Multimed Tools Appl (2018) 77:8785–8803
DOI 10.1007/s11042-017-4774-y

Reversible data hiding in dual Stegano-image using an improved center folding strategy

Li-Pin Chi[1] · **Chang-Han Wu**[2] · **Hsung-Pin Chang**[2]

Received: 26 September 2016 / Revised: 24 February 2017 / Accepted: 28 April 2017 /
Published online: 8 May 2017

Abstract In recent years, the dual stego-image reversible data embedding methods have been developed rapidly, e.g., exploiting modification direction, magic matrix, least significant bit matching, and center folding strategy. The kind of method can effectively embed secret data into two stego-images and maintain excellent image quality and enhance the security. In 2015, Lu et al. proposed a center folding strategy that can effectively encode messages as the smaller digits. The encoding procedure reduces the modification level of pixels, thereby maintaining good image quality. However, their strategy does not use the relationship between the adjacent digits to reduce the number of the largest digits. Inspired by joint neighboring coding, we proposed a dynamic encoding strategy to improve the center folding strategy. The encoding strategy can reduce the secret digits and decrease the occurrence frequency of the maximum digits, thereby substantially reducing the modification level of pixels. The advantage makes that the proposed method can achieve a higher PSNR value than previous methods under the same embedding rate.

Keywords Dual stego-image reversible data embedding · Center folding strategy · Joint neighboring coding · Dynamic encoding

✉ Chang-Han Wu
phd9712@cs.nchu.edu.tw

Li-Pin Chi
cp531220@ms23.hinet.net

Hsung-Pin Chang
hpchang@cs.nchu.edu.tw

[1] Aeronautical Research Laboratory, National Chung-Shan Institute of Science and Technology, Taichung 40722, Taiwan

[2] Department of Computer Science and Engineering, National Chung Hsing University, 250, Kuo Kuang Road, Taichung 40227 Taiwan, Republic of China

1 Introduction

The reversible data embedding method can embed messages into multimedia, e.g., images [1, 3, 4, 6, 7, 9–13, 15–17, 21–23], videos [5], DNA sequences [2, 8], and compression codes [18–20]. After the data have been embedded, the multimedia is very similar to the original multimedia, thus the method is very suitable for some applications, e.g., steganography, watermarking, and annotation [14].

There are four types of reversible data embedding methods, i.e., difference expansion [23], prediction error expansion [6, 10, 13, 21, 22], histogram shifting [17], and dual stego-images [8–16. In 2003, Tian [1] expanded the difference between two adjacent pixels to embed one bit. Fig. 1 shows a diagram of the difference expansion method, where $\{P_1,\ P_2\}$ denote two cover pixels, and $\left\{P_1^{'},\ P_2^{'}\right\}$ denote two stego pixels. Obviously, difference expansion may cause serious visual distortion of the stego-image. In order to avoid this distortion, Thodi and Rodriguez [21] utilized an inherent edge-detection method to derive the prediction value of the cover pixel. Afterwards, the prediction error between the cover pixel and its prediction value was expanded to embed one bit. Fig. 2 shows the color of most prediction errors as black, indicating that most prediction errors are small. As a result, the expansion of the prediction error does not cause serious distortion of the image.

In 2006, Ni et al. [17] proposed a histogram shifting method that only modifies the cover pixel between the peak point and the zero point to embed messages and maintain satisfactory image quality. Fig. 3 shows a histogram of cover pixels, where the peak point means the cover pixel with most occurrence frequency, and the zero point means the cover pixel with lowest occurrence frequency. The cover pixels between the peak point and the zero point were shifted by one unit to create an embedding space. Afterwards, the cover pixels that were equal to the peak point are used to embed messages. The histogram shifting method effectively can control the distortion of each pixel within 1. However, this method cannot embed a large amount of data.

Different from the three kinds of methods mentioned above, Chang et al. [1] skillfully utilized the exploiting modification direction (EMD) [25] to embed base-5 digits into two identical images. The method needs more spaces to storage two stego-images. However, any third party cannot efficiently extract data from one of the stego-images, indicating that the kind of method provides greater security. The method modifies all pixels, where the modification level ranges from 0 to 2. Lee et al. [11, 12] expanded the number of directions to reduce the number of modified pixels and to control the modification level between 0 and 1, thus the method can maintain the satisfactory quality of the stego-image. In addition, Chang et al. [4] utilized the magic matrix to embed more bits into the cover-image. However, in the data

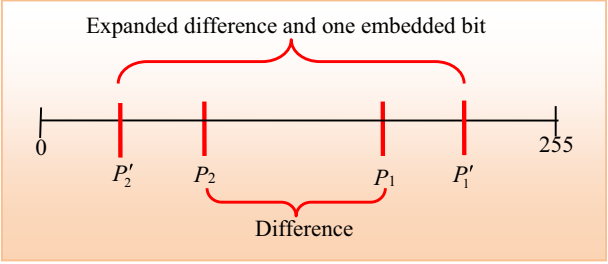


Fig. 1 Difference expansion method

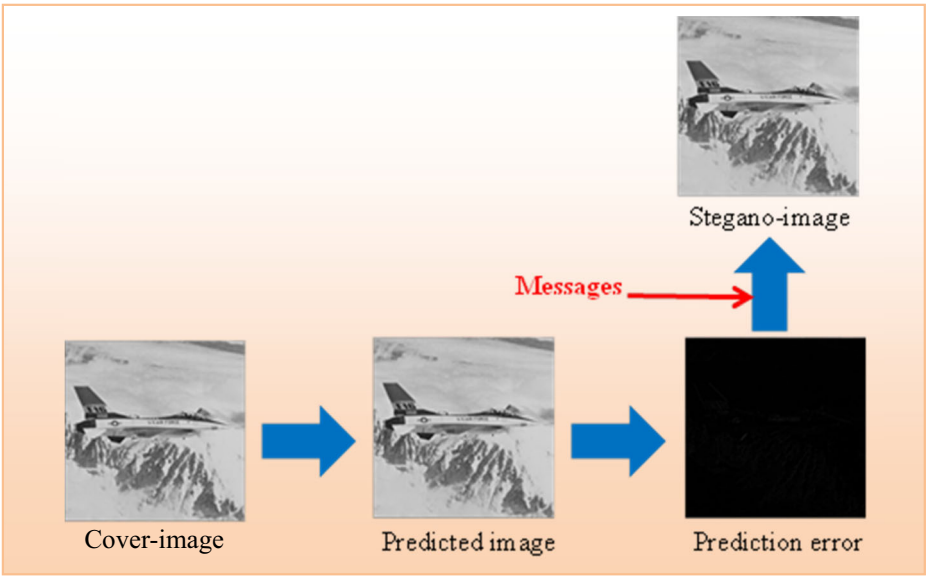
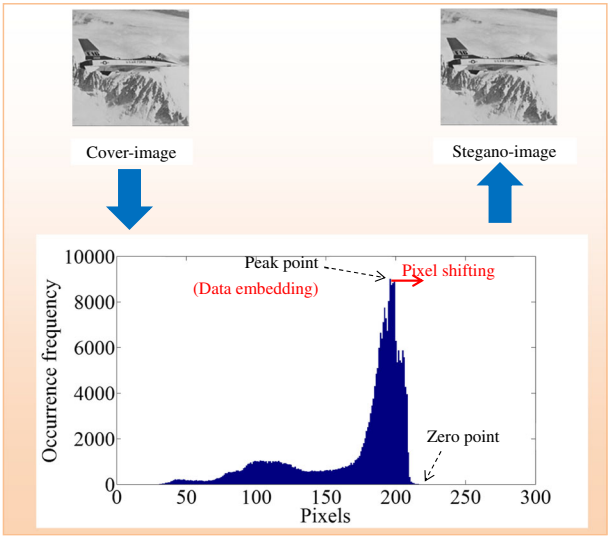


Fig. 2 Prediction error expansion method

embedding phase, they modified the pixels extensively, causing the image quality to decrease. Different from the above methods, Horng et al. [7] designed an embedding mechanism to embed K secret images and $(N - K)$ cheating images into N stego-images. However, each pixel in the stego image only used to embed one secret bit.

