Journal of Zhejiang University-SCIENCE C (Computers & Electronics)
ISSN 1869-1951 (Print); ISSN 1869-196X (Online)
www.zju.edu.cn/jzus; www.springerlink.com
E-mail: jzus@zju.edu.cn

Robust lossless data hiding scheme*

Xian-ting ZENG[†1,2], Xue-zeng PAN[1], Ling-di PING[1], Zhuo LI[1]

([1]*School of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China*)

([2]*College of Information Engineering, China Jiliang University, Hangzhou 310018, China*)

[†]E-mail: mico@cjlu.edu.cn

Received Mar. 29, 2009; Revision accepted May 18, 2009; Crosschecked Sept. 29, 2009

Abstract: This paper presents a robust lossless data hiding scheme. The original cover image can be recovered without any distortion after data extraction if the stego-image remains intact, and conversely, the hidden data can still be extracted correctly if the stego-image goes through JPEG compression to some extent. A cover image is divided into a number of non-overlapping blocks, and the arithmetic difference of each block is calculated. By shifting the arithmetic difference value, we can embed bits into the blocks. The shift quantity and shifting rule are fixed for all blocks, and reversibility is achieved. Furthermore, because the bit-0- and bit-1-zones are separated and the particularity of the arithmetic differences, minor changes applied to the stego-image generated by non-malicious attacks such as JPEG compression will not cause the bit-0- and bit-1-zones to overlap, and robustness is achieved. The new embedding mechanism can enhance embedding capacity and the addition of a threshold can make the algorithm more robust. Experimental results showed that, compared with previous schemes, the performance of the proposed scheme is significantly improved.

Key words: Watermarking, Lossless data hiding, Reversible data hiding, Robust lossless data hiding, Semi-fragile authentication
doi:10.1631/jzus.C0910177 **Document code:** A **CLC number:** TP309.2

1 Introduction

Data hiding is a technique that can embed data into cover media for the purposes of authentication, fingerprinting, copyright protection, security, and secret message transmission, etc. (Swanson *et al*., 1998; Hartung and Kutter, 1999; Marvel *et al*., 1999; Petitcolas *et al*., 1999; Katzenbeisser and Petitcolas, 2000; Langelaar *et al*., 2000). Very robust schemes (robust watermarking) have been developed, but they have low embedding capacity and introduce irreversible distortions. In contrast, some very high embedding capacity schemes have been proposed, but they are fragile and most of them experience some permanent distortions as a result of data hiding.

In some applications, such as medical imaging systems, law enforcement and military imagery, where the images must be in their original state for legal reasons or the images themselves are rare, it is desirable to reverse the stego-image back to the original one with no distortion. Some techniques have been published that satisfy this reversibility requirement (Honsinger *et al*., 2001; Fridrich *et al*., 2001; 2002; Tian, 2003; Alattar, 2004; Maniccam and Bourbakis, 2004; Celik *et al*., 2005; 2006; Ni *et al*., 2006; Lee *et al*., 2008; Lin and Hsueh, 2008). These are referred to as reversible, distortion-free, lossless or invertible data hiding techniques. However, they are fragile in the sense that the hidden data cannot be extracted correctly after the stego-image goes through some changes.

For some applications, however, it is desired that the hidden data will be robust against unintentional changes such as image compression and occasional unavoidable addition of random noise below a certain level, which does not change the content of an image.

* Project supported in part by the Major Science and Technology Special Project of Zhejiang Province, China (No. 2007C11088) and the Science and Technology Project of Zhejiang Province, China (No. 2008C21077)

Algorithms with this property are referred to as robust lossless data hiding algorithms (Ni *et al*., 2004). Robustness against image processing can be useful in the context of reversible data hiding; i.e., robustness permits conveying embedded information from lossless to lossy environments. It might enlarge the scope of lossless data hiding as it enables the lossless data hiding to convey information in a lossy environment. An example is the transmission of a compressed version of an image to a family doctor without losing embedded management information (Vleeschouwer *et al*., 2003).

