Reversible Data Hiding Based on Geometric Structure of Pixel Groups

Zhi-Hui Wang[1], Xu Zhuang[2], Chin-Chen Chang[3,4], Chuan Qin[4] and Yan Zhu[5]
(Corresponding author: Chin-Chen Chang)

School of Software, Dalian University of Technology[1]
Economy and Technology Development Area, Dalian 116620, China
Department of Computer Science and Technology, Southwest JiaoTong University[2]
Jinniu District, Chengdu, China, 610031
Department of Information Engineering and Computer Science, Feng Chia University[3]
Taichung, Taiwan, 40724
(Email: alan3c@gmail.com)
Department of Computer Science and Information Engineering, Asia University[4]
Taichung, Taiwan, 41354
Department of Computer Science and Technology, Southwest JiaoTong University[5]
Jinniu District, Chengdu 610031, China
(Received Sept. 23, 2013; revised and accepted Feb. 10 & May 10, 2014)

Abstract

Many reversible data hiding schemes have been developed in recent decades. Traditional schemes typically must deal with the problem of overflow and underflow, or transmission of hiding parameters. The new reversible data hiding scheme proposed in this paper is based on a combination of pixel groups' geometric structure and secret sharing mechanism. Experimental results confirm that the proposed scheme not only achieves the goal of hiding information without memorization of location map or any other parameter, but also generates very high quality stego-images with very large capacity of secret information embedded.

Keywords: Data hiding, geometric structure, reversible data embedding, watermarking

1 Introduction

Reversible data hiding conceals information (also called payload) into a cover image. Once an authentic user receives a stego-image containing the hidden information, the user can extract the hidden information exactly and recover the cover image without loss using a predefined procedure. Fridrich et al. [4] classified data hiding applications into two groups depending on the relationship between the hidden information and the cover image. The first group is the set of applications for which there is no relationship between the cover image and the hidden information. Both the coder and decoder are interested only in the hidden information. In the second group, the information has a close relationship to the cover image. For example, in digital watermarking, the hidden information usually is embedded into the cover image as a supplement to the cover image. In such cases, especially in medical, art, and military applications, two basic requirements must be met: (1) the receiver can extract the hidden information correctly and (2) the cover image can be recovered without distortion. The following three factors are generally used to evaluate a reversible data hiding scheme:

1) Payload capacity. Payload capacity is the maximum amount of information that can be embedded. The payload capacity is usually determined by the embedding algorithm and the size and content of the cover image. Generally, researchers use bits per pixel (bpp) to evaluate the payload capacity.

2) Imperceptibility. Imperceptibility means that people cannot perceive the existence of the hidden information with their eyes directly. For example, a meaningless image has no imperceptibility since it may readily attract a thief. In addition, if a coder adds information to the cover image directly, it is not imperceptible because of its expanded size. Thus, imperceptibility requires high similarity between the stego-image and the cover image, which means the stego-image has low distortion compared with the cover image. Generally, researchers use the peak signal to noise ratio (PSNR) to evaluate the degree of distortion of the stego-image versus the cover image.

3) Complexity of the algorithm. To indicate which pixels or pixel groups are used to embed information, many proposed schemes [5, 6, 7, 12] need a location map. Typically, these schemes use a compression technique to decrease the overhead of storing the location map. However, both the use of a location map and the compression technique increase the complexity of the algorithm.