In 2016, Jana et al. [9] used the pixel value differencing (PVD) [24] and difference expansion [23] to embed secret data into two stego-images. First of all, they calculated the deviation between the pre-determined lower bound and the difference of two adjacent pixels to generate the reference information of the image recovery. Afterwards, two adjacent pixels were

Fig. 3 Histogram shifting method



duplicated to embed secret data and the reference information of the image recovery, where the PVD method [24] was used to embed four secret bits into the first pixel-pair and the difference expansion was used to embed the reference information of the image recovery and one secret bit.

Unlike in the direction-based hiding methods, Lu et al. [15] designed several embedding rules according to least significant bit (LSB) matching, where each pixel was used to embed only one bit. In order to embed more bits and maintain satisfactory quality of the stego-image, Lu et al. [16] proposed a center folding strategy (CFS) that adaptively encodes several bits as the smaller digits. However, CFS cannot decrease the frequency of occurrence of the largest value. In this paper, we propose a dynamic encoding strategy to reduce the frequency of occurrence of the largest value. Thus, we can effectively encode massive digits as the smaller values.

The rest of the paper is organized as follows. Sections 2 and 3 present the CFS hiding method and the proposed method, respectively. Section 4 compares the proposed method and the five related methods in terms of the embedding rate and the quality of the stego-image. The conclusions are presented in Section 5.

2 Related work

Lee et al. proposed a direction-based hiding strategy that can embed base-5 digits into two stego-images, as shown in Fig. 4. Assume that a pair of pixels $\{P_{1,1}, P_{1,2}\}$ is {5, 14} and two secret digits are {4, 2}. First of all, the pixel-pair is duplicated to embed two secret digits. According to the fourth blue arrow-shaped path in Fig. 4, the first pixel-pair is modified to embed the first secret digit "4", i.e., $P'_{1,1} = P_{1,1} + 1 = 5 + 1 = 6$ and $P'_{1,2} = P_{1,2} - 1 = 14 - 1 = 13$. Afterwards, the second pixel-pair is modified according to the second red arrow-shaped path, i.e., $P''_{1,1} = P_{1,1} = 5$ and $P''_{1,2} = P_{1,2} + 1 = 14 + 1 = 15$. The method controls the modification level of pixel within 1, thereby maintaining good image quality. However, the method cannot effectively embed greater digits.

Chang et al. [4] used a magic matrix to embed base-9 secret digits into two stego-images, thus the hiding capacity of Chang et al.'s method is higher than Lee et al.'s method. The magic matrix is generated by

$$M(P_{x,y}, P_{x,y}) = (P_{x,y} + 3 \times P_{x,y}) \bmod 9. \tag{1}$$

$M(P_{x,y}, P_{x,y})$ represents the value of the magic matrix, and $P_{x,y}$ represents the cover pixel. Fig. 5 shows the magic matrix. Assume that the cover pixel is 5 and the secret digit is 4. The cover pixel "5" is duplicated and mapped into the magic matrix, i.e., $M(5, 5) = (5 + 3 \times 5) \bmod 9 = 2$. The matrix value $M(5, 5)$ is not equal to the secret digit "4," thus the pixels {5, 5} are modified according to the coordinates of diagonal of the current matrix value. The left-top matrix value $M(4, 6)$ is matched with the secret data, therefore its coordinates {4, 6} are used as the stego pixels. Although the method can embed base-9 secret digits, the modification level of pixel is large, decreasing the image quality.

In 2015, Lu et al. [16] proposed a CFS hiding method that decreases each decimal digit by the median to reduce the modification level in the data embedding phase. Fig. 6 shows a flowchart of the CFS hiding method.

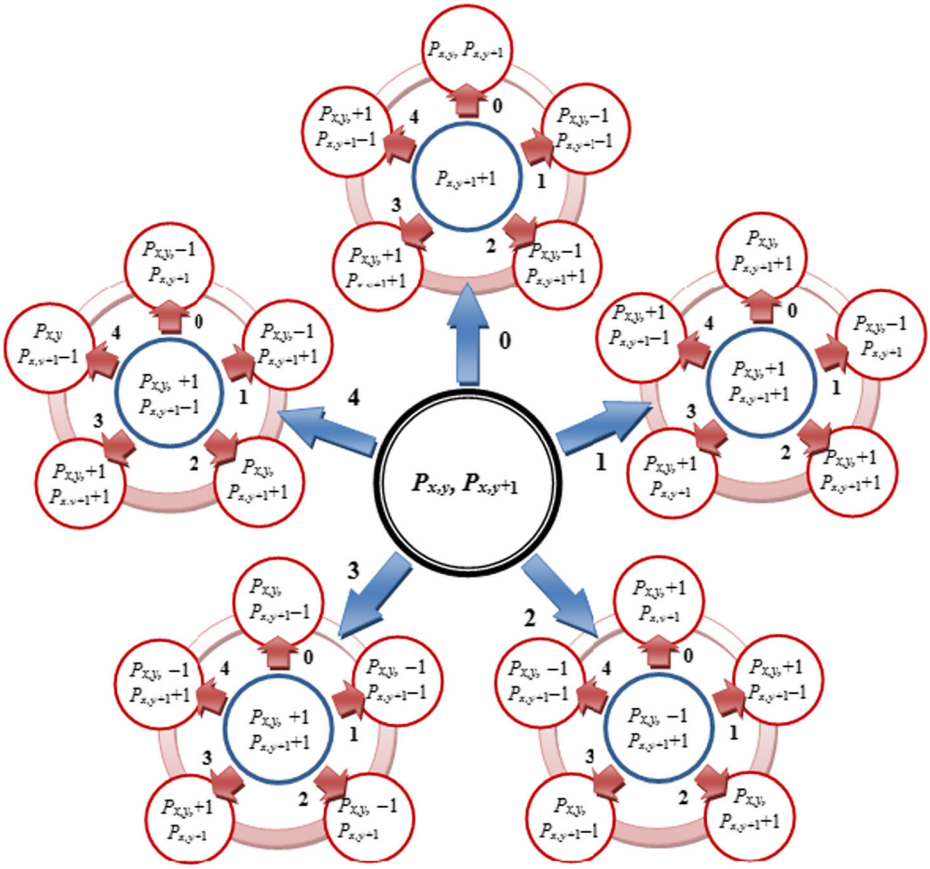


Fig. 4 Direction-based hiding method

First, the set of K embedded bits $\{s_1, s_2, \ldots, s_K\}$ is transformed into a decimal value d_i, where $d_i = \sum_{j=1}^{K} s_j \times 2^{j-1}$. The notation i denotes the ID number of the decimal value. The decimal value d_i ranges from 0 to $2^k - 1$, i.e., $0 \le d_i \le 2^k - 1$. Afterwards, the decimal value d_i is reduced by $d_i^{'} = d_i - 2^{K-1}$, making that the range of the reduced value $d_i^{'}$ is limited within $[-2^K$