To our knowledge, currently there are only three robust lossless data hiding schemes that protect against JPEG compression (Vleeschouwer *et al*., 2003; Zou *et al*., 2006; Ni *et al*., 2004; 2008). The scheme of Vleeschouwer *et al*. (2003) is based on the patchwork theory (Bender *et al*., 1996) and modulo-256 addition. By using a circular interpretation of bijective transformations, this scheme can achieve reversibility and robustness against high quality JPEG compression. One problem with this scheme is salt-and-pepper noise, and another drawback is that the PSNR values of the stego-images generated by this algorithm are very low. In addition, the embedding capacity of this method is very limited when a low bit error rate (BER) is maintained.

The scheme of Zou *et al*. (2006) is based on integer wavelet transform. After calculating the mean value of the HL1 or LH1 coefficients of each block, a bit 1 is embedded by shifting the mean value away from 0 by a shift quantity *S*. If a bit 0 is to be embedded, this block remains unchanged. Since the shift quantity *S* is fixed for all blocks, the original coefficients can be restored. Furthermore, the mean value of the coefficients in a block is a statistical quantity, and minor changes to the image caused by unintentional attacks such as JPEG compression will not cause the mean value to change much. Hence, this scheme is robust against high quality JPEG compression but its embedding capacity is low.

To avoid the drawbacks of the scheme of Vleeschouwer *et al*. (2003), Ni *et al*. (2004; 2008) proposed a robust lossless image data hiding scheme. This scheme achieves greater robustness and higher PSNR values of stego-images than that of Vleeschouwer *et al*. (2003). However, as a result of some error bits being introduced, error correction coding (ECC) must be applied for correction even though the stego-image remains unchanged. In addition, the embedding capacity of this scheme is also very low. Thus, a new robust lossless data hiding technique is called for that can avoid all these drawbacks.

In this study, we enhanced the scheme of Ni *et al*. (2004; 2008) by introducing two thresholds and a new embedding mechanism. The addition of thresholds can make the algorithm more robust, a new embedding mechanism can enhance capacity, and no error bits are introduced. Experimental results showed that the proposed scheme does not suffer from salt-and-pepper noise, and shows a significant improvement with respect to previous schemes in terms of embedding capacity and robustness.

2 Related studies

Ni *et al*. (2004; 2008) proposed a robust lossless data hiding scheme. In this scheme, the cover image is segmented into 8×8 image blocks. For an 8×8 image block, two subsets are split; i.e., subset *A* consists of all pixels marked by '+' and subset *B* consists of all pixels marked by '−' (Fig. 1).

+	−	+	−	+	−	+	−
−	+	−	+	−	+	−	+
+	−	+	−	+	−	+	−
−	+	−	+	−	+	−	+
+	−	+	−	+	−	+	−
−	+	−	+	−	+	−	+
+	−	+	−	+	−	+	−
−	+	−	+	−	+	−	+

Fig. 1 Difference pair pattern

A brief overview of this scheme follows:

1. Calculate the arithmetic average difference of block, denoted by α, and given by Eq. (1).

$$\alpha = \frac{1}{n}\sum_{i=1}^{n}(a_i - b_i), \tag{1}$$

where n=32, $a_i \in A$, and $b_i \in B$.

2. Classify the blocks into four different categories and use different bit-embedding schemes for each category.

3. For each category, two or three cases are

considered according to the value of α. In the data embedding process, except in cases in which the pixel grayscale values of a block are far away from the two bounds of the image histogram (0 and 255 for an 8-bit grayscale image), error bits may be introduced and ECC is then applied to correct them.

4. Select a threshold K. If α is kept within a specified threshold $-K$ and K, a bit 0 is embedded, and α is shifted by a shift quantity S beyond the threshold $-K$ or K to embed a bit 1.

Because the shift quantity S is fixed, the original arithmetic average difference can be restored. Furthermore, the arithmetic average difference in a block is a statistical quantity, and minor changes to the image caused by unintentional attacks such as JPEG compression will not cause the statistical quantity to change much, and robustness is achieved.

3 The proposed scheme

3.1 Foundation of the proposed scheme

First, an 8-bit grayscale image, denoted by C, is divided into a number of non-overlapping blocks each of size $m\times n$. Then, by introducing an $m\times n$ matrix, denoted by $\boldsymbol{M}$, we can calculate the arithmetic difference of block. The matrix $\boldsymbol{M}$ is given by

$$M(i,j)=\begin{cases}1, & \mathrm{mod2\_eq}(i,j)=1,\\ -1, & \mathrm{mod2\_eq}(i,j)=0,\end{cases} \tag{2}$$

where $i\in[1, m]$, $j\in[1, n]$, and mod2_eq(i,j) is a function which returns 1 if both i and j are odd or even and returns 0 otherwise. As an example, a matrix $\boldsymbol{M}$ with size 8×8 is shown in Fig. 2.