Many data hiding schemes have been proposed in recent decades. Fridrich et al. [4] proposed the RS method for uncompressed image formats. The RS method is based on embedding messages in the status (regular or singular) of groups of pixels. However, as reported in [6], there are two drawbacks to the RS method. First, since the optimal size of a group is four pixels, the maximum bpp is only 0.25. In addition, due to the overhead of embedding the comprised RS-vector, the actual bpp is always smaller than the maximum bpp. Second, the RS method does not use information of neighboring groups so that it may lose useful information, which may increase its performance. Apart from these two drawbacks, the RS method also needs a lossless data compression technique, which increases the complexity of the algorithm. In 2003, Tian [12] proposed a difference expansion (DE) based algorithm. The DE method is based on the Haar wavelet transform, and it uses differences between two grayscale values in a pixel pair to embed information. Based on the experimental results shown in [12], Tian's algorithm can achieve very large hiding capacity. Kamstra et al. [6] developed an LSB predication embedding technique that has the same main idea described in [4]. Both the LSB predication embedding technique and the method described in [4] find a subset that is losslessly compressed from the cover image and embed the payload into the excess space after compression. Since it is difficult to construct a predication function, although [6] used another function to estimate the correctness of LSB predications, this method achieves only a very low embedding capacity. Kamstra et al. [6] also proposed an improved scheme based on Tian's method [12] in which two issues were considered: one is the capacity control problem and the other is the overhead costs caused by the location map. Due to Tian's excellent work [12], there are many variants based on the DE transform [1, 2, 3, 5, 7]. However, all these schemes must deal with the capacity problem and the overhead caused by the location map. In 2006, Ni et al. [9] proposed a reversible data hiding scheme with high PSNR (greater than 48dB) and considerable pure payload. Their algorithm is also light since there is not any transform operation such as DCT and DWT. Based on this work, Tai et al. [11] and Li et al. [8] presented improved methods that can achieve larger hiding capacity but keep embedding distortion low.

In this paper, we present a reversible data hiding method for digital images based on a combination of pixel groups' geometric structure and secret sharing mechanism [10]. The proposed method has four main merits: (1) it has meaningful stego-images, (2) the meaningful stego-images have high image quality, which means low distortion compared with the original cover image, (3) the original cover image can be recovered without loss, and (4) it can hide secrets with very large capacity without the use of a location map or any other extra information.

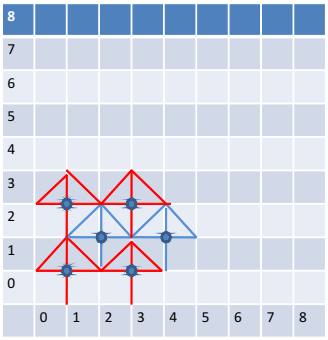

Figure 1: An example for illustrating data embedding rules

The rest of the paper is organized as follows. The proposed algorithm is presented in Section 2. Experimental results and further discussions are given in Section 3. Finally, conclusions are described in Section 4.

2 Proposed Algorithm

In this section, we provide an overview of our hiding algorithm, followed by a detailed presentation of the data embedding process and data extraction process.

2.1 Overview

Our algorithm uses a cover image to produce two stego-images (also called shadow images in secret sharing schemes) based on the to-be-embedded information and the cover image itself. We also call the to-be-embedded information a secret message in this paper. Any authentic user who holds these two shadow images at the same time can extract the secret message and recover the cover image without data loss. Next, we introduce the main idea of our method.

First, we create a two-dimensional pixel coordinate, as shown in Figure 1, and both the ordinate and abscissa of the pixel coordinate represent the grayscale value. Then we draw all the umbrellas in the pixel coordinate using a predefined method. Each umbrella has a center point called the embeddable point since just these points have the ability to embed a secret message without causing any confusion at the receiver end. We then create two copies of the cover image as the two shadows and pair all the pixels in these three images by a same pattern. Thus, each pair of the cover image can be located in the pixel coordinate as a point. In the data embedding process, if a pair of

the cover image is located on an embeddable point, we change the corresponding pair of the shadows, referring to the embedding rules to embed a secret value. The data extraction is the inverse process of the data embedding. Next, we will describe the method for creating the pixel coordinate and drawing the umbrellas.

For convenience, we use a grayscale image with each pixel consisting of 3 bits to explain the procedure of building umbrellas. In Figure 1, we have drawn four red umbrellas and two blue umbrellas in the pixel coordinate, and each umbrella has a center point. These center points are (1, 1), (3, 1), (2, 2), (4, 2), (1, 3), and (3, 3). In total, there are 32 umbrellas in the pixel coordinate, and all of them can be drawn using the following steps:

1) Assuming the point that is being scanned is (a, b), if (a, b) is an edge point in the pixel coordinate, it does not have all four neighboring points, so it can't be the center point of an umbrella. Therefore, in the scanning process, we can ignore all the edge points and scan row by row, from point (1, 1) to point (6, 6) in this example.

2) If (a, b) is not an edge point and not a point of any other umbrella, one can draw an umbrella by making (a, b) the center point. Otherwise, one goes to step 3 without doing anything.

