Multimed Tools Appl
DOI 10.1007/s11042-016-3591-z

An efficient reversible data hiding with reduplicated exploiting modification direction using image interpolation and edge detection

Chin-Feng Lee[1] · Chi-Yao Weng[2] · Kai-Chin Chen[1]

Received: 20 October 2015 / Revised: 22 March 2016 / Accepted: 6 May 2016

Abstract Data hiding is a technique that embeds a cluster of secret messages into the original image. The image with secret messages can be distributed on the Internet while the message embedded would not be easily discovered by a third party. In this way, the secret message can be well protected. At the same time, an image might be made with complex and smooth textures. If changes are made on complex textures, it is less easy for human eyes to discern the differences in the image; but if changes are made on smooth textures, the changes are easier to be discerned by human eyes. This paper proposes a data hiding method based on reduplicated exploiting modification direction (REMD), image interpolation, and canny edge detection. It aims at fulfilling two goals. First, conduct difference-embedding on the image's feature information, distribute the image, and use image interpolation to accomplish reversibility. Second, check the pixels that are located at the edge and insert different data payload according to the application demands. This allows users to flexibly adjust data payload embedded into the image's edge pixels according to their practical conditions, effectively considering both the image quality and payload. The experimental results demonstrate that the proposed method can achieve the data payload of 3.01bpp, so this is a reversible hiding technique with a very high embedding capacity. In the mean time, the average image quality is kept at an acceptable level, about 33 ± 1 dB.

Keywords Reversible data hiding · Exploiting modification direction · Image interpolation · Edge detection

✉ Chi-Yao Weng
cyweng@mail.nptu.edu.tw

[1] Department of Information Management, Chaoyang University of Technology, Taichung 41349 Taiwan, Republic of China

[2] Department of Computer Science, National Pingtung University, Pingtung 90003 Taiwan, Republic of China

1 Introduction

Nowadays, many people are taking advantage of internet, especially its temporal instantaneity, to share everything about their daily life. For example, they share information through Plurk, Facebook, Google+, or smart phones. Meanwhile, this advanced technology is utilized for illegal purposes. Among the cases of internet crimes commonly seen today is that third parties take extreme approach and perform some criminal acts. Because of these experiences, it becomes a crucial topic in keeping the confidentiality of digital image, protecting it from being intercepted or destroyed by illegal third parties, and confirming the privacy and intactness of the image.

Data hiding means that the secret message is transformed into senseless signs or random codes before it is delivered to the receiver. When the message is embedded and delivered, it is vulnerable to the awareness of the third party. At some cases, not being able to extract out the secret message, the third party conducts interception and destruction, making it impossible for the receiver to extract out the secret message.

Currently, there are two main directions in researching data hiding technique, reversible data hiding [4, 10, 11, 14–16, 20] and irreversible data hiding [1, 2, 6, 7, 12, 19]. Irreversible data hiding is often used in embedding data with high payload [3, 9, 17, 18, 21]. Its disadvantage is that the image quality would be destroyed more or less and the image cannot be recovered. A representative example is the exploiting modification direction Method (EMD), proposed by Zhang et al. in 2006 [22]. The main idea of EMD embedding is that for each group of n pixels, only one pixel is increased or decreased by 1 at most, and then secret digit in a $(2n+1)$-ary notational system can be embedded. This method ensures a high data payload (about 1.16bpp,) and keeps good image quality (about 52 dB). Other irreversible data hiding techniques include REMD proposed by Lee et al. in 2008 [10]. This method can repeat embedding more secret data on the same group of pixels. It only needs to slightly change neighboring pixels to achieve a high data payload (about 1.87bpp) and maintains good image quality (about 45 dB). Reversible data hiding is often used in special fields (such as military and medical images). Its payload is far less than that of irreversible data hiding because it needs to include extra information to help extract out the secret message and restore the original image. Classical examples of reversible data hiding include two steganographic images proposed by Chang et al. in 2007 [4]. The scheme employs the concept of EMD module function operation on two steganographic images to accomplish reversibility. Qin et al. in 2015 [15] also proposed two steganography images based on EMD. In their scheme, the first cover image is used to generate two similar images. The first similar image is applied to be hidden secret data using the EMD method. The second similar image is adaptively modified through referring the image recovery process. Similarly, in 2010, Kuo et al. proposed a technique of image interpolation based on EMD, which can accomplish reversibility [8]. For achieving a good payload performance, Zhang [20] found the optimal value transfer matrix. The matrix will be used to accomplish reversible data hiding which has good payload performance.

Moreover, as imaging technology advances rapidly, image digitalization has been widely applied to different fields [19], such as image forgery, watermarking, and image interpolation. In particular, image interpolation features a simple calculation, so it is most applied to picture resolution, scanner enlargement resolution, zoom function of digital camera, medical image, or displaying technique of plane monitor. Classical techniques of image interpolation include nearest neighbor interpolation (NNI) [6], which is the simplest method of image interpolation.

The main idea of NNI is among the 4 neighboring pixels, take one closest to the integer position of the new pixel to replace the new pixel. Its weakness is a bad performance in enlargement and serious blocky effect. Bilinear interpolation (BLI) [6] is also a commonly used technique of image processing. BLI uses 4 neighboring original pixels to obtain new pixels. It outperforms NNI in the final outcome. However, BLI has the problem of blurry effect when the image is enlarged. In 2009, Jung and Yoo [7] proposed neighbor mean interpolation(NMI), and in 2012, Lee and Huang [11] proposed interpolation by neighbor pixels (INP). These methods employ the feature that in most images, neighboring pixels share similarity. The image after interpolation still keeps the quality of the original image. Additionally, Qin et al. [14] proposed side match vector quantization (SMVQ) combined with image inpainting, and the methods accomplish reversibility.

Concluding the above methods, this paper develops a reversible data hiding technique that employs reduplicated exploiting modification direction (REMD) on image interpolation and canny edge detection. In this paper, feature information in an image, that is, differences of cover pixels and stego-pixels, is embedded before it is distributed. The image is delivered when no human eyes are able to discern any differences in the image, and image interpolation techniques will be utilized to accomplish reversibility. After the process, the receiver can take advantage of the image's feature information via difference-embedding to restore the image to its original state. In order to keep both high data payload and image quality, this paper adopts the edge detection technique to detect the location of edge pixels in the image. If it is an edge pixel, it will be inserted with more data; if it is not an edge pixel, it will be embedded with less data. In addition, this paper considers the location of edge pixels to adopt different embedding payload, so users can flexibly adjust the amount of data embedded in the image edge pixels according to their practical application. Thus, both the image quality and data payload are taken care of.

The rest of this paper is organized as follows: The related works conducts in Section 2. The Section 3 introduces the method proposed in this paper. The simulations of our proposed scheme and related works list in Section 4. Finally, the conclusion draws in Section 5.

2 Related works

This section introduces definition of signs mentioned in this paper, and two related works, including Lee et al.'s reduplicated exploiting modification direction (REMD) scheme and Chen et al.'s hybrid edge detector scheme, are fully reviewed in this section. The content of signs definition is as follows:

2.1 Definition of signs used in this paper

This part explains the meaning of signs used in this paper.

(1) $I(p_1, p_2, \ldots, p_n)$: The original image has n pixels, where $1 \le p_i \le 256$, and i is the location of pixel in the image.

(2) $I'(p_1', p_2', \ldots, p_n')$: The stego-image has n pixels, where $1 \le p_i' \le 256$, and i is the location of pixel in the image.

(3) $C(C_1, C_2, \ldots, C_n)$: The cover image has n pixels, where $1 \le C_i \le 512$, and i is the location of pixel in the image.
(4) $C^{(2)}(C_1^{(2)}, C_2^{(2)}, \ldots, C_n^{(2)})$: The stego-image has n pixels, wehre $1 \le C_i^{(2)} \le 512$, and i is the location of pixel in the image.
(5) s: The secret messages are randomly generated, $s \in \{0, 1\}$。
(6) $b_i^8\ b_i^7 \ldots b_i^1$: The binary expression of an original pixel p_i.
(7) b_i^1: The least significant bit (1LSB) from p_i. The b_i^1 is an indication bit for pixel restoration.
(8) g_i:A decimal value from the first 7 bits of p_i, i.e., the value of $\lceil (p_i + 1)/2 \rceil$.
(9) $g_i^{(1)}$ and $g_i^{(2)}$: The outcomes after two secret digits in a $(2n+1)$ -ary notational system have been embedded into g_i and $g_i^{(1)}$, respectively.
(10) $G = \{g_i | i = 1, 2, \ldots, n\}$, $G^{(1)} = \{g_i^{(1)} | i = 1, 2, \ldots, n\}$, and $G^{(2)} = \{g_i^{(2)} | i = 1, 2, \ldots, n\}$: The groups of n integer values falling within [0, 127].
(11) R: The bit set $\{b_i^1 | i = 1, 2, \ldots, n\}$ collecting from n least significant bits of every pixel value p_i.
(12) R': Indication bit set from adjusting R. The bit set is $\{b_i^{1\prime} | i = 1, 2, \ldots, n\}$.
(13) W, H: The image width and height, respectively.

