process and a standard quantization table (when QF=70, 80, 90 and 100). In order to demonstrate the performance of our method, we chose one state-of-the-art RDH method in the JPEG field for comparison. This method was proposed in [18]. Huang et al.'s RDH method [18] is based on the modification of quantized DCT coefficients and a block selection strategy.

Corresponding to different QF values (QF=70, 80, 90 and 100), the experimental results are shown in Tables 3-6, respectively. The first column of the table is the test image, and the second column is Huang et al.'s method. The third column is the proposed method, where SAEB means the same amount of embedding bits as Huang *et al.'s* method when their method achieves a pure payload. The last column is the pure payload of our proposed method, and the statistics for each column includes the PSNR values. As shown in Tables 3-6, we can clearly see that the proposed method has a higher pure payload. When our method carries the same amount of information equal to the pure payload of Huang et al.'s method, we can also have a good PSNR. Moreover, with the increase of QF, the PSNR of our proposed method will continually improve. Especially when QF=100, our proposed method not only has a better pure payload, but also has better visual quality when compared to Huang et al.'s method. The experimental results of the test image Peppers on the visual quality of our proposed method is shown in Fig. 9, and has good picture quality. The above experimental results indicate that our proposed method achieved a higher pure payload and good image quality.

For a more detailed comparison with [20], we choose "Boat" test image, as shown in Fig. 8. (f), with Q-factor 80 as the cover image. Comparison results are shown in Table 7. Obviously, the PSNRs of our method are better than those of [21] and [20] under the same hiding capacity. Zhang et al.'s method uses almost zero coefficients in mid-frequency of each DCT blocks to carry secret data, so their payload can achieve more than 130,000 bits whereas have very low visual quality. By contrast, the proposed method only changes a little zero coefficients for each blocks to enhance the hiding capacity while maintain good visual quality. For a normal JPEG image, most of it's coefficients in mid-frequency of each DCT blocks are 0. That means Zhang *et al.'s* method is very fragile in terms of security, and it is quite easily perceived by eavesdroppers.

Table 3. PSNRs and Pure Payload for Eight Test JPEG Images (QF=70)

Images	Huang *et al.'s* Method		Proposed Method		Proposed Method	
	Pure payload	PSNR (dB)	SAEB	PSNR (dB)	Pure payload	PSNR (dB)
Lena	17778	41.39	17778	38.11	100685	31.20

Peppers	21764	40.78	21764	37.38	100098	34.65
Plane	19742	40.13	19742	37.50	99781	30.46
Baboon	40466	34.02	40466	34.18	90388	31.47
Blonde	27354	38.23	27354	36.04	96929	30.93
Boat	25109	38.29	25109	36.49	96556	30.85
Elaine	26842	40.42	26842	36.82	98989	31.56
Lake	26872	38.14	26872	37.07	97039	33.28
Average	25740.88	38.92	25740.88	36.70	97558.13	31.80

Table 4. PSNRs and Pure Payload for Eight Test JPEG Images (QF=80)

Images	Huang *et al.*'s Method		Proposed Method		Proposed Method	
	Pure payload	PSNR (dB)	SAEB	PSNR (dB)	Pure payload	PSNR (dB)
Lena	22603	42.42	22603	39.08	98438	33.24
Peppers	28291	41.55	28291	39.36	98048	35.85
Plane	24220	41.33	24220	38.50	97824	32.46
Baboon	47784	34.59	47784	35.27	86891	33.44
Blonde	34121	38.88	34121	37.14	93965	33.25
Boat	31147	39.14	31147	37.60	91771	33.01
Elaine	35494	40.13	35494	38.07	96298	34.33
Lake	33220	38.72	33220	38.70	93799	35.89
Average	32110.00	39.59	32110.00	37.96	94629.25	33.93

Table 5. PSNRs and Pure Payload for Eight Test JPEG Images (QF=90)

Images	Huang *et al.*'s Method		Proposed Method		Proposed Method	
	Pure payload	PSNR (dB)	SAEB	PSNR (dB)	Pure payload	PSNR (dB)
Lena	34431	44.14	34431	42.92	93542	38.64
Peppers	44169	41.82	44169	44.41	87997	41.40
Plane	33439	43.52	33439	42.23	92826	37.80
Baboon	60338	36.32	60338	37.96	81034	37.24
Blonde	47883	40.25	47883	40.62	87743	38.40
Boat	45457	40.80	45457	41.19	80372	38.70
Elaine	57774	39.14	57774	40.78	86441	39.34
Lake	50976	39.26	50976	40.29	84166	37.93
Average	46808.38	40.66	46808.38	41.30	86765.13	38.68

Table 6. PSNRs and Pure Payload for Eight Test JPEG Images (QF=100)