3) If there is a next point, get it and perform step 2 for it. Otherwise, end the process.

In our proposed scheme, every center point is an embeddable point that can be used to embed a secret message. We can use the bellowing method to decide whether a point is an embeddable point or not. For point (i, j), each pixel consists of n bits:

1) If $i = 0$ or $j = 0$, (i, j) is not an embeddable point.

2) If $i = 2^{n-1}$ or $j = 2^{n-1}$, (i, j) is not an embeddable point.

3) If (i, j) is not an edge point and i, j are odd numbers, (i, j) is an embeddable point.

4) If (i, j) is not an edge point and i, j are even numbers, (i, j) is an embeddable point.

2.2 Data Embedding

In the data embedding process, we use all the embeddable points to embed the secret message. To improve the hiding capacity, we translate the to-be-embedded information to a seventeen-ary message, which means each digit value belongs to [0-16]. However, for convenience, we always use the hexadecimal system for computations in this paper.

An embeddable point (a, b) is a center point of an umbrella, shown in Figure 2, and there are four neighboring points (point (a-1, b), (a, b-1), (a+1, b), and (a, b+1))

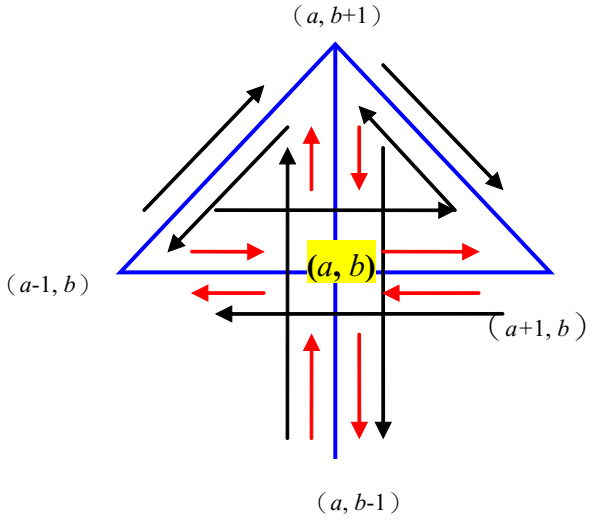

Figure 2: The proposed scheme

Table 1: Data embedding rules

Secret	EP	PPFS	PPSS
0	(a, b)	(a, b)	(a, b)
1	(a, b)	(a-1, b)	(a, b+1)
2	(a, b)	(a, b+1)	(a-1, b)
3	(a, b)	(a-1, b)	(a, b)
4	(a, b)	(a, b)	(a-1, b)
5	(a, b)	(a, b+1)	(a, b)
6	(a, b)	(a, b)	(a, b+1)
7	(a, b)	(a, b+1)	(a, b-1)
8	(a, b)	(a, b-1)	(a, b+1)
9	(a, b)	(a, b+1)	(a+1, b)
10	(a, b)	(a+1, b)	(a, b+1)
11	(a, b)	(a+1, b)	(a, b)
12	(a, b)	(a, b)	(a+1, b)
13	(a, b)	(a+1, b)	(a-1, b)
14	(a, b)	(a-1, b)	(a+1, b)
15	(a, b)	(a, b-1)	(a, b)
16	(a, b)	(a, b)	(a, b-1)

EP:Embeddable point
PFS:Pixel pair of first shadow
PSS:Pixel pair of second shadow

surrounding it. Thus, we can draw all arrows shown in Figure 2. That means there should be two different direction arrows between any two points linked by a blue line, e.g. (a-1, b) → (a, b), (a, b) → (a-1, b), (a, b+1) → (a, b-1), (a, b-1) → (a, b+1), and so on. We call the pixel pairs that determine the arrows as arrow pairs, e.g., ((a-1, b), (a, b)) is an arrow pair and ((a, b), (a-1, b)) is another arrow pair. Obviously, there are 16 arrow pairs in Figure 2 and we use these pairs to embed the secret message.