2.2 Lee et al.'s scheme

Lee et al. [10] proposed a novel data hiding technique based on module operation. The technique transforms the secret message into $(2n+1)$-ary secret digit, and then it divides the original image into groups of n pixels, represented by $(p_1, p_2, \ldots, p_n)$, $n \ge 3$. Every original pixel p_i is transformed and represented by 8 bits, $(p_i)_{10} = (b_i^8\ b_i^7\ b_i^6\ b_i^5\ b_i^4\ b_i^3\ b_i^2\ b_i^1)_2$. Divide each pixel p_i into two sets (G, R). G is a collection composed of the 7 Most Significant Bits (MSB). It will be used to embed 2 secret digits. R is a collection composed of one bit, which is the Least one Significant Bits (1-bit LSB) of each pixel p_i. It will be used as an indicator. Therefore, secret digits can be embedded twice to a same pixel pair without using the original image and extra messages. Lee et al.'s scheme thus achieves a higher payload (about 1.87bpp), and keeps a fair image quality (about 45 dB). The detail phases of data embedding and message extracting are shown as below.

A. *Embedding Phase*

Step1: Divides the original image into groups of n pixels, and separate it into two sets (G, R) where $G = (g_1, g_2, \ldots, g_n)$ and $R = \{b_i^1 | i = 1, 2, \ldots, n\}$.
Step2: Apply EMD method on G set by Eq. (1) and Eq. (2) to calculate how to carry a $(2n+1)$-ary secret digit $s_j, j = 1, 2, \ldots, k$.

$$f(G) = \left(\sum_{i=1}^{n} g_i \times i \right) \bmod (2n+1) \tag{1}$$

$$D = \left(s_j - f(G) \right) \bmod (2n+1) \tag{2}$$

Eq. (3) can be used to evaluate how to change a certain g_i value of G set by at most add or substrate one. The values produced after G adjustment are denoted by $G^{(1)}$.

$$g_i^{(1)} = \begin{cases} g_i, & \text{if } s_1 = f_1 \\ g_D + 1, & \text{if } s_1 \neq f_1 \text{ and } D \leq n \\ g_{(2n+1)-D} - 1, & \text{if } s_1 \neq f_1 \text{ and } D > n \end{cases} \tag{3}$$

Step 3: Conduct the second layer embedding (hide the secret message s_2) to increase the capacity by applying Step 2 on $G^{(1)}$ set again. The adjusted $G^{(1)}$ set values are referred to as $G^{(2)}$ set. R' refers to the outcomes of the adjustment which values change in $G^{(2)}$ set as $g_i^{(2)} + 1$ or $g_i^{(2)} - 1$. Here are two cases to explain the meaning of the least significant bit $b_i^{1''}$ in stego-pixel value p_i'.

Case 1: R is adjusted to be the set of indicator bits, where each bit is set as " 0 ", i.e., $b_i^{1\prime} = 0$ for $i = 1, 2, \ldots, n$, and $g_i^{(2)} = g_i^{(1)}$, for $i = 1, 2, \ldots, n$.

Case 2: R is adjusted to be the set of indicator bits, where there is only one bit " 0 " or " 1 " in R. (Case 2–1) When $b_i^{1\prime} = 0$ and $b_j^{1\prime} = 1$, for $j = 1, 2, \ldots, n$ and $j \neq i$, $g_i^{(2)} = g_i^{(1)} - 1$, and $g_j^{(2)} = g_j^{(1)}$ (Case 2–2) When $b_i^{1\prime} = 1$ and $b_j^{1\prime} = 0$ for $j = 1, 2, \ldots, n$ and $j \neq i$, $g_i^{(2)} = g_i^{(1)} + 1$ and $g_j^{(2)} = g_j^{(1)}$.

Step 4: Follow Step 1 to Step 3, we calculate $p_i' = 2g_i^{(2)} + b_i^{1\prime}$ for $i = 1, 2, \ldots, n$. An embedded stego-image is obtained.

B. *Extracting Phase*

Step 1: Divide the stego-image into groups of n values. Then each group $(p_1', p_2', \ldots, p_n')$ of pixel values can be partitioned into two sets, $G^{(2)}$ and R', where $G^{(2)}$ is a collection of the most significant 7-bit plane of every pixel and the least significant one bit plane.

Step 2: Use the group $G^{(2)}$ as an input of the function f in Eq. (1) to retrieve the second secret digit s_2 in $(2n+1)$-ary notational system. Then use the indicator bit set R' to restore $G^{(2)}$ to $G^{(1)}$.

Step 3: After restoring $G^{(1)}$, repeat Step 2 by taking $G^{(1)}$ as the input of function f in Eq. (1) to retrieve the first secret digit s_1.

2.3 Chen et al.'s scheme

Chen et al. [5] employs a LSB substitution technique [2] as a fundamental stage and invents a hybrid edge detector combining fuzzy edge detector and canny edge detector. This mixed edge detector helps produce a better stego-image. The experimental result of Chen et al.'s scheme demonstrates that it not only achieves high embedding capacity but also enhances the quality of the stego-image. It allows a flexible amount of embedded data and can effectively resist the image steganalysis. The processes of data embedding and extracting are given as follows.

A. *Embedding Phase*

Step 1: Employ hybrid edge detection to detect edge image from the original image.

Step 2: Divides the original image I into groups of n pixels as (p_1, p_2, p_n).

Step 3: The first value p_1 of each pixel groups uses its to record the status whether other pixels $(p_2, \ldots, p_n)$ are edge pixels. If p_i is non-edge pixel, the record status is "0"; If

p_i is an edge pixel, the record status is "1". Subsequently, the status of pixels from p_2 to p_n is replaces p_1 on least significant bits via (n-1)-bits LSBs.

Step 4: To embed the secret message s. Because the secret message is embedded into ($p_2, \ldots, p_n$), the data embedding process has to base on the edge image. In non-edge pixels, secret message of x-bits is embedded via x LSBs to directly replaces p_i on least significant bits; otherwise, in edge pixels, secret message of y-bits is embedded via y LSBs to directly replaces p_i on least significant bits. Values after embedding are represented as $p_i^{'}$, in which $i = 2, 3, \ldots, n$.

Step 5: Follow Step 2 to Step 4 is thus obtained a stego-image I'.

B. *Extracting Phase*

Step 1: Divide the stego-image I' into groups of n values.

Step 2: Based on the (n-1) LSBs in pixel $p_1^{'}$, which is used to extract out the status whether other pixels ($p_2, \ldots, p_n$). If what is extracted out from $p_1^{'}$ is recorded as "0", it means this is a non-edge pixel, then extract out x bits; if what is extracted out from $p_1^{'}$ is recorded as "1", it means this is an edge pixel, then extract out y bit. After the above is done, the secret message can be extracted out.

3 The proposed scheme

This paper is based on the reduplicated exploiting modification direction (REMD), proposed by Lee et al. in 2008 [10]. The concept of REMD is that every original pixel is transformed into binary system, and most significant bits of 7 bits are extracted out. They are transformed into denary system to be embedded with two secret digits. At last, a stego-image is produced. To keep both good data capacity and image quality, this paper adopts edge detection to detect the location of edge in an image. Pixels in edge can be embedded with more data, and differences are not likely to be discovered there; but in non-edge, less data can be embedded. Because the amount of embedding data depends on the location of edge, users can apply the technique to some special fields. If high image quality is required by users, payload is lower, and vice versa. Edge is the place where edge detection can be employed to find out serious variations of textures in pixels, or where a special area can be captured. Classical edge operates include Sobel, Prewitt, Laplacian, and Canny [1, 12, 13]. Canny edge detection is adopted in this paper, and hides the data via REMD. Original pixel is increased or decreased by 1 at most.

In order to accomplish reversibility, image interpolation is conducted in this paper. Image interpolation features a simple calculation, and the final image quality is so good that it is very close to the original image. Image would be enlarged, but it can be restored to accomplish reversibility, while image quality and data payload are considered. This part can be divided into 3 phases. Each phase will be explained in the following description as shown in Fig. 1.