1	−1	1	−1	1	−1	1	−1
−1	1	−1	1	−1	1	−1	1
1	−1	1	−1	1	−1	1	−1
−1	1	−1	1	−1	1	−1	1
1	−1	1	−1	1	−1	1	−1
−1	1	−1	1	−1	1	−1	1
1	−1	1	−1	1	−1	1	−1
−1	1	−1	1	−1	1	−1	1

Fig. 2 The matrix $\boldsymbol{M}$ of size 8×8

The arithmetic difference of each block, denoted by α, is given by

$$\alpha^{(k)}=\sum_{i=1}^{m}\sum_{j=1}^{n}C^{(k)}(i,j)\times M(i,j), \tag{3}$$

where the superscript '(k)' indicates the kth block, and $C^{(k)}(i,j)$ denotes the grayscale value of the pixel at the point (i, j) of the kth block. The distribution of α is shown in Fig. 3.

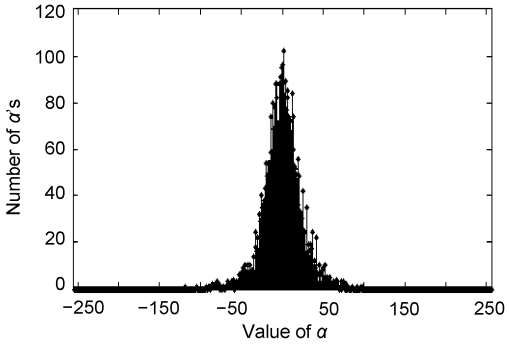

Fig. 3 The distribution of α

Next, we introduce two thresholds, denoted by T and G, respectively, both of which are positive integers. Assuming that $\alpha_{\max}$ is the largest absolute value among the values of α, we let $T=\alpha_{\max}$ (or $T\geq\alpha_{\max}$), and G is used to separate the different zones as described below.

The embedding process is as follows:

Scan each block and examine the arithmetic difference α. If a bit 0 is to be embedded, this block remains intact, and if a bit 1 is to be embedded, we can embed it into the block by shifting the arithmetic difference α. The shifting rule is given by

$$S^{(k)}(i,j)=\begin{cases}C^{(k)}(i,j)+\beta, & \\ \quad \alpha\geq 0 \text{ and } \mathrm{mod2\_eq}(i,j)=1, & \\ \quad \text{or } \alpha<0 \text{ and } \mathrm{mod2\_eq}(i,j)=0, & \\ C^{(k)}(i,j)-\beta, & \\ \quad \alpha\geq 0 \text{ and } \mathrm{mod2\_eq}(i,j)=0, & \\ \quad \text{or } \alpha<0 \text{ and } \mathrm{mod2\_eq}(i,j)=1, &\end{cases} \tag{4}$$

where $i\in[1,\ m]$, $j\in[1,\ n]$, $\beta=\lceil (T+G)/(mn)\rceil$, and the symbol $\lceil\cdot\rceil$ means 'to the nearest integer towards infinity'.

The resulting distribution of α is shown in Fig. 4. The values of α are kept within specified thresholds $-T$ and T as a result of embedding 0s, and the range of

$[-T, T]$ is called the bit-0-zone; the values of α are kept in the range of $[T+G, 2T+G]$ or $[-(2T+G), -(T+G)]$ as a result of embedding 1s, and the ranges of $[T+G, 2T+G]$ and $[-(2T+G), -(T+G)]$ are called bit-1-zones.