For an 8-bit grayscale cover image L and its two shadows $L1$ and $L2$, we first pair all the pixels in the cover image. For convenience, we can pair the two neighboring points row by row or column by column directly. If the size of the cover image is M×N, there are (M×N)/2 pixel pairs after pairing. Note that in this process, since the two shadows are copies of the cover image, all their pixels should be paired in the same way as the cover image. We use the following notations to represent the resultant sets of pairs for the cover image and its two shadows:

For cover image, PS={P_1, P_2,..., P_n}.

For the first shadow, PS_1={ P'_1, P'_2,..., P'_n}.

For the second shadow, PS_2={ P''_1, P''_2, ..., P''_n}.

Next, we scan PS, PS_1, and PS_2 simultaneously using the same scanning pattern, e.g., from the first element to the last sequentially. If a pixel pair in PS is an embeddable point, we embed a secret value into it by changing the corresponding pixel pairs of the two shadows in PS_1 and PS_2 according to the embedding rules. Since we have translated the binary secret message to a seventeen-ary message, we can use an embeddable point to embed a seventeen-ary value (0-16). Without losing generality, we use Table 1 to illustrate the embedding rules for the embeddable point (a, b). That means the pixel pairs of a certain arrow pair will be used to replace the corresponding original pixel pairs of these two shadows. In addition, the secret value 0 is mapped to the center point.

Now, we give an example of the data embedding process. We select the embeddable point (2, 2) in the pixel coordinate shown in Figure 1 to explain how to embed the secret message by using these umbrellas. Assuming the point (a, b) shown in Figure 2 is (2, 2), we can compute the four points surrounding (2, 2). They are (1, 2), (2, 1), (3, 2), and (2, 3). Then we use the embedding rules described in Table 1 to map a secret value to an arrow pair based on the value (0-16) of the secret itself. The embedding results are:

1) If secret = 0, ($a1$, $b1$) = (2, 2) and ($a2$, $b2$) = (2, 2);
2) If secret = 1, ($a1$, $b1$) = (1, 2) and ($a2$, $b2$) = (2, 3);
3) If secret = 2, ($a1$, $b1$) = (2, 3) and ($a2$, $b2$) = (1, 2);
4) If secret = 3, ($a1$, $b1$) = (1, 2) and ($a2$, $b2$) = (2, 2);
5) If secret = 4, ($a1$, $b1$) = (2, 2) and ($a2$, $b2$) = (1, 2);
6) If secret = 5, ($a1$, $b1$) = (2, 3) and ($a2$, $b2$) = (2, 2);
7) If secret = 6, ($a1$, $b1$) = (2, 2) and ($a2$, $b2$) = (2, 3);
8) If secret = 7, ($a1$, $b1$) = (2, 3) and ($a2$, $b2$) = (2, 1);
9) If secret = 8, ($a1$, $b1$) = (2, 1) and ($a2$, $b2$) = (2, 3);
10) If secret = 9, ($a1$, $b1$) = (2, 3) and ($a2$, $b2$) = (3, 2);
11) If secret = 10, ($a1$, $b1$) = (3, 2) and ($a2$, $b2$) = (2, 3);
12) If secret = 11, ($a1$, $b1$) = (3, 2) and ($a2$, $b2$) = (2, 2);
13) If secret = 12, ($a1$, $b1$) = (2, 2) and ($a2$, $b2$) = (3, 2);
14) If secret = 13, ($a1$, $b1$) = (3, 2) and ($a2$, $b2$) = (1, 2);
15) If secret = 14, ($a1$, $b1$) = (1, 2) and ($a2$, $b2$) = (3, 2);
16) If secret = 15, ($a1$, $b1$) = (2, 1) and ($a2$, $b2$) = (2, 2);
17) If secret = 16, ($a1$, $b1$) = (2, 2) and ($a2$, $b2$) = (2, 1).

By this way, we map a secret to a certain arrow pair based on the value of the secret itself. After we use all the embeddable points of a cover image to embed the secret messages, two shadows that have high similarity with the cover image are created. It is easy to extend this method to the grayscale image with each pixel consisting of 8 bits. That means the pixel coordinate in Figure 1 has the size of 255×255.