In Fig. 1, Phase 1 demonstrates the goal of this paper: Restore the image to its original state and accomplish reversibility. The image interpolation, a technique of image processing, is therefore used before embedding secret data. Phase 2 shows the procedure of embedding secret data, which includes three stages, as illustrated in Fig. 2. Phase 3 aims to extract out secret message and to restore the image. The three stages of the embedding secret data depicted in Phase 2 are:

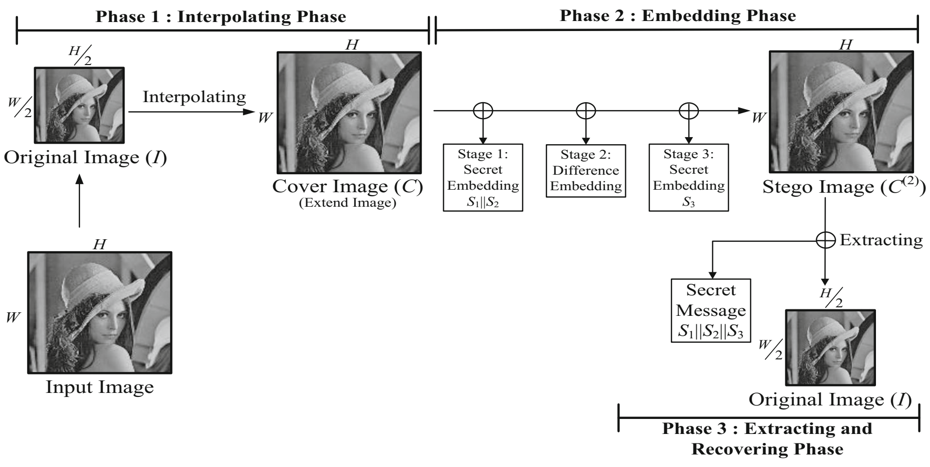

Phase 1 : Interpolating Phase
Phase 2 : Embedding Phase
H/2
W/2
Original Image (I)
Interpolating
H
W
Cover Image (C)
(Extend Image)
Stage 1: Secret Embedding S1||S2
Stage 2: Difference Embedding
Stage 3: Secret Embedding S3
Stego Image (C(2))
Extracting
Secret Message S1||S2||S3
Original Image (I)
Input Image
Phase 3 : Extracting and Recovering Phase

Fig. 1 The flow chart of proposed scheme

Stage 1: Apply Canny edge detector to the original image and make to an edge image. At the same time, decompose the cover image into *n*-bit plane slicing, where the gray-level cover image is separated into 8-bits plane slicing, and then, divide the cover image into 2 sets, *G* image and *R* image. *G* image is the 7-bits image plane, and *R* image is the least significant one bit plane. Next, embed secret data twice. First embedding is conducted by LSB substitution by edge detection (LSED), which embeds secret message S_1 into *G* image. This embedding depends on whether pixel is located at the edge to decide the amount of embedded data. LSB is taken on pixel, and then a $G^{(1)}$ image is created. Next, use the $G^{(1)}$ image and *R* image to conduct the second secret message embedding, S_2. REMD is taken to embed the secret message, which is realized through slight changing pixels. At last, a stego-image $C^{(2)}$ is created.

Stage 2: This step aims at restoring the image to its original state. Conduct difference-embedding on feature information of the image, that is, cover pixel and stego-pixel.

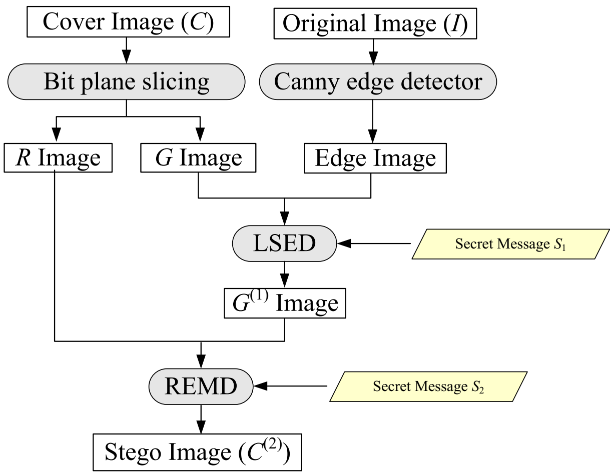


Fig. 2 The details of Embedding Stage 1

Stage 3: Although difference-embedding is taken to hide feature information into the image, some pixels are not covered. In order to increase the data payload, neighboring pixels, which share similarity, are used to embed secret message S_3.

Next, subsection A will introduce the preprocessing—secret message transferring. Subsection B and subsection C present the procedures of image interpolation and secret message embedding, respectively. And subsection D depicts how to extract out the secret message and restore the image.

A. ***Preprocessing of Secret Message Transferring***

Before the secret message is embedded, the secret message is transferred into $(2n+1)$-ary secret digits. Secret message s is divided into $\lfloor \log_2(2n+1) \rfloor$ segments and transferred into $(2n+1)$-ary secret digits. The digits are embedded to groups of n pixels. Its main purpose is to fully take advantage of the secret message and achieve high payload. For example when $n=3$, secret messages are converted into 7-ary secret digits. When the 7-ary secret digits are expressed as two-digits, the maximum value can be expressed as 66_7, and its decimal digits are $7^1 \times 6 + 7^0 \times 7 = 48_{10}$. Therefore, it can be learned that when the secret message is larger than 49_{10}, it is not able to embed secret messages. This paper aims to overcome this weakness and makes all the 7-ary digits presented in two-digits, $00_7 \sim 66_7$ be eligible for embedding secret messages. In this way, it is possible to reach a high data embedding capacity. The following illustrates how the secret message is transferred, as shown in Fig. 3.

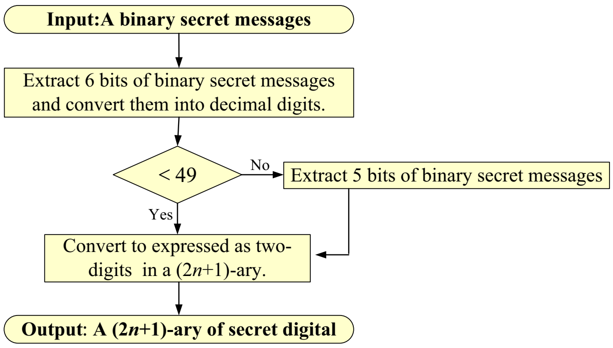
Step 1: Extract 6 bits of binary secret messages from a sequence of binary secret messages and convert them into decimal digits. If the value produced is less than or equal to 48, then go to Step 3; otherwise, go to Step 2.

Step 2: Extract 5 bits of binary secret messages from a sequence of secret messages and then convert them into two-digits as $(2n+1)$ -ary expressed as $s_1\ s_2$.

Step 3: Convert the binary secret messages into expressed as two-digits $s_1\ s_2$ in a $(2n+1)$ -ary notational system.

In the above preprocessing, the secret message is transferred and fully used. Next, this paper tries to perform reversibility, conducting difference-embedding on feature information to restore the cover image. Such is the phase of Image Interpolation, detailed in the subsection B.

Fig. 3 The flow chart of proposed scheme into preprocessing of secret message transferring


Input:A binary secret messages
Extract 6 bits of binary secret messages and convert them into decimal digits.
< 49
No
Extract 5 bits of binary secret messages
Yes
Convert to expressed as two-digits in a (2n+1)-ary.
Output: A (2n+1)-ary of secret digital

B. *Image Interpolation Phase*

To make reversibility possible, feature information of the image undergoes Difference-Embedding before delivery process and Image Interpolation. Image Interpolation, a technique of image processing [6, 7, 11], is conducted before embedding secret image. Regarding this, Interpolation by Neighbor Pixels (INP) [11] is employed in this paper, because in INP, neighboring pixels that share much similarity are used to produce enlarged interpolated image. Also, compared to other techniques of Image Interpolation, INP features a simpler calculation, and it keeps good image quality (25.53 dB). Among other techniques of Image Interpolation, NNI is the simplest method, but its image enlarged has the problem of Blocky effect, and its image quality is 20.26 dB. BLI is one of the most commonly used Image Interpolation techniques, but the image has the problem of Blurry effect, and its image quality is 20.22 dB. In 2009, Jung and Yoo [7] proposed NMI, which also uses the feature of adjacency in the image. Its image quality is 25.29 dB. In comparison, INP is the most ideal method; it therefore is employed in this paper.

Suppose that an original image I, sized $(H/2)\times(W/2)$, generates an interpolated image, which is the cover image C, sized $H\times W$, as Phase 1 in Fig. 1 illustrates. p_i is original pixel, C_i is cover pixel, and i is the location of pixel in the image. After the application of INP, an interpolated pixel is obtained, defined as $C_{i,j}$, in which $j=2, 3, 4$. Take p_1 in Fig. 4 as an example to illustrate how ($C_{1,2}$, $C_{1,3}$, $C_{1,4}$) is calculated. In addition, p_1 equals to C_1. Eq. (4) can be seen as a comparison. After the calculation, a cover image is produced, as illustrated in Phase 1 of Fig. 1.

$$C_{i,j}=\begin{cases} C_1=p_1, \\ C_{1,2}=\lfloor (p_1+(p_1+p_2)/2)/2 \rfloor, \\ C_{1,3}=\lfloor (p_1+(p_1+p_3)/2)/2 \rfloor, \\ C_{1,4}=\lfloor (C_{1,2}+C_{1,3})/2 \rfloor, \end{cases} \tag{4}$$

C. *Embedding Phase*

In this paper, secret message is embedded at C_i, as Fig. 4 shows. In other words, it is actually embedded at original pixel p_i. To start from the cover image of $H\times W$, C_i is first transferred into binary system and divided into 2 sets (G, R). G is primarily the location where secret message is embedded. Set $G=\{g_i|i=1, 2, \ldots, n\}$. R refers to the collection of what is extracted from the least significant bits of every pixel C_i's binary. b_i^1 is mainly used to record the pixels before the second embedding of secret message for the later restoration. Therefore, the action of repeated embedding can be made. Set $R=\{b_i^1|i=1, 2, \ldots, n\}$. Next, adopt Canny edge detection to find out the edge in the image, where more data are embedded. This paper

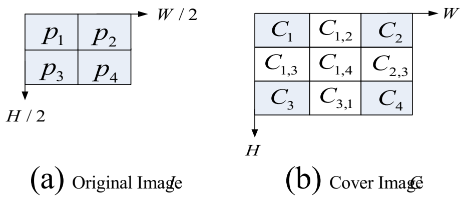


Fig. 4 The cover image generated from the original image by calculation INP

allows different data payload, which depends on if pixel is located at the edge. And then it adopts REMD to modify pixels in G, which is increased or subtracted by 1 at most to embed secret message for the second time. After first embedding of secret message is done in G, $G^{(1)}$ is generated, and it will be based on to conduct the second secret message embedding.