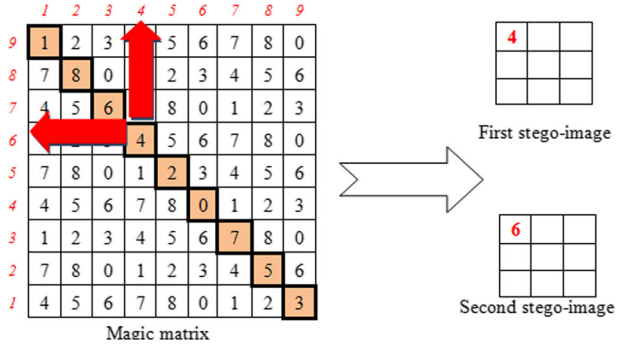


Fig. 5 Magic matrix based hiding method

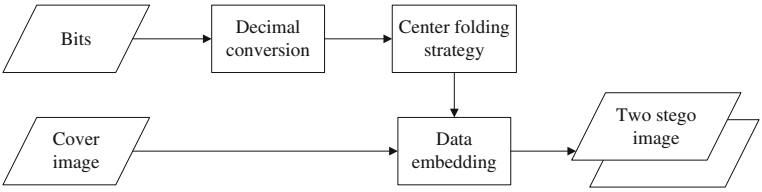


Fig. 6 Center folding strategy

$^{-1}$, $2^{K-1}-1$], i.e., $-2^{K-1} \le d_i^{'} \le 2^{K-1}-1$. Finally, the reduced digit $d_i^{'}$ is embedded into two stego-images, i.e., $P_{x,y,1} = P_{x,y} + \left\lfloor \frac{d_i^{'}}{2} \right\rfloor$ and $P_{x,y,2} = P_{x,y} - \left\lceil \frac{d_i^{'}}{2} \right\rceil$.

The above procedures are explained further by the following example. Let $K = 3$. Assume that the three embedded bits are {1, 1, 1} and that one cover pixel $P_{1,1}$ is 105. The three bits {1, 1, 1} are transformed into the decimal value d_1, i.e., $d_1 = \sum_{j=1}^{3} 2^{j-1} \times s_j = 2^0 \times 1 + 2^1 \times 1 + 2^2 \times 1 = 7$. The decimal value d_1 is decreased by $d_1^{'} = d_1 - 2^{K-1} = 7 - 2^{3-1} = 3$. Finally, the reduced value $d_1^{'} = 3$ is embedded into two stego pixels, i.e., $P_{1,1,1} = P_{1,1} + \left\lfloor \frac{d_1^{'}}{2} \right\rfloor = 105 + \left\lfloor \frac{3}{2} \right\rfloor = 106$ and $P_{1,1,2} = P_{1,1} - \left\lceil \frac{d_1^{'}}{2} \right\rceil = 105 - \left\lceil \frac{3}{2} \right\rceil = 103$.

When the legal user received two stego-images, he/she can use the following procedure to extract secret data and recover the cover image, as shown in Fig. 7. First of all, the differences between pixels in two stego-images are calculated to derive the encoded digits, i.e., $d_i^{'} = P_{x,y,1} - P_{x,y,2}$. After that, the encoded digits are decoded by $d_i = d_i^{'} + 2^{K-1}$. Finally, each decoded digits can be transformed into K secret bits.

The following example is used to further explain the extraction and recovery procedures. The pixels in two stego-images are 106 and 103, respectively, hence the difference between two pixels is 3 and it is just the encoded digit. Then, the encoded digit can be decoded correctly by $d_1 = d_1^{'} + 2^{3-1} = 3 + 4 = 7$. Finally, the decimal digit is converted into three secret bits, i.e., {1, 1, 1}. In addition, the cover pixel was recovered losslessly by $P_{x,y} = \left\lceil \frac{P_{x,y,1} + P_{x,y,2}}{2} \right\rceil = \left\lceil \frac{106+103}{2} \right\rceil = 105$.

The CFS method can reduce the embedded digits from [0, 2^{K-1}] to [-2^{K-1}, $2^{K-1}-1$]. However, it cannot decrease the frequency of occurrence of the largest value. Fig. 8 shows that the CFS method with $K = 2$ cannot effectively reduce the number of the largest value. This shortcoming decreases the visual quality of the stego-image.

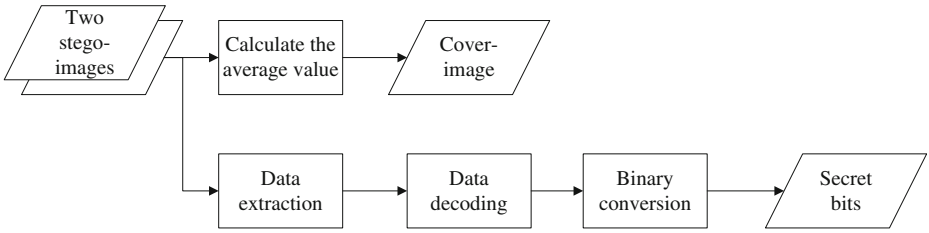
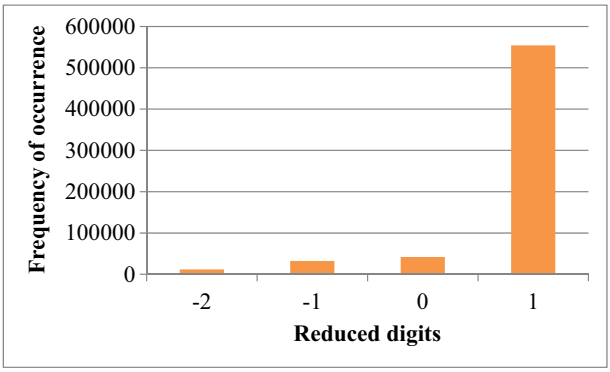


Fig. 7 Flowchart of data extraction and image recovery of the CFS hiding method

Fig. 8 Results of encoding the embedded image by the CFS method with $K = 2$

(a) Embedded image



(b) $K = 2$

3 Proposed method

In the Section, we proposed the dynamic encoding strategy to reduce the number of the largest values.