Table 2: Data extraction rules

($a1$,$b1$)-($a2$,$b2$)	Secret	Recovering
(0,0)	0	(($a1$,$b1$)=(($a1$,$b1$)
(-1,-1)	1	(($a1$,$b1$)=(($a1$+1,$b1$)
(1,1)	2	(($a1$,$b1$)=(($a1$,$b1$-1)
(-1,0) and ($a2$,$b2$)*	3	(($a1$,$b1$)=(($a2$,$b2$)
(1,0) and ($a1$,$b1$)*	4	(($a1$,$b1$)=(($a1$,$b1$)
(0,1) and ($a2$,$b2$)*	5	(($a1$,$b1$)=(($a2$,$b2$)
(0,-1) and ($a1$,$b1$)*	6	(($a1$,$b1$)=(($a1$,$b1$)
(0,2)	7	(($a1$,$b1$)=(($a1$,$b1$-1)
(0,-2)	8	(($a1$,$b1$)=(($a1$,$b1$+1)
(-1,1)	9	(($a1$,$b1$)=(($a1$,$b1$-1)
(1,-1)	10	(($a1$,$b1$)=(($a1$-1,$b1$)
(1,0) and ($a2$,$b2$)*	11	(($a1$,$b1$)=(($a2$,$b2$)
(-1,0) and ($a1$,$b1$)*	12	(($a1$,$b1$)=(($a1$,$b1$)
(2,0)	13	(($a1$,$b1$)=(($a1$-1,$b1$)
(-2,0)	14	(($a1$,$b1$)=(($a1$+1,$b1$)
(0,-1) and ($a2$,$b2$)*	15	(($a1$,$b1$)=(($a2$,$b2$)
(0,1) and ($a1$,$b1$)*	16	(($a1$,$b1$)=(($a1$,$b1$)

The notation (a,b)* represents that (a,b) is an embeddable point

2.3 Data Extraction

Once an authentic user receives two shadows, he/she can extract the embedded secret message and recover the cover image without loss using the following steps:

1) Pair all the pixels in these two shadows $L1$ and $L2$, respectively, according to the same pairing rule used in the data embedding process. This step produces two sets of pairs PS_1 (for $L1$) and PS_2 (for $L2$).

Figure 4: Cover image Lena and its two shadows

2) ScanPS_1 and PS_2 in the same order. To recover the cover image, we need to recover one of these two shadows. For convenience, we always use the first shadow for the recovery. We use the extraction rules shown in Table 2 to extract the secret value and perform a recovering operation for the first shadow.

After the above two steps, an authentic user can extract all the embedded information correctly and reconstruct the cover image exactly.

We use an example to clarify these steps. Assuming that the recevier is scanning the first pixel pairs in the two stego-images (denoted as $L1$ and $L2$), he gets ($a1$,$b1$) for $L1$ and ($a2$,$b2$) for $L2$. Then he performs a difference operation on these two pairs and gets the result ($a1$ - $a2$, $b1$- $b2$). Next he extracts the secret value and recovers the original image using rules shown in Table 2. For exampe, ($a1$,$b1$) = (1,2) and ($a2$,$b2$) = (2,2), the recevier gets ($a1$ - $a2$, $b1$- $b2$)=(-1, 0). From Table 2, he finds that he must know whether ($a1$,$b1$) is an embeddable point or ($a2$,$b2$) is an an embeddable point to decide the secret value is 3 or 12. Based on the discussion in Section 2.1, ($a2$,$b2$) is an an embeddable point so that the embedded secret is 3. The correspoding pixel pair of cover image is ($a2$,$b2$). Because recevier uses the first shadow image to recover the original image, he performs ($a1$,$b1$) = ($a2$,$b2$). Then, he scans next pairs in the same way as he done for the first pairs. Finally, he extracs all the secret values and the first stego image is the cover image after performing all recovery operations.

3 Experimental Results and Discussions

We used 12 standard cover images sized 512×512 pixels for the experiments. In our experiments, all pixels pairs are paired derectly row by row. Table 3 shows the experimental results and Figure 3 shows all the cover images; Figure 4 shows Lena and its shadows. From Table 3, we can see that using the method proposed in the paper, all of these cover images have very large hiding capacity but with low distortion shadows. All PSNR of shadows shown in Table 3 are greater than 51dB and the smallest payload size is also greater than 256kb. The comparison with some other schemes using Lena is shown in Table 4. In Table 4, we compare the payload size of some others methods when they achieve their maximum PSNR. We use this method to compare our result because our scheme always reach a very high PSNR so that we don't have any data to compare with other schemes with relative low PSNR. The detailed analyses of the experimental results are presented below.