Then, image interpolation is used to conduct difference-embedding on the feature information of the image, that is, cover pixel and stego-image, and send it to the receiver without being recognized by human eyes. The receiver can restore the image to its original state through the feature information of difference-embedding.

Step 1: Employ canny edge detection to detect edge image from the original image.

Step 2: Convert cover pixel C_i into binary as $(C_i)_{10}=(b_i^8\ b_i^7\ b_i^6\ b_i^5\ b_i^4\ b_i^3\ b_i^2\ b_i^1)_2$ and then divide C_i into two sets (G, R).

Step 3: In the two sets (G, R), G is made up with the most significant Bits of 7 bits extracted from C_i and transferred into denary. The value obtained is g_i. g_i is taken to record the status whether other pixels ($p_2, \ldots, p_n$) are edge pixels. If p_i is not an edge pixel, the record status is "0"; If p_i is an edge pixel, the record status is "1". These records help extract out correct secret message in the future. Subsequently, g_1 is transferred into binary, and the status whether ($p_2, \ldots, p_n$) is an edge pixel or not directly replaces g_1 on Least Significant Bits via (n-1)LSBs. Then $g_1^{(1)}$ can be obtained.

Step 4: Conduct first embedding on the secret message s_1. Because the secret message is embedded into ($g_2, \ldots, g_n$), the embedding has to base on the edge image. In non-edge pixels, secret message of x bits is embedded via x LSBs to directly replaces g_i on least significant bits; otherwise, in edge pixels, secret message of y bits is embedded via y LSBs to directly replaces g_i on least significant bits. Values after embedding are represented as $G^{(1)}$, in which $x, y \in \{1, 2, 3 \ldots, n\}$, $i = 2, 3, \ldots, n$. R set undergoes the same step. Further details will be given in Step 5.

Step 5: Conduct the second embedding of secret message s_2. Apply $G^{(1)}$ to Eq. (5), Eq. (6), and Eq. (7), and a $(2n+1)$ -ary secret digit s_2 is produced. After embedding, $G^{(2)}$ is obtained. Now R set is needed to indicate which $g_i^{(2)}+1$ or which $g_i^{(2)}-1$ will be used to retrieve the indicator bits of the first secret message embedding. Table 1 displays the adjustment of R, which is represented as R', in which $i = 1, 2, \ldots, n$ and $j \neq i$.

Notably, it is oblivious known that the step 5 will case the exception of overflow/underflow after data embedding. From the Eq. (7), in the case of our proposed scheme, the operation of $g_i^{(2)}+1$ or $g_i^{(2)}-1$ will occasion the result of 128 or −1, called as exception, while the pixel belongs the boundary pixel, 128 or 0. In our embedding function listed in Eq. (5), it is operated under the modulo number of $2n+1$. This implies that $2n+1$ difference varieties are accommodated. In our pixel adjustment strategy, the weight summation of the adjusted pixels is increased/

Table 1 Setting value of $g_i^{(2)}$ and R'

$g_i^{(2)}$	R'	
	$b_i^{1\prime}$	$b_j^{1\prime}$
$g_i^{(1)}-1$	0	1
$g_i^{(1)}$	0	0
$g_i^{(1)}+1$	1	0

decreased by $2n+1$. In addition to that, we applied the one bit overhead to keep the information whether the $G^{(1)}$ has boundary pixel or not.

$$f\left(G^{(1)}\right) = \left(\sum_{i=1}^{n} g_i^{(1)} \times i\right) \mathrm{mod}(2n+1) \tag{5}$$

$$D = (s_2 - f)\mathrm{mod}(2n+1) \tag{6}$$

$$g_i^{(2)} = \begin{cases} g_i^{(1)}, & \text{if } s_2 = f \\ g_D^{(1)} + 1, & \text{if } s_2 \neq f \text{ and } D \leq n \\ g_{(2n+1)-D}^{(1)} - 1, & \text{if } s_2 \neq f \text{ and } D > n \end{cases} \tag{7}$$

Step 6: After Step 1 to Step 5, transform $G^{(2)}$ into denary, and add the eightieth bit R', which is the stego-pixel collection after embedding, $C^{(2)}(C_1^{(2)}, C_2^{(2)}, \ldots, C_n^{(2)})$.

Step 7: Apply Eq. (8) to calculate the difference d_p between original pixel C_i and stego-pixel $C_i^{(2)}$. Apply Eq. (9) to calculate the to-be-recorded difference d_p, which directly replaces the low right in the non-overlapping 2×2 block to adjust stego-pixel. $C_{i,4}^{(2)}$ is obtained after the adjustment. In terms of $C_1^{(2)}$, the value of Eq. (9) is taken to directly replace the place of $C_{1,4}^{(2)}$. This step aims at realizing the restoration of image.

$$d_p = C_i - C_i^{(2)} \tag{8}$$

$$C_{i,4}^{(2)} = C_i^{(2)} + d_p \tag{9}$$

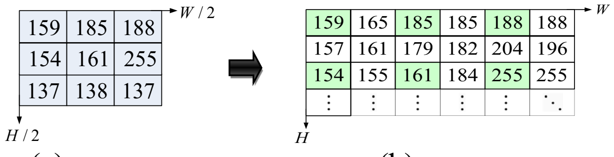
Step 8: Image Interpolation has been employed to record the difference d_p between cover pixel and stego-pixel on the position of $C_{i,4}^{(2)}$, but $C_{i,2}$ and $C_{i,3}$ are found not embedded with any secret message. In order to enhance the data capacity, one more embedding is needed. Use Eq. (10) and Eq. (11) to determine the secret message bits able to be embedded at $C_{i,2}$ and $C_{i,3}$ directly replaced the Least Significant bits of $C_{i,2}$ and $C_{i,3}$ by $l_{i,j}$ LSBs. The purpose of embedding can thus be achieved. This value is represented by $C_{i,2}^{(2)}$ and $C_{i,3}^{(2)}$.

$$d_{i,j} = \left\lfloor \left(C_i^{(2)} + C_{i+1}^{(2)}\right)/2 \right\rfloor - C_i^{(2)}, \text{ where } C_{i,2} \text{ and } C_{i,3} \tag{10}$$

$$l_{i,j} = \left\lfloor \log_2 \left|d_{i,j}\right| \right\rfloor, \text{ where } C_{i,2} \text{ and } C_{i,3} \tag{11}$$

Step 9: Repeat Step 8 and then a stego-image can be completed, as given in Phase 2 of Fig. 1.

The example will be given to show a real case of this paper. Suppose there is an original image I with pixel values (159, 185, 188, 154, 161, 255, 137, 138, 137), as shown in Fig. 5a. Apply INP to calculate, a cover image C is produced, as shown in Fig. 5b. Take $p_1 = 159$ in



159	185	188
154	161	255
137	138	137

(a) Original image I

159	165	185	185	188	188
157	161	179	182	204	196
154	155	161	184	255	255
⋮	⋮	⋮	⋮	⋮	⋱

(b) Cover image C

Fig. 5 The cover image generated from the original image using INP

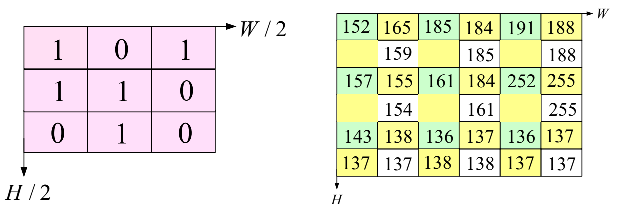
Fig. 5 as an instance to explain how to calculate ($C_{1,2}$, $C_{1,3}$, $C_{1,4}$): $C_{1,2}=\lfloor(159+((159+185)/2))/2\rfloor=165$, $C_{1,3}=\lfloor(159+((159+154)/2))/2\rfloor=157$, $C_{1,4}=\lfloor(165+157)/2\rfloor=161$, and so on. A cover image will be completed, as illustrated in Phase 1 of Fig. 1.

After image interpolation is used, a cover image sized $H\times W$ is produced, as illustrated in Fig. 5b. First, employ canny edge detection to find edges of the original image, then a canny edge image can be generated, as shown in Fig. 6a. Set $n=3$, divide the cover image C_i into 3 pixel groups and two secret messages, $s_1=(010100)_2$ and $s_2=(6\ 0\ 5)_7$. As Fig. 5b illustrates, the first pixel group is (C_1, C_2, C_3) =(159, 185, 188). Transfer every pixel into binary and divide them into G and R. g_i is defined by the Most Significant Bits of the former 7 bits, which are, respectively, $g_1=(1001111)_2=79$, $g_2=(1011100)_2=92$, $g_3=(1011110)_2=94$. Consequently, it can be learned that $G=(79,92,94)$ and $R=(1, 1, 0)$.