3.1 Dynamic encoding and data embedding

Fig. 9 shows the flowchart of our data embedding. A set of K bits $\{s_1, s_2, \ldots, s_K\}$ is converted into a decimal value d_i, i.e., $d_i = \sum_{j=1}^{K} s_j \times 2^{j-1}$ and $0 \le d_i \le 2^K - 1$. The notation i

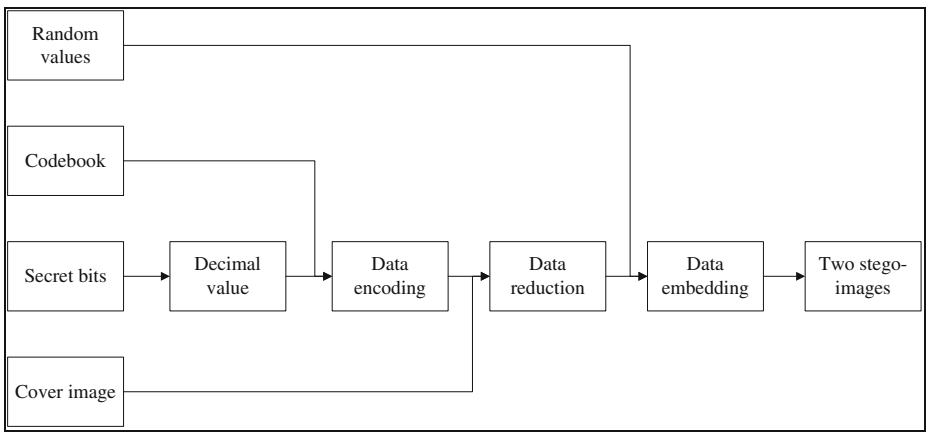


Fig. 9 Flowchart of data embedding of the proposed method

denotes the ID number of the decimal value. In order to encode the decimal value d_i, one codebook with the size of 2^K is generated, where the initial values of codewords and their indices are set sequentially from 0 to $2^K - 1$. The index of codewords can be used effectively to replace the decimal values, thereby reducing the modification level. The encoding procedure is as follows

The first decimal value is matched with the codeword from the codebook, and its index I_1 is used to represent the decimal value. Then, the index is encoded further by the formula

$$I_i^{'} = \begin{cases} \frac{I_i}{2}, & \text{if } I_i \text{ is an even number,} \\ -\frac{I_i + 1}{2}, & \text{otherwise.} \end{cases} \tag{2}$$

Figure 10 compares the original decimal digits and the encoded indices $I_i^{'}$. Moreover, the index of the codeword is updated from I to 0. The updating procedure helps match the next decimal value with the first codeword, thereby encoding the smaller digit. Note that the indices of other codewords are increased by 1 to ensure the validity of the codebook. Finally, the encoded index $I_i^{'}$ is embedded into two stego-images, i.e.

$$P_{x,y,1} = \begin{cases} P_{x,y} + \left\lfloor \frac{I_i^{'}}{2} \right\rfloor, & \text{if } r = 0, \\ P_{x,y} - \left\lceil \frac{I_i^{'}}{2} \right\rceil, & \text{if } r = 1, \end{cases} \text{and} P_{x,y,2} = \begin{cases} P_{x,y} - \left\lceil \frac{I_i^{'}}{2} \right\rceil, & \text{if } r = 0, \\ P_{x,y} + \left\lfloor \frac{I_i^{'}}{2} \right\rfloor, & \text{if } r = 1, \end{cases} \tag{3}$$

where r is a random value, which is generated by the pre-determined random seed [7, 9, 11].

The above procedures can be illustrated further by the following algorithm.

Step 1: Convert a set of K secret bits into a decimal value, i.e., $d_i = \sum_{j=1}^{K} s_j \times 2^{j-1}$.

Step 2: Generate one codebook, where it consists of 2^K codewords.

Step 3: Select an index I_i from the codebook, where its codeword is equal to the decimal value.

Step 4: Reduce the index by Eq. (2).

Step 5: Classify the pixel $P_{x,y}$ into the embeddable pixels and non-embeddable pixels. The pixels between 2^{K-1} and $256 - 2^{K-1}$ are labeled as the embeddable pixels. Otherwise, other pixels are the non-embeddable pixels.

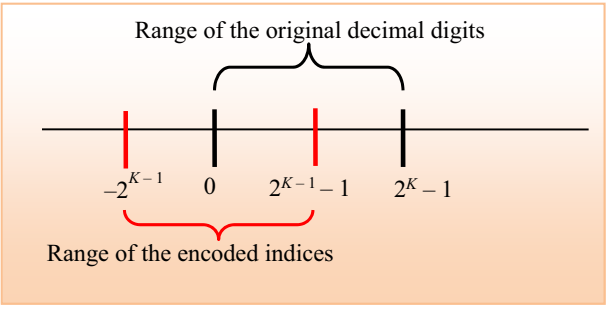


Fig. 10 Comparison of the decimal digits and the encoded indices

Step 6: According to the pre-determined random value, embed the reduced index I_i' into two stego pixels $\{P_{x,y,1}, P_{x,y,2}\}$ by Eq. (3).

Step 7: Update the index of the codeword, i.e., modifying the index of the selected codeword from I_i to 0 and increasing the index of the other codewords by 1.

Step 8: Perform the above steps for the next set of secret bits until all secret bits have been embedded.

Figure 11 illustrates the above procedures. Assume that the 12 embedded bits are {1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1} and the four cover pixels are {105, 105, 8, 30}. Let $K = 3$ and $r = \{0, 0\}$. As a result, one codebook with the size of 2^3 is established, where the initial values of the codewords and their indices are set sequentially from 0 to 7. The first trio of bits {1, 1, 1} are transformed into a decimal digit, i.e., $d_1 = 1 \times 2^0 + 1 \times 2^1 + 1 \times 2^2 = 7$. The procedure for transforming other bits is the same as the above procedure

These decimal digits are reduced by the following procedure. First, the first decimal digit $d_1 = 7$ matches the last codeword from the codebook, thus the first decimal digit is replaced by the index of the last codeword, i.e., $I = 7$. Then, the index is reduced by Eq. (2), i.e., $I' = -(7 + 1)/2 = -4$. In addition, the index of the codeword is updated from 7 to 0, thereby enhancing the possibility of priority-based matching.

The second decimal digit $d_2 = 7$ matches the first codeword in the updated codebook, thus its index "0" is used to represent the second decimal digit. The codebook remains unchanged because the index of the codeword "7" is 0. The procedure for encoding other indices is the same as the above procedure.