Table 3: Experimental results

Images	PSNR 1(db)	PSNR 2(db)	Payload(bits)
Lena	52.05	52.07	524288
Baboon	51.48	51.50	524216
Airplane	54.82	54.81	262172
Goldhill	51.94	51.96	524288
Boat	55.12	55.12	261760
Girl	51.41	51.42	348120
Woman	55.63	55.62	262964
Crowd	55.05	55.05	300240
Lake	55.12	55.11	262192
Barbara	55.56	55.55	263312
Bridge	53.32	53.33	449368
Couple	52.62	52.63	523752

PSNR 1: PSNR for fisrt shadow
PSNR 2: PSNR for second shadow

Table 4: Comparison with other schemes of cover image Lena with size 512×512×8. In the comparison, the payload of our method is divided by two because two stego-images are used in our method.

Schemes	Payload(bits)	PSNR
Tian's scheme [12]	39,566	44.20
Ni et al.' scheme [5]	5460	48.2
Tai et al.'s scheme [11]	24,377	48.35
Li et al.'s scheme APD_1 [8]	32,995	50.82
Li et al.'s scheme APD_2 [8]	60,785	48.40
Our scheme	262,144	52.05

3.1 Imperceptibility

Since we use two shadows having high similarity with the cover image to embed the secret message, it is difficult for anyone to perceive the embedded information in these two shadows with his/her eyes directly. As shown in Table 3, the lowest PSNR is 51.41dB among all the shadows, and it is already very high compared with other methods [1-4, 10].

Consider an extreme situation where all the pixel pairs of the cover image are embeddable points. In the worst case that each grayscale value of all pixels of the two shadows will be incremented or decremented by 1, the most mean square error (MMSE) is equal to 1. Therefore, the PSNR of each shadow versus the original cover image is:

$$PSNR = 10 \times \log_{10}(\frac{255 \times 255}{2}) = 48.13db. \quad (1)$$

The low limitation of PSNR of our method is already very high, and the experimental results show that the

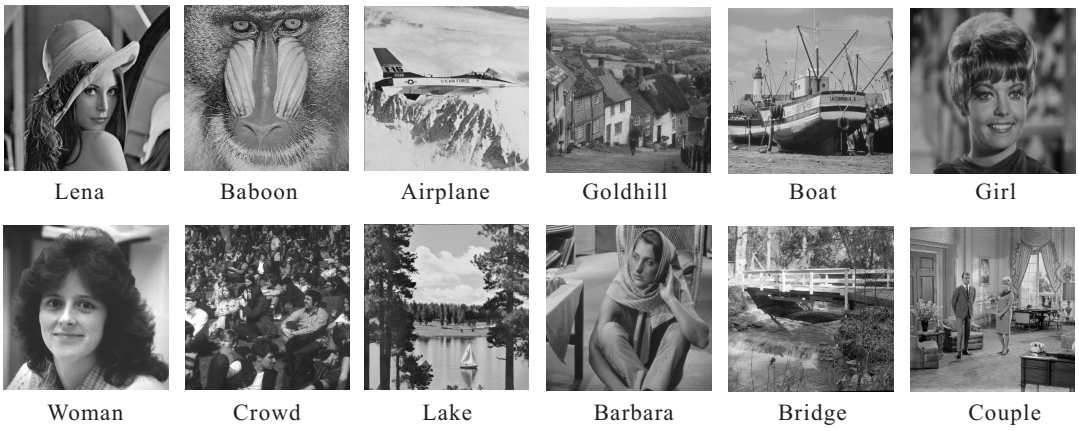

Figure 3: All cover images

actual PSNR is usually greater than the low limitation for 3dB-7dB. Thus, the two shadows produced by our algorithm have high performance in imperceptibility.

3.2 Capacity

For convenience, we think of the binary secret message as transformed to hexadecimal instead of seventeen-ary. Thus, each pixel pair can embed four bits of information. The hiding capacity for each stego-image in the above extreme situation is equal to:

$$payload = 4 \times (\frac{512 \times 512}{2 \times 2}) = 262,144 \quad \text{(bits)}. \tag{2}$$

As you can see, in an ideal situation, our algorithm can achieve very high performance with both large capacity and high PSNR. The experimental results show that all the pixel pairs of the cover images Lena and Goldhill are embeddable points so that these two images can achieve the maximum hiding capacity in our method.