Images	Huang *et al.*'s Method		Proposed Method		Proposed Method	
	Pure payload	PSNR (dB)	SAEB	PSNR (dB)	Pure payload	PSNR (dB)
Lena	70021	52.74	70021	54.83	75052	54.51
Peppers	49069	52.72	49069	57.92	56519	57.24
Plane	70770	53.01	70770	54.54	75980	54.24
Baboon	25199	53.97	25199	58.74	35990	57.42
Blonde	45788	52.93	45788	55.93	53880	55.27
Boat	46147	52.83	46147	55.61	54243	54.38
Elaine	38920	53.00	38920	57.30	48327	56.32
Lake	40407	53.31	40407	55.93	49169	54.86
Average	48290.13	53.06	48290.13	56.35	56145.00	55.53

Table 7. PSNRs Comparisons between the Proposed Method and Kuo *et al.'s* Method [22], Zhang *et al.'s* Method[20] (QF=80)

Payload(bit)	Kuo *et al.'s* Method [22]	Zhang *et al.'s* Method [20]	Proposed Method
5000	42.5395	44.5976	45.5614
12288	31.7602	39.7804	41.5250
26214	-	37.0356	38.2714
52429	-	32.2727	35.3318
78643	-	31.0415	33.6089
131072	-	29.1812	-

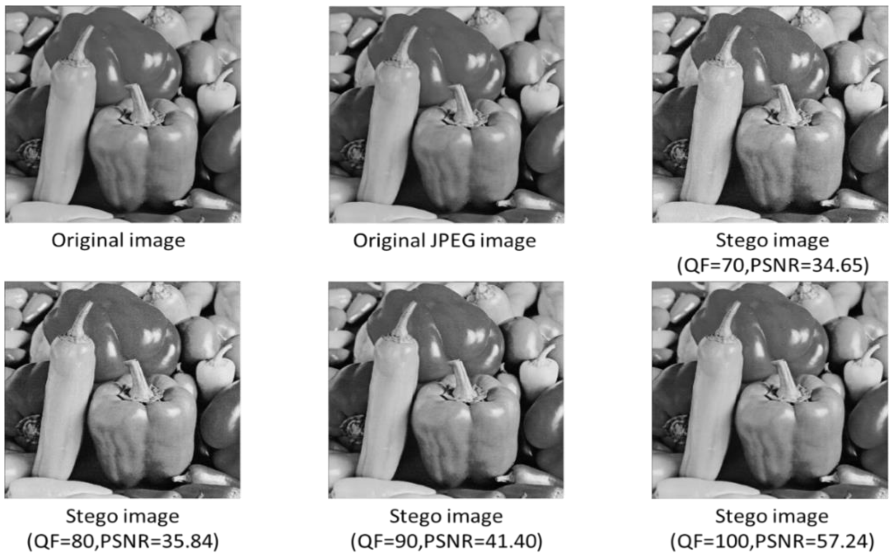

Original image
Original JPEG image
Stego image (QF=70,PSNR=34.65)
Stego image (QF=80,PSNR=35.84)
Stego image (QF=90,PSNR=41.40)
Stego image (QF=100,PSNR=57.24)

Fig. 9. Peppers test results when QF=70, 80, 90, and 100.

5. Conclusions

Recently, Huang *et al.* [18] proposed an RDH method based on HS and used a block selection strategy in JPEG images. However, this method does not fully use all the AC coefficients in the quantized DCT block. In this paper, we proposed a high capacity reversible data hiding method based on HS and utilization of the (7, 4) Hamming code in JPEG images. A few parts of the zero AC coefficients were used to embed the secret, but this was not perceived by the eavesdropper due to the small number of zeros that are changed. Our proposed RDH method for JPEG images is able to embed an average of 50,000-100,000 secret message bits. Compared to Kuo *et al.'s* [22] and Zhang *et al.'s* [20] methods, our method has better visual quality and PSNRs under the same hiding capacity. Compared to Huang *et al.'s* [18] method, our method still has very high visual quality. With the increase of QF, the PSNR of our proposed method will only continue to improve. Therefore, if a sender needs a larger capacity to hide more secret messages, a low QF will be more helpful (e.g., QF=70). If the sender needs high visual quality to avoid an eavesdropper's suspicion, then a QF=100 is a better choice.

References

[1] Barton, J. M. (1999). Method and apparatus for embedding authentication information within digital data. US, US 5912972 A.

[2] Fridrich, A. J., Goljan, M., & Du, R. (2002). Lossless data embedding for all image formats. *Proceedings of SPIE - The International Society for Optical Engineering*.

[3] Celik, M. U., Sharma, G., Tekalp, A. M., & Saber, E. (2005). Lossless generalized-lsb data embedding. *IEEE*

Transactions on Image Processing A Publication of the IEEE Signal Processing Society, 14(2), 253-266.

[4] Tian, J. (2003). Reversible data embedding using a difference expansion. *IEEE Transactions on Circuits & Systems for Video Technology*, *13(8)*, 890-896.