In order to use a more flexible way to insert secret message, g_1 is used to save if other pixels (p_2, p_3) are edge pixels. This ensures that the secret message bits can be correctly extracted out later. In Fig. 6a, it can be seen that (p_2, p_3)=(0, 1). Transfer $g_1=79$ into binary, $g_1=(1001111)_2=79$. Use (n-1)LSBs to directly replace g_1. Because $n=3$, there should be 2-bits LSBs to replace g_1. After g_1 is replaced by (p_2, p_3) =(0, 1), the result $g_1{}^{(1)}=(1001\underline{01})_2=77$ is obtained.

Conduct the first embedding on the secret message $s_1=(010100)_2$. As the secret message is going to be inserted into (g_2, g_3), it should depend on the edge image to do insertion. If it is a non-edge pixel (x), 1 bit of secret message is inserted; if it is an edge pixel (y), 1 bit of secret message is inserted. In Fig. 6a, it can be seen that the edge of (p_2, p_3) is 0, 1, which will be inserted into (g_2, g_3) according to edge pixels. In the same way, transfer (g_2, g_3) into binary, which are, respectively, $g_2=(1011100)_2$, $g_3=(1011110)_2$. Conduct the insertion of secret message. The corresponding value of g_2 is $p_2=0$. Because this is a non-edge pixel, 1 bit is inserted. After a secret message $s_1=(0)_2$ is extracted from s_1, g_2 is directly replaced by 1LSBs, and the result $g_2{}^{(1)}=(101110\underline{0})_2$ is obtained. The corresponding value of g_3 is $p_3=1$. This is an edge pixel, so 1 bit is inserted. After a secret message $s_1=(1)_2$ is extracted from s_1, g_3 is directly replaced by 1LSBs, and the result $g_3^{(1)}=(101111\underline{1})_2$ is obtained. After the embedding, $G^{(1)}$ is transferred into denary, and $G^{(1)}=(77, 92, 95)$.

Conduct the second embedding on the secret message $s_2=(6\ 0\ 5)_7$. Extract a secret digit $s_2=3$ from s_2 to conduct this second message embedding. Apply Eq. (5) to $G^{(1)}=(77, 92, 95)$.



1	0	1
1	1	0
0	1	0

(a) Canny edge image

152	165	185	184	191	188
	159		185		188
157	155	161	184	252	255
	154		161		255
143	138	136	137	136	137
137	137	138	138	137	137

(b) Stego-Image $C^{(2)}$

Fig. 6 The result of proposed scheme into embedding phase

After the calculation, it is known that $f = 0$. And use Eq. (6) and Eq. (7) to judge which $g_i^{(1)}$ should be adjusted, and it can learned that $D = 6$, $D > n$. Set $g_{(2n+1)-D}^{(2)}$ $-1 =$ $g_{7-6}^{(1)}$ $-1 = 77$ $-1 = 76$. After the embedding, it is known that $G^{(2)}$ =(76, 92, 95). Now $R = (1, 1, 0)$ is required to serve as an indicator pixel for the extraction of second-time embedded secret message in later step. R is adjusted via Table 1. After the adjustment, $R' = (0, 1, 1)$. Consequently, a group of stego-pixels is obtained, (152, 185, 191). Conduct the adjustment in order on (154, 161, 255, 137, 138, 137), the result (157, 161, 252, 143, 136, 136) is obtained, as well as a stego-image, as highlighted by green color in Fig. 6b.

In order to recover the image in later step, apply Eq. (8) to calculate 9 difference values, d_p, which are (159–152, 185–185, 188–191, 154–157, 161–161, 255–252, 137–143, 138–136, 137–136) = (7, 0, −3, −3, 0, 3, −6, 2, 1). Use Eq. (9) to calculate, and the value obtained directly replaces the low right in every 2×2 block in Fig. 6. The results of the replacement are (152 + 7, 185 + 0, 191 + (−3), 157 + (−3), 161 + 0, 252 + 3, 143 + (−6), 136 + 2, 136 + 1) = (159, 185, 188, 154, 161, 255, 137, 138, 137), as highlighted by white color in Fig. 6b.

At this moment, $C_{i,2}$ and $C_{i,3}$ are not embedded with secret message. In order to enhance data capacity, conduct embedding on these 2 places again. Use Eq. (10) to calculate the difference. After the calculation, $C_{i,2}$ respectively gets (16, 3, 0, 2, 45, 0, −4, 0, 0). And calculate the difference of $C_{i,3}$. The calculative result is (2, −7, 0, 173, 149, 0, 30, −58, 0). Insert a cluster of secret messages $s = (01010111010101001101001101001111)_2$. Use Eq. (11) to determine the embeddable secret message bits. The least significant bits of $C_{i,2}$ and $C_{i,3}$ are directly replaced by LSB. In this way, the embedding can be completed. Its value is represented by $C_{i,2}^{(2)}$ and $C_{i,3}^{(2)}$. The results, respectively, are (165, 184, 188, 155, 186, 255, 138, 137, 137) and (157, 148, 137, 182, 156, 138, 205, 231, 137), as highlighted by yellow color in Fig. 6b. After the above steps are done, obtained stego-image can be sent to the receiver.

D. ***Extracting and Recovering Procedure***

The following includes a list of steps that the receiver should take to extract out secret message and recover the image:

Step 1: Divide the stego-image into non-overlapping 2×2 blocks. Use Eq. (10) and Eq. (11) for calculation, and it can be learned what are the secret message bits embedded. The secret message can be directly extracted out from the Least Significant Bits of $C_{i,2}^{(2)}$ and $C_{i,3}^{(2)}$ by $l_{i,j}$ -LSBs.

Step 2: Divide the stego-image into binary digits, which are transferred from groups of n pixels ($C_1^{(2)}$, $C_2^{(2)}$, …, $C_n^{(2)}$). Divide them into two sets, $G^{(2)}$ and R'.

Step 3: Apply $G^{(2)}$ to Eq. (5), and then the secret digit of the second-time embedding, s_2 can be extracted. Use the indicator bit message of R' and follow Table 1 to restore $G^{(2)}$ to $G^{(1)}$.

Step 4: When restoring to $G^{(1)}$, extract out the first $g_1^{(1)}$ of $G^{(1)}$, which is used to extract out the status whether other pixels (p_2 , …, p_n) saved at this value are edge pixels via $(n-1)$ -bits LSBs. If the bit extracted out from $g_1^{(1)}$ is recorded as "0", it means this is a non-edge pixel, then extract out x bits; if the bit extracted out from $g_1^{(1)}$ is recorded as "1", it means this is an edge pixel, then extract out y bit. After the above is done, the secret message of the first-time embedding, s_1, can be extracted out.

Step 5: Use Eq. (12) to extract out the embedded difference value d_p, and use Eq. (13) for calculation. The cover pixel can be obtained. After the above is done, it can be restored to the cover image, as illustrated in Phase 3 of Fig. 1.

$$d_p = C_{i,4}^{(2)} - C_i^{(2)} \tag{12}$$

$$p_i = C_i^{(2)} + d_p \tag{13}$$

Now the receiver gets a stego-image sized $H \times W$, as shown in Fig. 6b. First, divide the stego-image into non-overlapping 2×2 blocks. Use Eq. (10) and Eq. (11) for calculation, and then the secret message can be taken out. Then, divide the stego-image into digits, which are transferred from groups of 3 pixels ($C_1^{(2)}$, $C_2^{(2)}$, …, $C_n^{(2)}$). Divide them into 2 sections, $G^{(2)}$ and R'. $g_i^{(2)}$ is represented by the most significant bits of the former 7 bits, which are, respectively, $g_1^{(2)} = (1001100)_2 = 76$, $g_2^{(2)} = (1011100)_2 = 92$, $g_3^{(2)} = (1011111)_2 = 95$. So $G^{(2)}$ =(76, 92, 95) and R' =(0, 1, 1). Apply $G^{(2)}$ =(76, 92, 95) to Eq. (5) to take out the secret digit of $s_2 = 6$. After the above is done, the secret message of the second-time embedding, $s_2 = (6\ 0\ 5)_7$, can be extracted. Then, following up the Table 1, $G^{(2)} = (76, 92, 95)$ can be restored to $G^{(1)} = (77, 92, 95)$.

Next, use $g_1^{(1)} = (1001101)_2$ from $G^{(1)} = (77, 92, 95)$ to extract out the status whether other pixels (p_2, p_3) saved at this value are pixels via 2-bits LSBs, and 0 and 1 can be obtained. Therefore, it can be learned that if the $g_1^{(1)}$ is recorded as "0", this is a non-edge pixel. Then extract out a bit $s_1 = (0)_2$ from LSBs of $g_2^{(1)}$ =$(101110\underline{0})_2$. If $g_3^{(1)}$ is recorded as "1", this is an edge pixel. Then extract out a bit $s_1 = (1)_2$ from LSBs of $g_3^{(1)} = (101111\underline{1})_2$. After the above is done, the secret message of the first-time embedding $s_1 = (010100)_2$ can be extracted out. Lastly, extract out the difference value d_p. Use Eq. (12) to calculate and get $159 - 152 = 7$. And use Eq. (13) to calculate and get the original pixel $152 + 7 = 159$. After the above steps are completed, the original image can be restored, as illustrated in Fig. 6a.