3.3 Complexity

Assuming the size of the cover image is M×N, recall that in the data embedding process we use two copies (shadows) of the cover image and then pair all the pixels in these three images using the same pairing rule. Actually, we can pair the cover image, assuming the resultant pair set is $P = \{P1, P2, ..., Pn\}$, and then produce two copies $P1$ and $P2$ of the pair set P. In this way, we achieve the same purpose but the time complexity is reduced to O(MN).

For data extraction, we need to first pair the shadows and then scan the resultant pair sets simultaneously. The time complexity of pairing is O(2MN) and of scanning is also O(2MN).

Our proposed algorithm is very light since we don't use any complex transform operation (such as DWT and DCT) or any compression technique.

3.4 Capacity Control

We have assumed that all the embeddable points will be used to embed the secret value. Obviously, this is not practical and we provide a simple answer to solve this problem.

We can use header information to indicate how many embeddable points are used to embed the secret message. For a grayscale image sized M×N, the maximum number of embeddable points is (M×N)/2. If we use the first n embeddable points as the header information, for easy in calculation, we also use hexadecimal here; since each embeddable point can embed four bits, we can choose n for the inequality:

$$2^{4n} - 1 \geq (M \times N)/2. \tag{3}$$

For example, considering the grayscale image sized 512×512, the maximum number of embeddable points in this image is (512×512)/2 = 131072. Because $2^{4\times5} - 1 \geq 131072$, we select 5 as the value of n. That means if the size of the cover image is 512×512, the first five embeddable points in the cover image are always used to indicate the number of embeddable points that are used to embed information in the embedding process. Thus, for the data extraction process, the decoder needs to first extract the header information and decide how many embeddable points should be extracted.

3.5 Comparisons

Table 4 shows a simple comparison with the other schemes since these four papers present their experimental results for Lena clearly. Tian's DE method [12] can achieve high embedding capacity when the capacity is larger than 260,000 bits, but PSNR of the stego-image is lower than 30dB. In such case, the imperceptibility of the stego-image decreases. As shown Table 4, when Tian's method achieves a high PSNR, its payload suddenly drops. On the other hand, because of the requirement for a location map, it is not easy for Tians method [12] to deal with

the capacity control problem, so this method is not capable of embedding small payloads with low distortions as described in [6]. The RS method [4] cannot achieve higher performance in both capacity and imperceptibility than our method since its maximum bpp is 0.25 due to the optimal size of a group is four. The variants of the DE method, such as [5, 6, 7], also need to face up to the capacity control problem. Although these schemes propose new methods to decrease the size of the location map and control the embedding capacity, they are complex compared with our method since they usually need a lossless data compression technique. Ni et al.'s method [9] and Li et al.'s [8] can achieve very high PSNR compared with many other schemes, but both the PSNR and the hiding capacity are lower than these in our proposed method. Based on the discussions and the experimental results shown in Table 4, our method can achieve higher performance in imperceptibility, embedding capacity, and algorithm complexity than all the other methods [1, 2, 3, 4, 5, 6, 7, 9, 11, 12] mentioned in this paper.

4 Conclusions

In this paper, we proposed a reversible data hiding algorithm based on a combination of pixel groups' geometric structure and secret sharing mechanism. Since our method does not need a location map to indicate which pixel pairs are used to embed the secret message and each pixel can be changed at most one value, our algorithm has high performance in both capacity and imperceptibility. As the embeddable points are not edge points in the pixel coordinate, there would not be any overflow or underflow in our scheme. The proposed scheme is also light since we do not use any image compression technique. Our experimental results confirm that the proposed reversible data hiding algorithm outperforms all the other algorithms mentioned in the paper.

Acknowledgments

This work was supported by the National Nature Science Foundation of China under Grant No. 61272374.

References

[1] A. M. Alattar, "Reversible watermark using difference expansion of triplets," in *Proceedings of International Conference on Image Processing*, vol. 1, pp. 501–504, 2003.