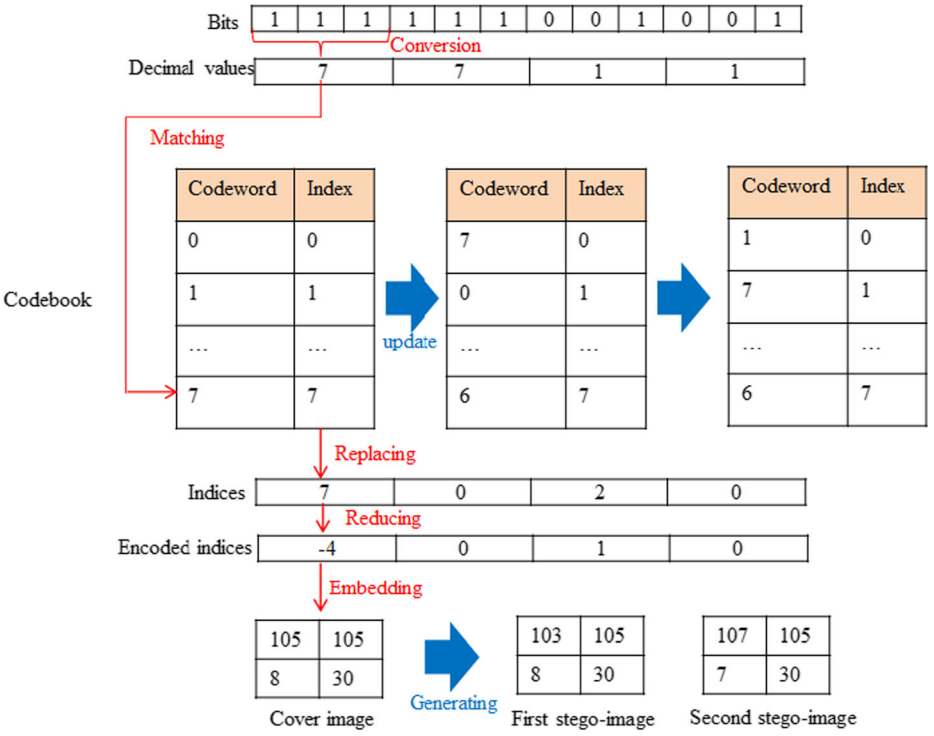


Fig. 11 Example of dynamic encoding and embedding

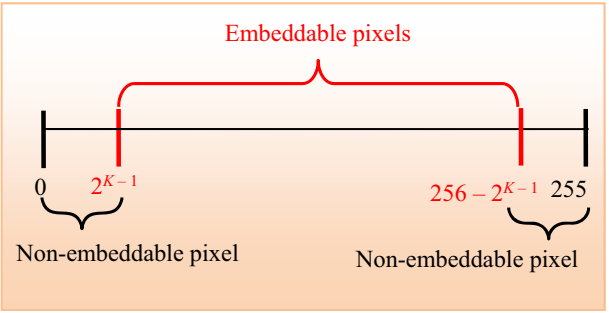


Fig. 12 Solution for the overflow and underflow problems by pixel classification

After the above procedure, these encoded indices {−4, 0, 2, 0} are embedded into the cover-image to obtain two stego-images. The first encoded index $I_1^{'} = -4$ is embedded into the first cover pixel $P_{1,1} = 105$ to obtain two stego-pixels, i.e., $P_{1,1,1} = 105 + \left\lfloor \frac{-4}{2} \right\rfloor = 103$ and $P_{1,1,2} = 105 - \left\lceil \frac{-4}{2} \right\rceil = 107$. Then, the second encoded index $I_2 = 0$ is embedded into the second pixel $P_{1,2} = 105$ to obtain two stego-pixels, i.e., $P_{1,1,1} = 105 + \left\lfloor \frac{0}{2} \right\rfloor = 105$ and $P_{1,1,2} = 105 - \left\lceil \frac{0}{2} \right\rceil = 105$. Fig. 11 shows the results of dynamic encoding and embedding.

The proposed method can embed a larger amount of secret data and maintain good image quality. However, a few overflow and underflow problems occur in the data embedding phase. In order to avoid these problems, the proposed method classifies the cover pixels into the embeddable pixels and non-embeddable pixels, as shown in Fig 12. The cover pixels between 2^{K-1} and $256 - 2^{K-1}$ are classified into the embeddable pixels because the modification level of pixel ranges from -2^{K-1} to $2^{K-1} - 1$. The non-embeddable pixels remain unchanged to effectively avoid the overflow and underflow problems. In addition, the strategy does not need to any extra data to discriminate between the embeddable pixels and non-embeddable pixels.

3.2 Dynamic decoding and image recovery

Fig. 13 shows the flowchart of our extraction and recovery procedure. When the legal user received two stego-images, he/she can correctly decode the embedded bits and

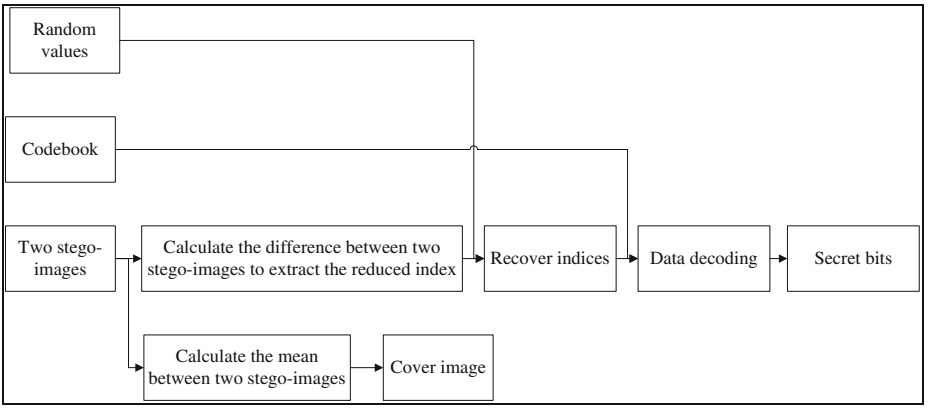


Fig. 13 Flowchart of data extraction and image recovery

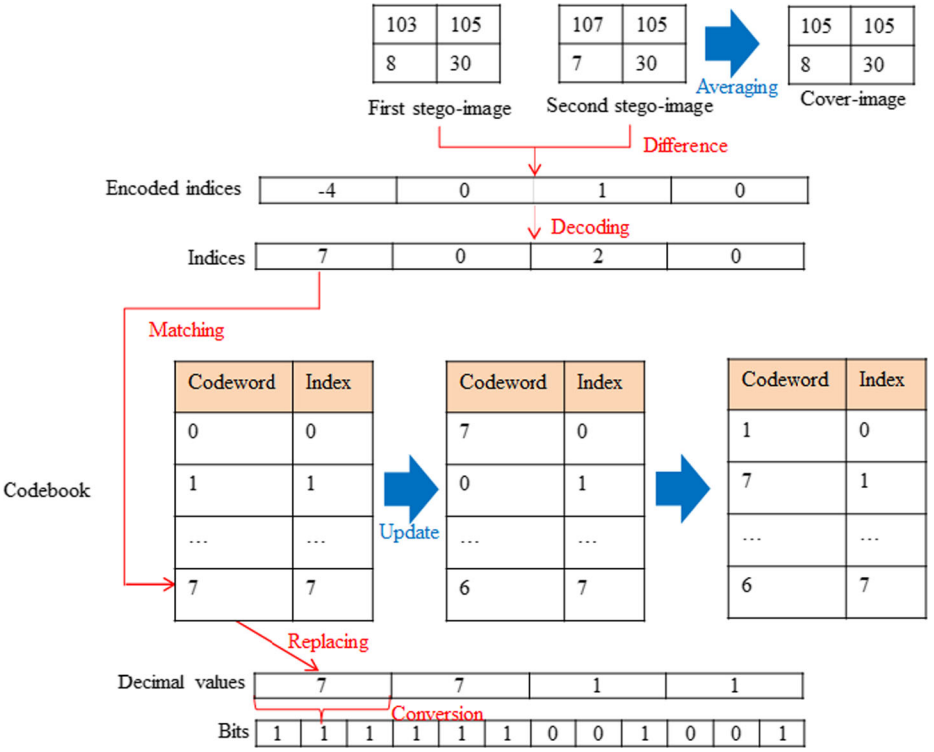


Fig. 14 Example of dynamic decoding and recovery

losslessly recover the original image. First of all, a pair-pixel in two stego images $\{P_{x,y,1},\ P_{x,y,2}\}$ is classified into the embedded pixel or the non-embeddable pixel. If two pixels are greater than 2K – 1 or smaller than 256 – 2K – 1, then it is classified into the non-embeddable pixel. There is no secret bit in the non-embeddable pixel. In addition, the pixel is just the cover pixel. Other pixels between 2K – 1 and 256 – 2K – 1 are used to extract secret data and recover the cover image by the following procedures. The codebook with the size of 2^K is generated in the same way as the dynamic encoding strategy. According to the pre-determined random value, the difference between pixels in two stego-images is calculated to obtain the embedded index, i.e.