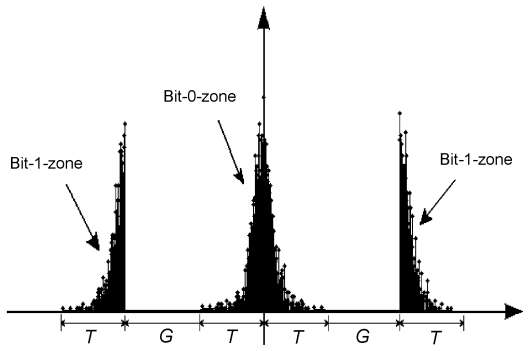

Fig. 4 Distribution of α after embedding data

Note that the pixel values of some blocks need to be added or subtracted by β when data bits are embedded into the blocks. The shift quantity β is referred to as the embedding level in this paper. In addition, we can embed a bit into each block. Hence, if the size of a grayscale image is $H\times W$ and the block size is $m\times n$, the embedding capacity of the proposed scheme is $\lfloor H/m\rfloor\times\lfloor W/n\rfloor$, where the symbol $\lfloor\cdot\rfloor$ means 'the largest integer less than or equal to'. For example, given a grayscale image of size 512×512, the embedding capacity of the proposed scheme is 8192 bits if the image is divided into 4×8 blocks.

Owing to the particularity of α, for any block, no matter whether a fixed number is added to or subtracted from the grayscale value of each pixel, the value of α of the block remains unaltered; i.e., the binary bit embedded into the block remains unaltered. Furthermore, since the threshold G is introduced and the bit-0-zones and bit-1-zones are separated by a distance G (Fig. 4), minor changes applied to the stego-image generated by non-malicious attacks such as JPEG compression will not cause the value of α to change much. In fact, as long as the bit-0-zones and bit-1-zones do not overlap, the hidden data can be extracted exactly; thus, the hidden data are robust against non-malicious attacks such as JPEG compression.

Extraction is the reverse process. Scan the blocks of the stego-image and calculate α of each block in the same sequential order as that used in the embedding phase. If $\alpha\in[-T, T]$, a bit 0 is extracted, and if $\alpha>T$ or $\alpha<-T$, a bit 1 is extracted. In addition, if the stego-image has not been altered, the cover image can be recovered by Eq. (5), where $i\in[1, m]$, $j\in[1, n]$ and $\beta=\lceil (T+G)/(mn)\rceil$.

$$R^{(k)}(i,j)=\begin{cases} S^{(k)}(i,j)-\beta, \\ \quad \alpha>T \ \text{ and } \ \mathrm{mod}2\_\mathrm{eq}(i,j)=1, \\ \quad \text{or } \alpha<-T \ \text{ and } \ \mathrm{mod}2\_\mathrm{eq}(i,j)=0, \\ S^{(k)}(i,j)+\beta, \\ \quad \alpha>T \ \text{ and } \ \mathrm{mod}2\_\mathrm{eq}(i,j)=0, \\ \quad \text{or } \alpha<-T \ \text{ and } \ \mathrm{mod}2\_\mathrm{eq}(i,j)=1, \\ S^{(k)}(i,j), \quad \text{otherwise.} \end{cases} \tag{5}$$

3.2 Prevention of overflow/underflow

The image histogram has four types (Fig. 5). For type A, no pixel has a grayscale value smaller than β or larger than $255-\beta$; i.e., all the pixel values are kept in the range of $[\beta, 255-\beta]$, and our algorithm works well in this case. For other types, if measures have not been taken, overflow/underflow may occur in the data embedding process. For example, for type B, underflow may occur, and for type C, overflow may occur.

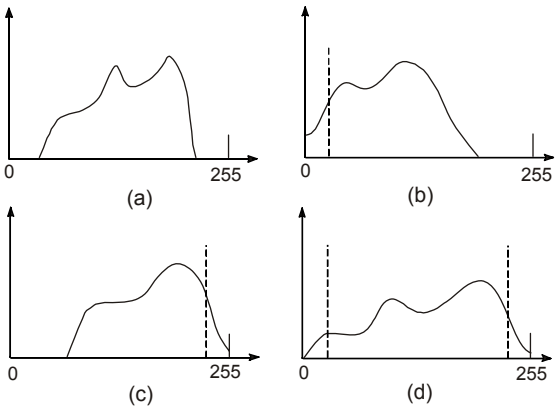

Fig. 5 Four types of image histogram
(a) Type A; (b) Type B; (c) Type C; (d) Type D

To prevent overflow/underflow, we need to keep the grayscale values of pixels in the range $[\beta, 255-\beta]$ before data embedding. Hence, we can preprocess the cover images as follows.