4 Experimental results

In the experiment, we used 15 cover images as its experimental objects. Every cover image, all sized 512×512, is an 8-bits grayscale image. They are "Baboon", "Jet(F14)", "GoldHill", "Boats", "Sailboat", "Toys", "Barbara", "Zelda", "Tiffany", "Jet(F16)", "Lena", "Cartoon", "Peppers", "House"及"Splash", as shown in Fig. 7. In the experimental process, secret message is made up a cluster of 0 and 1, which are randomly produced.

Bit per pixel (bpp) and peal signal to noised ratio (PSNR) are used to evaluate efficiency of the inserting method. The pl is used for the evaluation of payload using Eq. (14). Payload is to calculate how many bits can be inserted into a pixel carried by an image, that is, Bit Rate. $|S|$ is the total inserting amount of secret message. H and W are respectively the height and width of the image.

$$pl = \frac{|S|}{H \times W} (\text{bpp}) \tag{14}$$

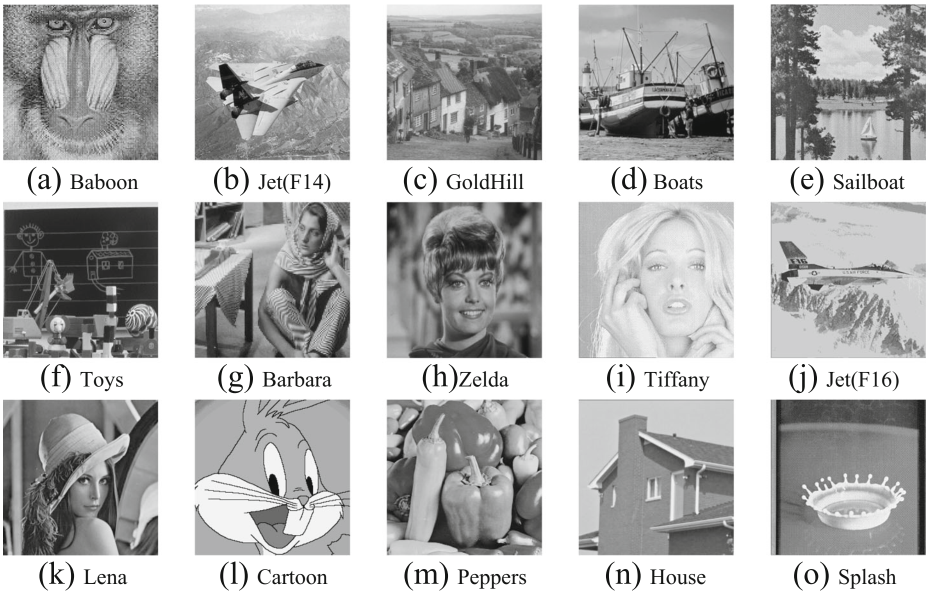


Fig. 7 Fifteen 8- bit grayscale test images with is 512 × 512

PSNR is used to identify the image quality. When PSNR is larger, the degree of distortion is smaller; When PSNR is smaller, the degree of distortion is larger. Its definition is as presented in Eq. (15).

$$PSNR = 10 \times \log_{10}\left(\frac{255^2}{MSE}\right) \tag{15}$$

The definition of mean square error (MSE) is shown as Equation (16):

$$MSE = \frac{1}{H \times W}\sum_{i=1}^{H}\sum_{j=1}^{W}\left(\text{Cover Image}_{(i,j)} - \text{Stego Image}_{(i,j)}\right)^2 \tag{16}$$

This paper is based on REMD, which can repeat hiding secret message to a same pixel pair. During the first embedding, Canny Edge Detection is used to detect edge pixels and non-edge pixels, where different amount of data is embedded. During the second embedding, REMD is utilized, which only needs to slightly change neighboring pixels to achieve the purpose of embedding. Lastly, this paper adopts Image Interpolation to conduct difference-embedding on the image's feature information before sending it to the receiver. According to the above-mention, we applied the three proposed types, such as Type A (REMD + Interpolation), Type B (Canny + Interpolation), and Type C (Canny + REMD + Interpolation, to demonstrate our proposed performances.

For the Type A, we first discuss the maximum payload and image quality under different *n* values. Table 2 lists the 15 images' different performances of maximum payload (Payloadmax) under various n values and image quality. The maximum payload includes Pure-Payload (PayloadPure) and the message hidden in feature information. In the performances of Maximum Payload, "Barbara", "Baboon", and "Jet(F14)" have the highest payload, which are, respectively, 1.36, 1.51, and 1.39bpp. Textures of these 3 images are most complex, and they

Table 2 The performance of Ours Type A approach in terms of maximum payload (Payload_{max}) and image quality (PSNR)

n	3		4		5	
Images	Payload_{max}	PSNR	Payload_{max}	PSNR	Payload_{max}	PSNR
Baboon	1.51	41.32	1.42	41.56	1.31	41.68
Jet(F14)	1.39	42.23	1.29	42.56	1.18	42.66
GoldHill	1.20	44.35	1.11	44.80	1.00	45.04
Boats	1.20	43.19	1.10	43.55	0.99	43.73
Sailboat	1.22	43.41	1.12	43.81	1.01	43.95
Toys	1.09	43.72	1.00	44.16	0.89	44.25
Barbara	1.36	41.98	1.27	42.32	1.16	42.39
Zelda	1.07	45.45	0.97	46.11	0.87	46.31
Tiffany	1.06	45.51	0.96	46.13	0.85	46.39
Jet(F16)	1.11	44.00	1.02	44.43	0.91	44.60
Lena	1.14	44.22	1.04	44.68	0.93	44.92
Cartoon	1.04	40.27	0.94	40.39	0.83	40.36
Peppers	1.11	44.38	1.01	44.84	0.90	45.02
House	1.07	44.70	0.98	45.25	0.87	45.39
Splash	1.00	45.09	0.90	45.69	0.79	45.93
Average	1.17	43.59	1.08	44.02	0.97	44.17

have the best performance in payload among the 15 images. Moreover, their PSNR is kept around 41 dB, which demonstrates an acceptable image quality. Among the 15 images, "Splash" has the worst performance in Maximum Payload, 0.79. Textures of "Splash" are relatively smoother, so it cannot have good payload. Only images with complex textures are able to generate better payload. In terms of the condition under different n values, when n =3, the average payload of every image can achieve 1.17bpp, and PSNR is kept around 43.59; when n =4, the average payload can achieve 1.08, and PSNR is kept around 44.02; When n =5, the average payload can achieve 0.97bpp and PSNR is about 44.17 dB.

Next, we compare our experimental result with the Kuo et al.'s method depicted in Table 3. From Table 3, it can be seen that this paper achieves a better performance in payload than Kuo et al. The average payload of this paper is 1.22bpp, while the average payload of Kuo et al.'s

Table 3 Comparison results of Type A with Kuo et al.'s method

Test Images	Type A(RMED + Interpolation)		Kuo et al. Scheme [8]	
	Payload_{max}(bpp)	PSNR(dB)	Payload_{max}(bpp)	PSNR(dB)
Baboon	1.51	41.32	1.53	30.09
Jet(F16)	1.11	44.00	0.81	35.93
Lena	1.14	44.22	0.97	35.70
Peppers	1.11	44.38	0.78	37.84
Average	1.22	43.48	1.03	34.89

Table 4 Each texture image of quantitative indexs, where each image with sized 256 × 256

Test Images	Non-edge Pixels (Bits)	Percentage (%)	Edge Pixels (Bits)	Percentage (%)
Baboon	54,783	84 %	10,753	16 %
Jet(F14)	57,593	88 %	7943	12 %
GoldHill	57,926	88 %	7610	12 %
Boats	59,232	90 %	6304	10 %
Sailboat	59,346	91 %	6190	9 %
Toys	59,471	91 %	6065	9 %
Barbara	59,779	91 %	5757	9 %
Zelda	59,794	91 %	5742	9 %
Tiffany	59,846	91 %	5690	9 %
Jet(F16)	59,880	91 %	5656	9 %
Lena	60,486	92 %	5050	8 %
Cartoon	60,677	92 %	4859	8 %
Peppers	60,808	93 %	4728	7 %
House	61,056	93 %	4480	7 %
Splash	61,185	93 %	4351	7 %

method is 1.03bpp. The difference is 0.19bpp. That is to say, in this paper every cover pixel can carry 0.19bpp more than that of Kuo et al.'s method. Secondly, in *Kuo* et al.'s method, average PSNR of stego images is 34.89 dB, while this paper has 43.48 dB, which is also a better performance. However, same as Kuo et al.*'s* method, this paper cannot effectively achieve both high payload and good image quality. In addition, in the performance of pure-

Table 5 The Type B (Canny + Interpolation) performance of Payload$_{pure}$ and PSNR (when $x = 0$, $n = 3$ and $y \geq 1$)