$$I_i^{'} = \begin{cases} P_{x,y,1} - P_{x,y,2}, & \text{if } r = 0, \\ P_{x,y,2} - P_{x,y,1}, & \text{if } r = 1. \end{cases} \tag{4}$$

After obtaining the embedded value $I_i^{'}$, the original index of the dynamic codeword can be derived by

$$I_i = \begin{cases} 2I_i^{'}, & \text{if } I_i^{'} \geq 0, \\ 2 \times \left|I_i^{'}\right| - 1, & \text{otherwise.} \end{cases} \tag{5}$$

The codeword of the index I_i is just the decimal representation of the set of K bits. Note that the index of the codeword needs to update from I_i to 0 to ensure the validity of the codebook, and the indices of other codewords need to increase by 1. The above procedures are repeated until all messages have been decoded. Moreover, the mean of two stego-images is just the cover-image, i.e., $P_{x,y} = \left\lceil \frac{P_{x,y,1}+P_{x,y,2}}{2} \right\rceil$. The above procedures can be illustrated further by the following algorithm.

Step 1: Generate one codebook.
Step 2: Classify a pixel-pair $\{P_{x,y,1}, P_{x,y,2}\}$ in two stego images into the embedded pixel-pair and the non-embedded pixel-pair. If both stego pixels belong to $[2^{K-1}, 256 - 2^{K-1}]$, there is a secret digit between two pixels. The secret digit can be extracted correctly by the following steps. Otherwise, other pixels are label as the non-embedded pixels.
Step 3: According to the pre-determined random value and Eq. (4), the difference between pixels in two stego-images is calculated to obtain the embedded index.
Step 4: Recover the original index by Eq. (5).
Step 5: According to the original index, extract the codeword from the codebook.
Step 6: Transform the codeword into K secret bits.
Step 7: Recover the pixel by $P_{x,y} = \left\lceil \frac{P_{x,y,1}+P_{x,y,2}}{2} \right\rceil$.


Lena


Baboon


F-16


Tiffany


Sailboat


Barbara


Boats


Girl


Goldhill


Zelda

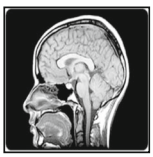
(a) Ten cover-images, each size of which is 512 × 512


Logo with the size of 350×350


Dolphin with the size of 375 × 352


Google with the size of 600× 177


IM with the size of 430 × 224


Brain with the size of 269 × 360

(b) Five embedded images

Fig. 15 Test images

Table 1 Results of embedding the images "Logo" and "Dolphin" into different cover-images

Cover-image	K	Logo			Dolphin		
		$\mathfrak{R}$ (bpp)	$PSNR_1$ (dB)	$PSNR_2$ (dB)	$\mathfrak{R}$ (bpp)	$PSNR_1$ (dB)	$PSNR_2$ (dB)
Lena	2	1	60.65	58.84	1	53.36	53.22
	3	1.5	53.36	51.48	1.5	48.90	49.41
	4	2	49.52	50.20	2	44.61	45.03
Baboon	2	1	60.65	58.84	1	53.36	53.22
	3	1.5	53.36	51.48	1.5	48.90	49.41
	4	2	49.52	50.20	2	44.61	45.03
F-16	2	1	60.65	58.84	1	53.36	53.22
	3	1.5	53.36	51.48	1.5	48.90	49.41
	4	2	49.52	50.20	2	44.61	45.03
Tiffany	2	1	60.65	58.84	1	53.36	53.22
	3	1.5	53.36	51.48	1.5	48.90	49.41
	4	2	49.52	50.20	2	44.61	45.03
Sailboat	2	1	60.65	58.84	1	53.36	53.22
	3	1.5	53.36	51.48	1.5	48.90	49.41
	4	2	49.52	50.20	2	44.61	45.03
Barbara	2	1	60.65	58.84	1	53.36	53.22
	3	1.5	53.36	51.48	1.5	48.90	49.41
	4	2	49.52	50.20	2	44.61	45.03
Boats	2	1	60.65	58.84	1	53.36	53.22
	3	1.5	53.36	51.48	1.5	48.90	49.41
	4	2	49.52	50.20	2	44.61	45.03
Girl	2	1	60.65	58.84	1	53.36	53.22
	3	1.5	53.36	51.48	1.5	48.90	49.41
	4	2	49.52	50.20	2	44.61	45.03
Goldhill	2	1	60.65	58.84	1	53.36	53.22
	3	1.5	53.36	51.48	1.5	48.90	49.41
	4	2	49.52	50.20	2	44.61	45.03
Zelda	2	1	60.65	58.84	1	53.36	53.22
	3	1.5	53.36	51.48	1.5	48.90	49.41
	4	2	49.52	50.20	2	44.61	45.03

Step 8: Update the index of the codeword, i.e., modifying the index of the selected codeword from I_i to 0 and increasing the index of the other codewords by 1.

Step 9: Perform the above steps for the next set of pixels until all secret bits have been extracted.

Figure 14 illustrates the above procedure. Let $K = 3$. The codebook with the size of 8 is generated to decode the embedded bits, where the initial values of codewords and their indices are set sequentially from 0 to 7. The decoding and recovery procedures are as follows. First of all, the difference between the two stego-images is calculated to extract the encoded indices {−4, 0, 1, 0}. All encoded indices can be decoded correctly by Eq. (5) to obtain the original indices. For example, the first encoded digit "–4" is decoded by $I_1 = 2 \times |-4| - 1 = 7$. Its codeword is just the decimal representation of the three original bits, i.e., $7_{10} = (111)_2$. Afterwards, the index of the codeword is updated from 7 to 0. In addition, the other indices are increased by 1 to assure the validity of the codebook. The pixel is recovered without any distortion, i.e., $P_{1,1} = \left\lceil \frac{103+107}{2} \right\rceil = 105$

Table 2 Results of embedding the images "Google" and "IM" into different cover-images