[2] A. M. Alattar, "Reversible watermark using difference expansion of a generalized integer transform," *IEEE Transactions on Image Process*, vol. 13, pp. 1147–1156, 2004.

[3] A. M. Alattar, "Reversible watermark using difference expansion of quads," in *Proceedings of IEEE International Conference on Acoustics*, vol. 3, pp. 337–380, 2004.

[4] J. Fridrich, M. Goljan, and R. Du, "Lossless data embeddingnew paradigm in digital watermarking," *Journal of Applied Signal Processing*, vol. 1, pp. 185–196, 2002.

[5] Y. J. Hu, H. K. Lee, and J. W. Li, "De-based reversible data hiding with improved overflow location map," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 19, pp. 250–260, 2009.

[6] L. Kamstra and H. J. A. M. Heijmans, "Reversible data embedding into images using wavelet techniques and sorting," *IEEE Transactions on Image Process*, vol. 14, pp. 2082–2090, 2005.

[7] H. J. Kim, Y. Q. Shi, J. Nam, and H. G. Choo, "A novel difference expansion transform for reversible data embedding," *IEEE Transactions on Information Forensics and Security*, vol. 3, pp. 456–465, 2008.

[8] Y. C. Li, C. M. Yeh, and C. C. Chang, "Data hiding based on the similarity between neighboring pixels with reversibility," *Digital Signal Processing*, vol. 20, pp. 1116–1128, 2010.

[9] Z. Ni, Y. Q. Shi, N. Ansari, and W. Su, "Reversible data hiding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 16, pp. 354–362, 2006.

[10] A. Shamir, "How to share a secret," *Communication of the ACM*, vol. 11, pp. 612–613, 1979.

[11] W. L. Tai, C. M. Yeh, and C. C. Chang, "Reversible data hiding based on histogram modification of pixel differences," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 19, pp. 906–910, 2009.

[12] J. Tian, "Reversible data embedding using a difference expansion," *IEEE Transactions on Circuits System Video Technology*, vol. 13, pp. 890–896, 2012.

Zhi-Hui Wang received the BS degree in software engineering in 2004 from the North Eastern University, Shenyang, China. She received her MS degree in software engineering in 2007 and the PhD degree in software and theory of computer in 2010, both from the Dalian University of Technology, Dalian, China. Since November 2011, she has been a visiting scholar of University of Washington. Her current research interests include information hiding and image compression.

Xu Zhuang received his B.S. degree in Computer Science from Southwest Jiaotong University (SWJTU), Chengdu, China, in 2006, and is currently pursuing the Ph.D. degree in the Software Engineering Laboratory in Southwest Jiaotong University (SWJTU), Chengdu, China. His research interests include data mining, data hiding and information security.

Chin-Chen Chang received his Ph.D. degree in computer engineering from National Chiao Tung University. His first degree is Bachelor of Science in Applied

Mathematics and master degree is Master of Science in computer and decision sciences. Both were awarded in National Tsing Hua University. Dr. Chang served in National Chung Cheng University from 1989 to 2005. His title is Chair Professor in Department of Information Engineering and Computer Science, Feng Chia University, from Feb. 2005. He is a Fellow of IEEE and a Fellow of IEE, UK. His research interests include database design, computer cryptography, image compression and data structures.

Chuan Qin received the B.S. and M.S. degrees in electronic engineering from Hefei University of Technology, Anhui, China, in 2002 and 2005, respectively, and the Ph.D. degree in signal and information processing from Shanghai University, Shanghai, China, in 2008. Since 2008, he has been with the faculty of the School of Optical-Electrical and Computer Engineering, University of Shanghai for Science and Technology, where he is currently a Lecturer. He also has been with Feng Chia University at Taiwan as a Postdoctoral Researcher from July 2010 to June 2012. His research interests include image processing and multimedia security.

Yan Zhu received her B.S. and M.S. degrees in Computer Science from Southwest Jiaotong University (SWJTU), Chengdu, China, in 1986 and 1989, respectively. She received her Ph.D. degree in Computer Science from Darmstadt University of Technology, Germany in 2004. Yan Zhu is currently a professor of the School of Information Science and Technology, SWJTU and the director of the Laboratory of Software Engineering. Her research interests include data mining, Web information security, and Web spam detection.