$x = 0$, $n = 3$								
y	1		3		5		6	
Images	Payload$_{pure}$ (bpp)	PSNR (dB)	Payload$_{pure}$ (bpp)	PSNR (dB)	Payload$_{pure}$ (bpp)	PSNR (dB)	Payload$_{pure}$ (bpp)	PSNR (dB)
Baboon	0.68	51.56	0.74	49.65	0.81	41.97	0.87	39.36
Jet(F14)	0.54	51.02	0.59	49.71	0.64	42.94	0.68	40.76
GoldHill	0.35	50.61	0.40	49.42	0.45	43.06	0.50	40.47
Boats	0.34	50.46	0.37	49.46	0.42	43.83	0.45	41.51
Sailboat	0.36	50.49	0.40	49.55	0.44	43.89	0.47	41.62
Toys	0.28	50.19	0.27	49.26	0.32	43.18	0.35	41.89
Barbara	0.51	50.70	0.54	49.80	0.58	44.01	0.62	41.46
Zelda	0.22	50.23	0.25	49.33	0.30	43.76	0.34	41.49
Tiffany	0.19	50.15	0.22	49.19	0.27	44.11	0.31	42.19
Jet(F16)	0.25	50.12	0.28	49.31	0.33	44.08	0.36	41.33
Lena	0.28	50.25	0.30	49.52	0.34	44.41	0.37	42.29
Cartoon	0.16	53.55	0.19	53.46	0.24	46.45	0.28	41.95
Peppers	0.24	50.21	0.27	49.48	0.30	44.71	0.33	42.40
House	0.21	50.06	0.23	49.34	0.27	44.20	0.30	41.91
Splash	0.13	49.86	0.16	49.18	0.20	44.50	0.23	42.78

Table 6 The Type B (Canny + Interpolation) performance of Payload$_{pure}$ and PSNR (when $x = 1$, $n = 3$ and $y \geq 1$)

$x = 1$, $n = 3$								
y	1		3		5		6	
Images	Payload$_{pure}$ (bpp)	PSNR (dB)	Payload$_{pure}$ (bpp)	PSNR (dB)	Payload$_{pure}$ (bpp)	PSNR (dB)	Payload$_{pure}$ (bpp)	PSNR (dB)
Baboon	0.82	51.05	0.88	49.32	0.95	41.87	1.01	39.29
Jet(F14)	0.69	50.52	0.73	49.29	0.79	42.95	0.83	40.55
GoldHill	0.50	50.10	0.54	48.95	0.60	42.99	0.65	40.35
Boats	0.49	49.95	0.53	49.08	0.57	43.76	0.60	41.46
Sailboat	0.52	49.97	0.55	49.14	0.59	43.86	0.62	41.53
Toys	0.39	49.68	0.42	48.90	0.47	43.05	0.50	41.83
Barbara	0.67	50.21	0.70	49.41	0.74	43.73	0.77	40.97
Zelda	0.37	49.73	0.40	48.91	0.45	43.61	0.49	41.41
Tiffany	0.35	49.62	0.38	48.81	0.43	43.86	0.47	41.96
Jet(F16)	0.41	49.61	0.44	48.87	0.48	43.81	0.51	41.27
Lena	0.43	49.74	0.46	49.09	0.50	44.35	0.52	42.20
Cartoon	0.31	52.32	0.34	52.24	0.40	46.09	0.43	42.08
Peppers	0.40	49.70	0.43	49.03	0.46	44.48	0.49	42.15
House	0.37	49.55	0.39	48.92	0.43	44.12	0.46	41.98
Splash	0.29	49.34	0.32	48.77	0.36	44.36	0.39	42.69

payload, in the proposed method of Type A, when $n = 3$, meaning 3 pixels are a group, 2 pixels at most are modified when inserting secret message, and 0 pixel at least. So the message

Table 7 The Type C (Canny + REMD + Interpolation) performance of Payload$_{pure}$ and PSNR when $x = 1$, $n = 3$ and $y \geq 1$

$x = 1$, $n = 3$								
y	1		3		5		6	
Images	Payload$_{pure}$ (bpp)	PSNR (dB)	Payload$_{pure}$ (bpp)	PSNR (dB)	Payload$_{pure}$ (bpp)	PSNR (dB)	Payload$_{pure}$ (bpp)	PSNR (dB)
Baboon	1.45	39.28	1.52	38.96	1.62	37.87	1.72	37.06
Jet(F14)	1.31	40.73	1.35	40.39	1.43	39.29	1.50	38.52
GoldHill	1.09	43.44	1.14	42.93	1.22	41.18	1.30	40.39
Boats	1.07	42.40	1.10	42.03	1.16	40.76	1.22	39.83
Sailboat	1.11	42.54	1.14	42.20	1.20	40.83	1.26	40.30
Toys	0.95	42.77	0.99	42.39	1.05	41.06	1.10	40.25
Barbara	1.27	40.17	1.31	39.93	1.36	39.17	1.41	38.73
Zelda	0.94	46.24	0.97	45.45	1.05	43.11	1.11	42.25
Tiffany	0.91	46.35	0.94	45.27	1.02	43.08	1.07	42.73
Jet(F16)	0.98	43.59	1.02	43.20	1.08	41.79	1.12	40.74
Lena	1.01	43.6	1.04	43.26	1.09	41.86	1.13	41.20
Cartoon	0.71	41.63	0.76	40.94	0.84	40.35	0.89	39.71
Peppers	0.97	44.06	1.00	43.64	1.05	42.29	1.09	41.44
House	0.95	45.15	0.98	44.64	1.03	43.17	1.07	42.30
Splash	0.83	45.54	0.86	44.99	0.92	43.07	0.96	42.36

hidden in the feature information roughly makes ((2–0)/2)/3 ≈ 0.333. Lastly, after Maximum Payload subtracts the message hidden in feature information, pure-payload is produced, which can also be generated in this paper; Kuo et al. did not particularly show the performance of pure-payload in their paper, which, however, can be deplored in their paper: when $n = 4$, which is the best moment in their paper, 3 pixels at most are inserted, and 0 pixel at least, so the message hidden in the feature information roughly makes ((3–0)/2)/4 ≈ 0.375. After subtraction, the result is pure-payload. Therefore, it can be inferred that this paper has a better performance in pure-payload than Kuo et al.'s method.

The Type B applies the Edge Detection to detect an edge image from the original image. The pixel in non-edge or edge can tolerate the change which the difference number of secret message bits are inserted into. Here, we employ two conditions, such as "Quantitative indicator of textures in each image" and "changes of parameters x and y", to show the simulation of the edge detection and the experimental result of the proposed scheme. In this condition 1, we utilize the canny edge detection approach to detect edge bits and non-edge bits

Table 8 The performance of Ours Type C on 512 × 512 stego-image when $x = y$ and $n = 3$

$n = 3$			
(x, y)	Baboon	Cartoon	Lena
(1,1)	PSNR=39.28dB Payload$_{pure}$=1.45bpp (381365bits)	PSNR=41.63dB Payload$_{pure}$=0.71bpp (186734bits)	PSNR=43.69dB Payload$_{pure}$=1.01bpp (264261bits)
(2, 2)	PSNR=38.91dB Payload$_{pure}$=1.64bpp (429772bits)	PSNR=39.97dB Payload$_{pure}$=1.01bpp (265663bits)	PSNR=42.67dB Payload$_{pure}$=1.22bpp (319781bits)
(3, 3)	PSNR=37.63dB Payload$_{pure}$=1.86bpp (487583bits)	PSNR=37.25dB Payload$_{pure}$=1.40bpp (368161bits)	PSNR=40.12dB Payload$_{pure}$=1.51bpp (396189bits)

Table 8 (continued)

(4, 4)	PSNR=35.64dB Payload$_{pure}$=2.15bpp (562657bits)	PSNR=36.08dB Payload$_{pure}$=1.89bpp (495170bits)	PSNR=36.94dB Payload$_{pure}$=1.92bpp (504063bits)
(5, 5)	PSNR=34.30dB Payload$_{pure}$=2.43bpp (638247bits)	PSNR=34.02dB Payload$_{pure}$=2.40bpp (629358bits)	PSNR=34.89dB Payload$_{pure}$=2.42bpp (6336896bits)
(6, 6)	PSNR=32.49dB Payload$_{pure}$=3.10bpp (812910bits)	PSNR=32.06dB Payload$_{pure}$=3.05bpp (800322bits)	PSNR=33.06dB Payload$_{pure}$=3.01bpp (788115bits)

in each image, which will help quantification and later experimental, as displayed in Table 4. From the Table 4, it also shows that “Baboon”, “Jet(F14)”, “GoldHill”, and “Boats” have the highest percentage of edge pixels.

For the condition 2, we present how the experiment compares the 15 images sized 512×512 and analyzes their different x and y variables. The parameter x denotes the number of bits inserted into non-edge pixels and the parameter y denotes the number of bits inserted into edge pixels. In other words, x bits of secret message replace the place of x LSBs on every non-edge pixel, and y bits of secret message replaces the place of y LSBs on every edge pixel.

Suppose $n=3$ and $x=0$, parameter y is larger or equals to 1. In other words, the experimental result shown in Table 5 embeds secret on edge pixel and uses LSBs to make replacement on every pixel. Meanwhile, non-edge pixels are hidden with any message. To look at the performance of Pure-Payload listed in Table 5, “Barbara”, “Baboon”, and “Jet(F14)”, these three images have better performance. When $y=6$, their Pure-Payload is respectively 0.62, 0.87, and 0.68bpp. These three images have relatively complex textures. In contrast, “Cartoon” and “Splash” have the lowest Pure-Payload. When $y=6$, their Pure-Payload is relatively 0.28 and 0.23bpp. These two images have smoother textures than other images, so they have less edge pixels and embed fewer messages. For showing other simulations, we set the parameters, e.g. $x=1$, $n=3$, and $y\geqq 1$, to be an others cases. The experimental result is listed in Table 6.