Cover-image	K	Google			IM		
		$\mathfrak{R}$ (bpp)	$PSNR_1$ (dB)	$PSNR_2$ (dB)	$\mathfrak{R}$ (bpp)	$PSNR_1$ (dB)	$PSNR_2$ (dB)
Lena	2	1	58.99	58.65	1	56.12	56.34
	3	1.5	53.48	54.58	1.5	49.48	51.23
	4	2	52.05	52.71	2	49.20	50.04
Baboon	2	1	58.99	58.65	1	56.12	56.34
	3	1.5	53.48	54.58	1.5	49.48	51.23
	4	2	52.05	52.71	2	49.20	50.04
F-16	2	1	58.99	58.65	1	56.12	56.34
	3	1.5	53.48	54.58	1.5	49.48	51.23
	4	2	52.05	52.71	2	49.20	50.04
Tiffany	2	1	58.99	58.65	1	56.12	56.34
	3	1.5	53.48	54.58	1.5	49.48	51.23
	4	2	52.05	52.71	2	49.20	50.04
Sailboat	2	1	58.99	58.65	1	56.12	56.34
	3	1.5	53.48	54.58	1.5	49.48	51.23
	4	2	52.05	52.71	2	49.20	50.04
Barbara	2	1	58.99	58.65	1	56.12	56.34
	3	1.5	53.48	54.58	1.5	49.48	51.23
	4	2	52.05	52.71	2	49.20	50.04
Boats	2	1	58.99	58.65	1	56.12	56.34
	3	1.5	53.48	54.58	1.5	49.48	51.23
	4	2	52.05	52.71	2	49.20	50.04
Girl	2	1	58.99	58.65	1	56.12	56.34
	3	1.5	53.48	54.58	1.5	49.48	51.23
	4	2	52.05	52.71	2	49.20	50.04
Goldhill	2	1	58.99	58.65	1	56.12	56.34
	3	1.5	53.48	54.58	1.5	49.48	51.23
	4	2	52.05	52.71	2	49.20	50.04
Zelda	2	1	58.99	58.65	1	56.12	56.34
	3	1.5	53.48	54.58	1.5	49.48	51.23
	4	2	52.05	52.71	2	49.20	50.04

The second encoded index is 0, and its original index is decoded by $I_2 = 2 \times 0 = 0$. According to the index $I_2 = 0$, its codeword "7" in the updated codebook is transformed into the three original bits, i.e., $7_{10} = (111)_2$. Since the index of the codeword is 0, we do not update the codebook. Finally, the second cover pixel is recovered losslessly by $P_{1,2} = \left\lceil \frac{105+105}{2} \right\rceil = 105$. The procedures for decoding other digits and recovering other pixels are the same as the above procedures.

4 Experimental results

In this section, we compare the proposed method and four related methods in terms of the embedding rate and the visual quality of the stego-image, thereby confirming the effectiveness of the proposed method. The embedding rate $\mathfrak{R}$ was computed by

$$\mathfrak{R} = \frac{\|S\|}{2 \times \left\|P_{x,y}\right\|} \text{ (bpp)}, \tag{6}$$

where bpp is bit per pixel; $\|S\|$ is the number of embedded bits, and $\|P_{x,y}\|$ is the number of original pixels.

Table 3 Results of embedding the image "Brain" into different cover-images

Cover-image	K	$\mathfrak{R}$ (bpp)	$PSNR_1$ (dB)	$PSNR_2$ (dB)
Lena	2	1	54.08	54.29
	3	1.5	48.79	49.08
	4	2	44.56	44.85
Baboon	2	1	54.08	54.29
	3	1.5	48.79	49.08
	4	2	44.56	44.85
F-16	2	1	54.08	54.29
	3	1.5	48.79	49.08
	4	2	44.56	44.85
Tiffany	2	1	54.08	54.29
	3	1.5	48.79	49.08
	4	2	44.56	44.85
Sailboat	2	1	54.08	54.29
	3	1.5	48.79	49.08
	4	2	44.56	44.85
Barbara	2	1	54.08	54.29
	3	1.5	48.79	49.08
	4	2	44.56	44.85
Boats	2	1	54.08	54.29
	3	1.5	48.79	49.08
	4	2	44.56	44.85
Girl	2	1	54.08	54.29
	3	1.5	48.79	49.08
	4	2	44.56	44.85
Goldhill	2	1	54.08	54.29
	3	1.5	48.79	49.08
	4	2	44.56	44.85
Zelda	2	1	54.08	54.29
	3	1.5	48.79	49.08
	4	2	44.56	44.85

The visual similarity between the stego-image and the cover-image was measured by the peak signal-to-noise ratio (PSNR)

$$\mathrm{PSNR} = 10 \times \log_{10} \frac{255^2}{\mathrm{MSE}}\ (\mathrm{dB}), \text{where}\, \mathrm{MSE} = \frac{1}{\left\|P_{x,y}\right\|} \sum_{x=1}^{H} \sum_{y=1}^{W} \left(P_{x,y} - P_{x,y,z}\right)^2, \qquad (7)$$

MSE is mean square error between the cover-image and the stego-image.

Figure 15 shows ten cover-images and five embedded images. Tables 1, 2 and 3 show the results of embedding different images into each cover-image. The embedding rate becomes greater as K increases. When the embedding rate is 2 bpp, the PSNR value of each stego-image can achieve 44 dB at least. This is because the proposed method can reduce the frequency of occurrence of the largest value and encode them as the smaller digits. In addition, the proposed method is very suitable for embedding the smooth image. For example, the PSNR values of two stego-images can maintain 52 dB after embedding the image "Google".

Figure 16 shows the comparison among the proposed method and the related methods [4, 7, 9, 11, 12, 15, 16]. Both our method and the CFS method decreases the embedded digit by 2^{K-1}, thus embedding these reduced digits into the image only needs to slightly modify pixels. The