[5] Alattar, A. M. (2004). Reversible watermark using the difference expansion of a generalized integer transform. *IEEE Transactions on Image Processing A Publication of the IEEE Signal Processing Society*, *13(8)*, 1147-56.

[6] Thodi, D. M., & Rodriguez, J. J. (2007). Expansion embedding techniques for reversible watermarking. *IEEE Transactions on Image Processing*, *16(3)*, 721-30.

[7] Ni, Z., Shi, Y. Q., Ansari, N., & Su, W. (2004). Reversible data hiding. *International Workshop on Digital Watermarking*. Springer, Berlin, Heidelberg.

[8] Lee, S., Suh, Y., & Ho, Y. (2006). Reversiblee image authentication based on watermarking. *IEEE International Conference on Multimedia and Expo*.

[9] Hong, W., Chen, T. S., & Shiu, C. W. (2009). Reversible data hiding for high quality images using modification of prediction errors. Elsevier Science Inc.

[10] Li, X., Li, B., Yang, B., & Zeng, T. (2013). A general framework to histogram-shifting-based reversible data hiding. *IEEE Transactions on Image Processing*, *22(6)*, 2181-2191.

[11] Chang, C. C., Kieu, T. D., & Chou, Y. C. (2008). A high payload steganographic scheme based on (7, 4) hamming code for digital images. *Proceedings of the International Symposium on Electronic Commerce and Security*.

[12] Mao, K. (2016) Removable data hiding scheme in encrypted images based on (7, 4) hamming code (Report). *China: Technical Report of School of Computer Science and Technology of Anhui University*.

[13] Hamming, R. W. (1998). Coding and information theory. World Publishing Corporation.

[14] Morelos-Zaragoza, R. H. (2006). The art of error correcting coding. John Wiley & Sons.

[15] Li, X., Yang, B., & Zeng, T. (2011). Efficient reversible watermarking based on adaptive prediction-error expansion and pixel selection. *IEEE Transactions on Image Processing a Publication of the IEEE Signal Processing Society*, *20(12)*, 3524-33.

[16] Xuan, G., Shi, Y. Q., Ni, Z., Chai, P., Cui, X., & Tong, X. (2007). Reversible data hiding for JPEG images based on histogram pairs. *Proceedings of the International Conference on Image Analysis and Recognition*.

[17] Sakai, H., Kuribayashi, M., & Morii, M. (2008). Adaptive reversible data hiding for JPEG images. *Proceedings of the International Symposium on Information Theory and ITS Applications* (pp. 1-6).

[18] Huang, F., Qu, X., Kim, H. J., & Huang, J. (2016). Reversible data hiding in jpeg images. *IEEE Transactions on Circuits & Systems for Video Technology*, *26(9)*, 1610-1621.

[19] Wallace, G. K. (1992). The JPEG still picture compression standard. *Communications of the Acm*, *38(1)*.

[20] Zhang, X., Yu, C., Wang, X., Ding, F., & Tang, Z. (2013). A reversible data hiding scheme for jpeg images. *ICIC Express Letters*, *7(9)*, 2575-2580.

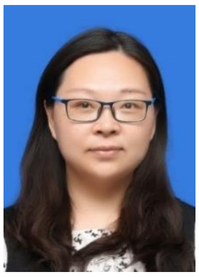
[21] Qian, Z., & Zhang, X. (2012). Lossless data hiding in jpeg bitstream. *Journal of Systems & Software*, *85(2)*, 309-313.

[22] Kuo, W. C., & Kuo, S. H. (2012). Reversible data hiding for JPEG based on EMD. *Proceedings of the* 2012 *Seventh Asia Joint Conference on Information Security*.

Chin-Cheng Chang is a fellow of IEEE and a Fellow of IEE. He is also a member of the Chinese Language Computer Society, the Chinese Institute of Engineers of the Republic of China, and the Phi Tau Phi Society of the Republic of China. His research interests include database design, computer cryptography, and data compression**.**

Chia-Chen Lin is a fellow of IET. She is currently an professor of the Department of Computer Science and Information Management, Providence University, Sha-Lu, Taiwan. She is also a member of IEEE and a member of ACM. Her research interests include image and signal processing, image hiding, mobile agent, and electronic commerce.

Ran Tang was born in Anhui province, China, in 1992. He graduated from Anhui University of Science and Technology, now in Anhui University, studying computer application technology master's degree. He current research interests include JPEG image data hiding and Sudoku data hiding.



Wan-Li Lyu was born in Anhui province, China, in 1974. She received the M.S. degree in computer science and technology with Guangxi University and the Ph.D. degree in computer science and technology with Anhui University. She was a postdoctoral research fellow in Department of Information Engineering and Computer Science at Feng Chia University from August 2013 to July 2014. Since July 2004, she is a lecturer in School of Computer Science and Technology, Anhui University. Her current research interests include image processing, computer cryptography and information security.