The Type C (Canny + REMD + Interpolation) focuses on the employment of different data amount embedded to edge or non-edge pixels. It is expected that users can effectively apply this method to their practical application. They can flexibly adjust the amount of data embedded to edge pixels. Related experiment is conducted to test if the proposed method can include both the image quality and data payload. Table 7 shows the performance in terms of pure-embedding payload and image quality when parameters are set as $x=1$, $n=3$ and $y\geq 1$. From Table 7, "Baboon" has the best performance, followed by "Barbara" and "Jet(F14)". When $y=6$, their pure-payload is respectively 1.72, 1.41, and 1.50bpp. Among other images, "Cartoon" and "Splash" have the lowest pure-paylaod. When $y=6$, their pure-payload is respectively 0.89 and 0.96bpp. In the performance of image quality, from Table 7, it can be observed that when y is 1 to 6, PSNR of every image does not significantly decrease. Image quality does decrease with increasing parameter x and y. The lowest image quality is kept at 37 dB and above. In addition to that, for seeking more embedding capacity, Table 8 shows showing the simuation of all stego images when x and y are same and $n=3$.

Take the 3 images, "Baboon", "Cartoon", and "Lena", as an example. The "Baboon" at first has the image quality at 39.28 dB and pure-payload roughly at 1.45bpp(381365bits). When the combination of x and y is (6, 6), PSNR is 32.49 dB, and the image quality is kept within an acceptable level. The other 2 images, "Cartoon" and "Lena", have smoother textures. At first, their image quality is 41 dB, which is satisfactory image quality to human eyes. Their pure-payload is respectively 0.71bpp and 1.01bpp. All combinations of parameters have gone through the experiment, and when the combination is (6, 6), the pure-payload and image quality of these 2 images are not far from the combination (6, 6) applied to "Baboon".

Table 9 The comparison result of Our Type C with Jung and Yoo Scheme

Test Images	Ours Type C ($x=3$, $y\geq 1$)		Jung and Yoo Scheme [7]	
	Payload$_{pure}$(bpp)	PSNR(dB)	Payload$_{pure}$(bpp)	PSNR(dB)
Baboon	1.90	37.23	1.37	37.11
Jet(F14)	1.77	38.19	1.15	38.69
GoldHill	1.60	39.57	0.79	42.19
Boats	1.59	39.05	0.71	40.60
Sailboat	1.61	39.30	0.77	40.58
Toys	1.51	39.27	0.50	41.46
Barbara	1.75	37.94	1.07	38.11
Zelda	1.48	40.69	0.52	45.64
Tiffany	1.47	40.69	0.45	45.95
Jet(F16)	1.52	39.87	0.54	41.91
Lena	1.53	39.80	0.63	42.22
Cartoon	1.43	37.17	0.29	38.51
Peppers	1.51	40.01	0.56	42.86
House	1.46	40.94	0.49	43.98
Splash	1.41	40.82	0.30	45.28
Average	1.57	39.37	0.67	41.67

Therefore, it can be concluded that after the combination of x and y is (6, 6), every image can keep its image quality within an acceptable level, 33 ± 1 dB. This indicates that this paper can make flexible adjustment to have both high data payload and good image quality.

Table 8 compares the performances of pure-payload and image quality between the method proposed by Jung et al. and Ours Type C (Canny + REMD + Interpolation). 15 images sized 512×512 are the experimental objects. When $x = 3$ and $n = 3$, the performance of Ours Type C is not far from that of the method proposed by Jung et al. In the average value listed in Table 9, it can be observed that this paper utilizes Canny detection to effectively enhance payload. The average payload of Ours Type C is 1.57bpp, 0.9bpp higher than that of Jung et al's method, which proves Ours Type C has a significantly better performance in average payload. In regard to image quality, the stego images in Jung et al.'s method have PSNR at 41.67 dB, and PSNR of the stego images in this paper is 39.37 dB, which is a worse performance. However, consider that the method proposed in this paper can be flexibly used to practical application, that is, it is possible to flexibly adjust the amount of data embedded into edge pixels. Therefore, it can be reasonably said that the method proposed in this paper is better than the method proposed by Jung et al. Also, it effectively includes both payload and image quality.

5 Conclusions

This paper finds the advantage of that in REMD, it only needs to slightly change neighboring pixels for the purpose of embedding. Image interpolation is adopted to conduct difference-embedding on the image's feature information before it is sent to the receiver to achieve reversibility. Most importantly, this paper aims to give a method useful in practical application, allowing flexible adjustment on data amount to be inserted into edge pixels. Experiments are conducted to prove that this paper can effectively enhance data payload and keep acceptable image quality. The experimental results demonstrate that data payload can achieve 3.01bpp, ensuring this is a reversible data hiding with high payload. Meanwhile, image quality is kept at about 33 ± 1 dB.

Acknowledgments This work was supported in part by the Ministry of Science and Technology, Taiwan, under Contract MOST 104-2221-E-324-013.

References

1. Allebach J, Wong PW (1996) Edge-directed interpolation. Proceedings of IEEE International Conference on Image Processing:707–710
2. Chan CK, Cheng LM (2004) Hiding data in images by simple LSB substitution. J Pattern Recognit 37(3): 469–474
3. Chang YT, Huang CT, Lee CF, Wang SJ (2013) Image Interpolating based data hiding in conjunction with pixel-shifting of histogram. J Supercomput 66(2):1093–1110
4. Chang CC, Kieu D, Chou YC (2007) Reversible data hiding scheme using two steganographic images. Proceedings of 2007 I.E. TENCON:1–4

5. Chen WJ, Chang CC, Hoang Ngan Le T (2010) High payload steganography mechanism using hybrid edge detector. Expert Syst Appl 37(4):3292–3301
6. Jensen K, Anastassiou D (1995) Subpixel edge localization and the interpolation of still images. IEEE Trans Image Process 4:285–295
7. Jung KH, Yoo KY (2009) Data hiding method using image interpolation. Comput Stand Interfaces 31(2): 465–470
8. Kuo WC, Hsu WT (2010) Reversible data hiding scheme based on EMD method. Proceedings of 2010 Taiwan Academic Network Conference (in Chinese version)
9. Lee CF, Chang CC, Pai PY, Liu CM (2015) Adjustment hiding method based on exploiting modification direction. Int J Netw Secur 17(5):607–618
10. Lee CF, Chuang LY, Chang CC (2008) Hiding information employing reduplicating embedding. Proceedings of 2008 I.E. Asia-Pacific Services Computing Conference:825–828
11. Lee CF, Huang YL (2012) An efficient image interpolation increasing payload in reversible data hiding. Expert Syst Appl 39(8):6712–6719
12. Lehmann TM, Gonner C, Spitzer K (1999) Survey: interpolation methods in medical image processing. IEEE Trans Med Imaging 18(11):1049–1075
13. Li X, Orchard MT (2001) New Edge-Directed Interpolation. IEEE Trans Image Process 10(10):1521–1527
14. Qin C, Chang CC, Chiu YP A novel joint data-hiding and compression scheme based on SMVQ and image inpainting. IEEE Trans Image Process 23(3):696–978
15. Qin C, Chang CC, Hsu TJ (2015) Reversible data hiding scheme based on exploiting modification direction with two steganographic images. Multimed Tools Appl 74(15):5861–5872
16. Qin C, Zhang XP (2015) Effective reversible data hiding in encrypted image with privacy protection for image content. J Vis Commun Image Represent 31:154–164
17. Shen SY, Huang LH (2015) A data hiding scheme using pixel value differencing and improving exploiting modification directions. Comput Secur 48:131–141
18. Sun HM, Weng CY, Wang SJ, Yang CH (2013) Data embedding in image-media using weight-function on modulo operation. ACM Trans Embed Comput Syst 12:21:1–21:10
19. Unser M, Aldroubi A, Eden M (1995) Enlargement or reduction of digital images with minimum loss of information. IEEE Trans Image Process 4(3):247–258
20. Zhang XP (2013) Reversible data hiding with optimal value transfer. IEEE Trans Multimed 15(2):316–325
21. Zhang X, Wang S (2006) Efficient steganographic embedding by exploiting modification direction. IEEE Commun Lett 10:781–783
22. Zhang X, Wang S (2006) Efficient steganographic embedding by exploiting modification direction. IEEE Commun Lett 11(10):781–783

Chin-Feng Lee received her Ph.D. in Computer Science and Information Engineering from National Chung Cheng University, Taiwan. She is currently aprofessor of Information Management at Chaoyang University of Technology, Taichung, Taiwan. Her research interests include steganography, image processing, information retrieval and data mining.

Chi-Yao Weng received his Ph.D. in Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan. He is currently an Assistant Professor with the Department of Computer Science at National Pingtung University, Pingtung, Taiwan. His current research interests include watermarking, media design, and multimedia security.

Kai-Chin Chen received herMaster degree inInformation Management from Chaoyang University of Technology, Taichung, Taiwan. Herresearch interests include information hiding, steganography, and image processing.