For an image that does not belong to type A, we can reset the pixel values of less than $\beta+1$ to $\beta+1$ and

those larger than 255−(β+1) to 255−(β+1); i.e., we can keep the grayscale values of pixels in the range [β+1, 255−(β+1)]. The original values and coordinates of the pixels, whose original values are less than β+1 or larger than 255−(β+1), are saved as overhead information in a predefined format. Next, the overhead information can be compressed losslessly and be embedded back into the cover image by employing a lossless data hiding scheme (Li *et al*., 2009). Finally, we can obtain a preprocessed cover image in which the grayscale values of pixels are kept in the range [β, 255−β], and the preprocessed image will act as a cover image to embed secret data.

In this way, our algorithm can work for any type of image. Clearly, if a preprocessed cover image is used to embed secret data, the restored image generated by Eq. (5) is not the original cover image even if the stego-image has not been altered. Hence, we need to recover the original cover image using the lossless data hiding scheme (Li *et al*., 2009). However, if the stego-image has been altered by JPEG compression, we can extract the hidden data as the cover image cannot be restored.

3.3 Data embedding

We can embed bits into a cover image (or a preprocessed cover image) using the Embed_bits function. In this function, we scan the cover image and select each block in turn in a predefined order for embedding data, assuming that the maximum payload is to be embedded into the cover image.

Algorithm 1 Embed_bits(C, B, T, G, m, n, S)

Input: C, the cover image; B, bit stream to be embedded into the cover image; T, a threshold, which is set to an integer that satisfies $T\geq\alpha_{\max}$; G, a threshold, which is used to separate the different zones, such as the bit-0-zone, bit-1-zone, etc.; $m\times n$, the block size.

Output: S, the stego-image.

Generate the matrix $\boldsymbol{M}$;
Num_of_Blk←⌊height(C)/m⌋×⌊width(C)/n⌋;
$\beta\leftarrow\left\lceil\dfrac{T+G}{m\times n}\right\rceil$; S←C; p←1;
For (k=1 to Num_of_Blk)
  Compute the value of α of the kth block;
  b←B(p); p←p+1;
  If (b=1) then
    If ($\alpha\geq$0) then
      For (i=1 to m; j=1 to n)
        If (mod2_eq(i, j)=1) then $S^{(k)}(i,j)\leftarrow S^{(k)}(i,j)+\beta$;
        If (mod2_eq(i, j)=0) then $S^{(k)}(i,j)\leftarrow S^{(k)}(i,j)-\beta$;
      Endfor
    Endif
    If (α<0) then
      For (i=1 to m; j=1 to n)
        If (mod2_eq(i, j)=1) then $S^{(k)}(i,j)\leftarrow S^{(k)}(i,j)-\beta$;
        If (mod2_eq(i, j)=0) then $S^{(k)}(i,j)\leftarrow S^{(k)}(i,j)+\beta$;
      Endfor
    Endif
  Endif
Endfor

3.4 Data extraction and original image recovery

With the unaltered stego-image, we can scan the stego-image and calculate the value of α of each block in the same predefined order as used in the embedding phase. If α lies in the bit-0-zone, i.e., [−T, T], a bit 0 is extracted, and if α lies in the bit-1-zone, a bit 1 is extracted. Extracting the hidden data and recovering the cover image are achieved using the Extract_bits function. Some parameters used in the embedding phase, such as the block size $m\times n$, the thresholds T and G, are also needed for the Extract_bits function.

Algorithm 2 Extract_bits(S, m, n, T, G, B, R)

Input: S, the stego-image; $m\times n$, the block size; T and G, the thresholds, which are used in the embedding phase.

Output: B, the hidden data (in bits); R, a restored image.

Generate the matrix $\boldsymbol{M}$;
Num_of_Blk ←⌊height(S)/m⌋×⌊width(S)/n⌋;
$\beta\leftarrow\left\lceil\dfrac{T+G}{m\times n}\right\rceil$; B←NULL; R←S; p←1;
For (k=1 to Num_of_Blk)
  Compute the value of α of the kth block;
  If $\alpha\in$[−T, T] then {B(p)←0; p←p+1;}
  If (α>T) then {B(p)←1; p←p+1;}
    For (i=1 to m; j=1 to n)
      If (mod2_eq(i, j)=1) then $R^{(k)}(i,j)\leftarrow R^{(k)}(i,j)-\beta$;
      If (mod2_eq(i, j)=0) then $R^{(k)}(i,j)\leftarrow R^{(k)}(i,j)+\beta$;
    Endfor
  Endif
  If (α<−T) then {B(p)←1; p←p+1;}
    For (i=1 to m; j=1 to n)
      If (mod2_eq(i, j)=1) then $R^{(k)}(i,j)\leftarrow R^{(k)}(i,j)+\beta$;
      If (mod2_eq(i, j)=0) then $R^{(k)}(i,j)\leftarrow R^{(k)}(i,j)-\beta$;
    Endfor
  Endif
Endfor