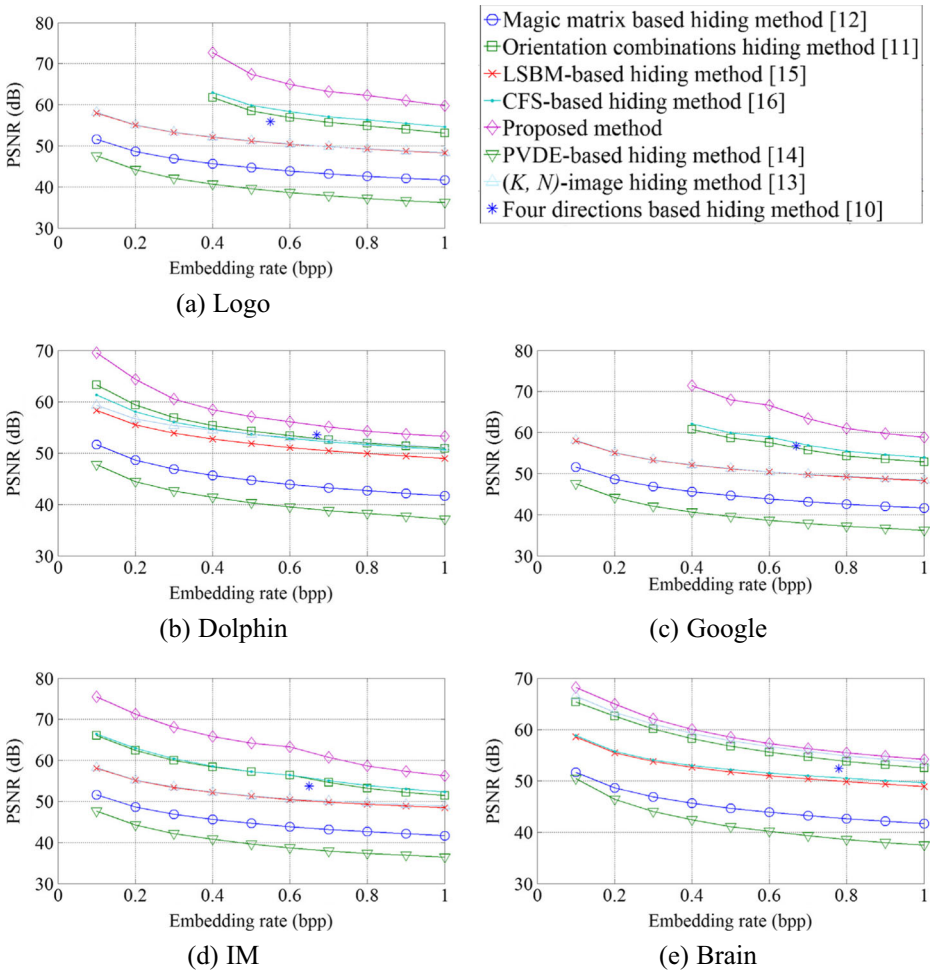


Fig. 16 Comparison among the proposed method and the seven related methods

advantage implies that our method and the CFS method have better image quality than other methods. However, the CFS method cannot decrease the occurrence frequency of the maximum digits, that is why the PSNR values of Dolphin and Brain of the CFS-based hiding method are smaller than that of Lee and Huang's method. Our method can decrease the embedded digits and the occurrence frequency of the maximum digits, thus the proposed method has better image quality than previous methods

5 Conclusions

This paper proposed a dynamic encoding strategy to reduce the frequency of occurrence of the largest value, thereby decreasing the modification level in the data embedding stage. Experimental results showed that the PSNR value of the proposed method is greater than that of the

CFS hiding method. In the future, we will analyze the correlation coefficient among the embedded bits in the complex images to further enhance the encoding performance.

References

1. Chang CC, Kieu TD, Chou YC (2007) Reversible data hiding scheme using two steganographic images. In Proceedings of IEEE Region 10 International Conference (TENCON), pp. 1–4
2. Chang CC, Lu TC, Chang YF, Lee RCT (2007) Reversible data hiding schemes for deoxyribonucleic acid (DNA) medium. International Journal of Innovative Computing, Information and Control 3(5):1145–1160
3. Chang CC, Chou YC, Kieu TD (2009) Information hiding in dual images with reversibility. In Proceedings of the Third International Conference on Multimedia and Ubiquitous Engineering, pp. 145–152
4. Chang CC, Lu TC, Horng G, Huang YH, Hsu YM (2013) A high payload data embedding scheme using dual stego-images with reversibility. In Proceedings of Third International Conference on Information, Communications and Signal Processing, pp. 1–5
5. Chung KL, Huang YH, Chang PC, Mark Liao HY (2010) Reversible data hiding-based approach for intra-frame error concealment in H.264/AVC. IEEE Trans Circuits Syst Video Technol 20(11):1643–1647
6. Fallahpour M (2008) Reversible image data hiding based on gradient adjusted prediction. IEICE Electronics Express 5(20):870–876
7. Horng G, Huang YH, Chang CC, Liu Y (2014) (k, n)-image reversible data hiding. Journal of Information Hiding and Multimedia Signal Processing 5(2):152–164
8. Huang YH, Chang CC, Wu CY (2014) A DNA-based data hiding technique with low modification rates. Multimedia Tools and Applications 70(3):1439–1451
9. Jana B, Giri D, Mondal SK (2016) Dual-image based reversible data hiding scheme using pixel value difference expansion. International Journal of Network Security 18(4):633–642
10. Lee CF, Chen HL (2012) Adjustable prediction-based reversible data hiding. Digital Signal Process 22(6): 941–953
11. Lee CF, Huang YL (2013) Reversible data hiding scheme based on dual stegano-images using orientation combinations. Telecommun Syst 52(4):2237–2247
12. Lee CF, Wang KH, Chang CC, Huang YL (2009) A reversible data hiding scheme based on dual steganographic images. In Proceedings of the Third International Conference on Ubiquitous Information Management and Communication, pp. 228–237, 2009
13. Lee CF, Chen HL, Tso HK (2010) Embedding capacity raising in reversible data hiding based on prediction of difference expansion. J Syst Softw 83(10):1864–1872
14. Lu TC, Lu CM, Chang CC (2007) Multimedia security techniques, Taiwan: CHWA
15. Lu TC, Tseng CY, Wu JH (2015) Dual imaging-based reversible hiding technique using LSB matching. Signal Process 108:77–89
16. Lu TC, Wu JH, Huang CC (2015) Dual-image-based reversible data hiding method using center folding strategy. Signal Process 15:195–213
17. Ni Z, Shi YQ, Ansari N, Su W (2006) Reversible data hiding. IEEE Trans Circuits Syst Video Technol 16(3):354–362
18. Qin C, Chang CC, Chen YC (2013) Efficient reversible data hiding for VQ-compressed images based on index mapping mechanism. Signal Process 93(9):2687–2695
19. Qin C, Chang CC, Chiu YP (2014) A novel joint data-hiding and compression scheme based on SMVQ and image inpainting. IEEE Trans Image Process 23(3):969–978
20. Qin C, Chang CC, Horng G, Huang YH, Chen YC (2015) Reversible data embedding for vector quantization compressed images using search-order coding and index parity matching. Security and Communication Networks 8(6):899–906
21. Thodi DM, Rodriguez JJ (2004) Prediction-error based reversible watermarking. In Proceedings of International Conference on Image Processing 3, 1549–1552
22. Thodi DM, Rodriguez JJ (2007) Expansion embedding techniques for reversible watermarking. IEEE Trans Image Process 16(3):721–730
23. Tian J (2003) Reversible data embedding using a difference expansion. IEEE Trans Circuits Syst Video Technol 13(8):890–896
24. Wu DC, Tsai WH (2003) A steganographic method for images by pixel-value differencing. Pattern Recogn Lett 24(9–10):1613–1626
25. Zhang X, Wang S (2006) Efficient steganographic embedding by exploiting modification direction. IEEE Commun Lett 10(11):781–783

Li-Pin Chi was born in Taichung, Taiwan, R.O.C., in 1964. He received the B.S. degree from the Chung Cheng Institute of Technology, Taoyuang, Taiwan, in 1991. He received the M.S. degree from the Department of Electrical Engineering of Da-Yeh University, Taiwan, in 1999, and awarded the Ph.D. degree from the Department of Electrical Engineering of the Feng-Chia University in 2010, Taichung, Taiwan. His current researches are in antenna theory and design, electromagnetic wave propagation.

Chang-Han Wu received the B.S. degree in Electrical and Electronic Engineering from National Defense University, and the M.S. degree in Degree Program of ECE and CS Colleges from National Chiao Tung University. He is currently pursuing the Ph.D. degree in Computer Science and Engineering from National Chung Hsing University.

Hsung-Pin Chang received the B.S., M.S., and Ph. D degrees in Computer and Information Science from National Chiao Tung University, Taiwan. He was a postdoctoral researcher in National Chiao Tung University in 2002–2003, and an assistant professor in Electrical Engineering at National Changhua University of Education, Chaunghua, Taiwan, from 2003 to 2004. Since 2004, he has been with the Department of Computer Science and Engineering at National Chung Hsing University, Taiwan, where he is now an associate professor. His research interests include wireless sensor networks, network protocols, operating systems, and embedded systems.