Obviously, when the stego-image has not been altered, the hidden data can be extracted correctly and the cover image can also be recovered without any

distortion by calling the Extract_bits function. Note that if a preprocessed cover image is used to embed data, the Extract_bits function returns the preprocessed cover image, and the original cover image can be recovered using the same lossless data hiding scheme as used in the preprocessing process.

In a lossless environment, the stego-image usually remains unaltered. However, in a lossy environment, the stego-image is difficult to retain unchanged. To verify whether the stego-image has been altered, the hash value of a cover image (or a preprocessed cover image) is embedded into the cover image as a part of the hidden data. After the hidden data are extracted and a restored cover image is obtained by calling the Extract_bits function, we can obtain a new hash value of the restored cover image. As any change to the stego-image will result in a change to the hash value, the original hash value that is included in the extracted data can be compared with the new hash value to verify whether the stego-image has been altered depending on whether they are identical or not.

3.5 Data extraction for compressed stego-images

If the stego-image has been altered, the original cover image cannot be recovered exactly, so we focus on the hidden data extraction.

The distribution of α will change as a result of JPEG compression. One such example is shown in Fig. 6, where parts of the bit-0- and bit-1-zones are overlapping. When the stego-image has gone through JPEG compression to some extent, many α's change their locations and step into the wrong zone, leading to a difficult recognition of the right zone before compression, and challenging the robustness. Hence, to extract the hidden data correctly, a certain adjustment is necessary.

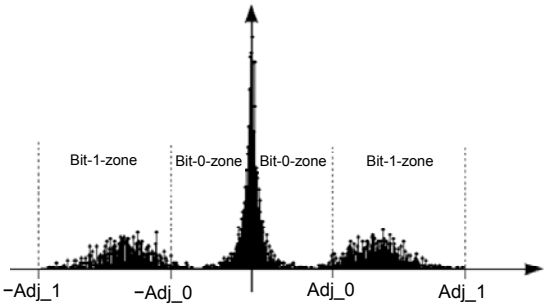

Fig. 6 Distribution of α of a stego-image that has gone through JPEG compression to some extent

In this case, after obtaining the distribution of α of the compressed stego-image, we can determine the new bit-0- and bit-1-zones by using the numbers of 0s and 1s in the hidden data. Hence, the numbers of 0s and 1s, which are denoted by N_0 and N_1, respectively, are also needed in this case.

As shown in Fig. 6, we can obtain an Adj_0 such that the number of α's in the range of [−Adj_0, Adj_0] is equal to N_0, and can obtain an Adj_1 such that the number of α's in the range of [−Adj_1, Adj_1] is equal to (N_0+N_1). In other words, we can determine Adj_0 and Adj_1 by using N_0 and N_1. With Adj_0 and Adj_1, we can obtain new thresholds T and G by

$$T = \mathrm{Adj\_0},\ G = \mathrm{Adj\_1} - 2\mathrm{Adj\_0}. \tag{6}$$

Next, we can call the Extract_bits function described above with new thresholds T and G generated by Eq. (6), and the hidden data can be extracted correctly even if the stego-image has gone through JPEG compression to some extent.

4 Experimental results

Six commonly used grayscale images (Fig. 7), each 512×512, were used to evaluate the performance of the proposed scheme. The secret data used in our experiments were generated by a pseudo-random number generator. In robustness testing, all stego-images were compressed by JPEG2000. Robustness against JPEG compression was measured by three parameters: the JPEG compression quality factor, the surviving bit rate (bpp) and the bit error rate (BER). The first two parameters were used to control image quality during lossy compression, and the last denoted the percentage of bits that had errors relative to the total bits of the hidden data. In general, the lower was the surviving bit rate (bpp) (or the lower was the JPEG compression quality factor) and the lower was the BER, the better was the robustness. The measurement of image quality used in the experiments was the peak signal-to-noise ratio (PSNR). Note that all the data shown below are the average of test results for 100 runs on the test images.

As the pixel values of an image are altered by β as a result of data bits embedding, the embedding level (i.e., β) will influence the image visual quality

directly. Assuming that the maximum payload is embedded into the cover image, the relationship between the value of PSNR and the embedding level is shown in Fig. 8. The smaller the value of β, the larger the value of PSNR. However, the values of $\alpha_{\max}$ of some images were sometimes large, i.e., the threshold T was large, so the value of β could not be small. As an example, for the image Baboon, its $\alpha_{\max}$ was 393 when the block size was 8×8, which means $\beta=\left\lceil\frac{T+G}{m\times n}\right\rceil=\left\lceil\frac{393+G}{8\times 8}\right\rceil>6$ as the threshold $G\geq 0$.

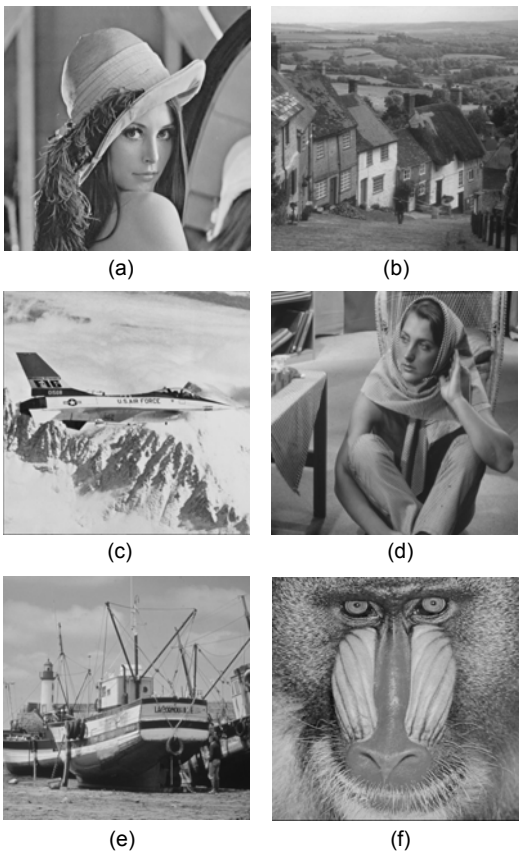

Fig. 7 Test images: (a) Lena; (b) GoldHill; (c) Airplane; (d) Barbara; (e) Boat; (f) Baboon

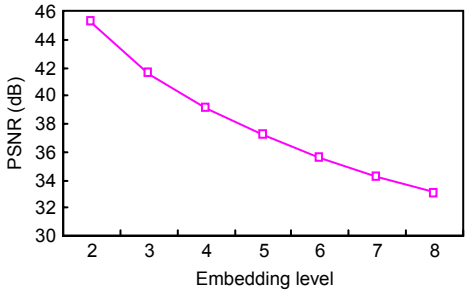

Fig. 8 Relationship between embedding level and PSNR

Considering that the value of PSNR will be about 33 dB when the embedding level is 8, we let the maximum embedding level be 8 in our experiments. Hence, to prevent overflow/underflow, the pixel values of the test images must be in the range of [8, 247] before data embedding. This meant that two images (Boat and Baboon) among the six test images needed to be preprocessed before data embedding. As mentioned above, we could first keep the pixel values of the images Boat and Baboon in the range of [9, 246], and the original values and coordinates of the pixels, whose original values were less than 9 or larger than 246, were saved as overhead information in the predefined format. Next, the overhead information could be compressed losslessly and can be embedded back into the cover image by employing the lossless data hiding scheme (Li *et al*., 2009). Finally, we could obtain the preprocessed cover images in which the grayscale values of pixels were kept in the range of [8, 247], and the preprocessed images could act as cover images to embed secret data.

To test the robustness of the proposed scheme, we first divided all test images into blocks of size 8×8, and let $T=\alpha_{\max}$ and $\beta=4$ except for the image Baboon where $\beta=7$. This meant that a message of length 4096 bits could be embedded into every test image. After embedding data, we compressed all the stego-images with various compression levels. Finally, we extracted the hidden data from the compressed stego-images, and the results that satisfy BER<1% are shown in Table 1.

Table 1 Performance of the proposed scheme

Image	PSNR (dB)	Payload (bits)	T	G	EL	Rbn[a] (bpp)	BER[b] (%)
Lena	39.11	4096	118	138	4	0.81	0.893
GoldHill	39.11	4096	109	147	4	1.22	0.657
Airplane	39.11	4096	137	119	4	0.91	0.729
Barbara	39.12	4096	155	101	4	1.26	0.687
Boat	39.10	4096	101	155	4	1.05	0.563
Baboon	34.14	4096	393	55	7	1.65	0.839

EL=Embedding level, Rbn=Robustness. [a] Average robustness=1.15 bpp, i.e., the average compression ratio is 8/1.15=6.96; [b] average BER=0.73%. T was set to $\alpha_{\max}$ and the threshold G was set to an integer that gave an embedding level of 4, except for the image Baboon where the embedding level was 7

The robustness (in bpp) in Table 1 is the lowest bit rate at which the test images could resist JPEG

compression at a given BER. The average JPEG compression ratio was 6.96 (Table 1), which means that the compressed stego-images were on average 14.38% of the size of the original images while more than 99.2% of the hidden data could be extracted correctly. This implies that our scheme is very robust.

As the threshold G is used to separate the bit-0- and bit-1-zone, as threshold G increases, the robustness against JPEG compression will strengthen. The experimental results (Fig. 9) support this observation, where the threshold T was set to an integer that satisfied $T>\alpha_{\max}$. The JPEG compression quality factor was the lowest compression quality factor at which the test images could resist the JPEG compression when the BER was less than 1%. Fig. 9 shows the relationship between the JPEG compression level and the threshold G. It can be observed that as the threshold G increased, the JPEG compression quality factor decreased. This implies that to obtain higher robustness, we can choose a larger G.

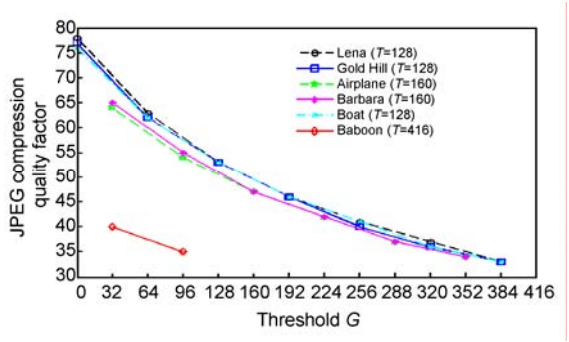

Fig. 9 Relationship between the JPEG compression quality factor and the threshold G with a block size of 8×8 for a BER of <1%

To illustrate the relationship between the JPEG compression quality factor and the embedding level in a more direct way, we conducted a set of experiments on all the test images. In this set of experiments, we also used 8×8 block size and different embedding levels to observe the robustness against image compression. Clearly, with an increase in the embedding level, the robustness against JPEG compression strengthens (Fig. 10). For instance, when the embedding level was 7, the lowest compression quality factor at which the test images could resist the JPEG compression for a BER of <1% was less than or equal to 40, and the value of PSNR was larger than 34 dB (Fig. 8). This means that the proposed scheme can offer high robustness and good image quality.

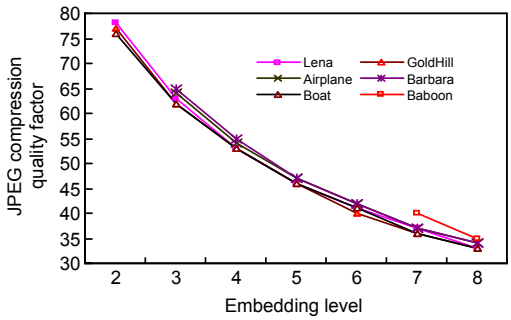

Fig. 10 Relationship between the JPEG compression quality factor and the embedding level with a block size of 8×8 for a BER of <1%

Table 2 shows the performance of the proposed scheme with different block sizes on image Lena. For any block size, the threshold T was set to its corresponding $\alpha_{\max}$, and the threshold G was set to an integer that gave an embedding level of 8, i.e., β=8. When the block size was 4×4, the proposed scheme had the maximum payload (16 384 bits) but the stego-image had the lowest robustness against JPEG compression when maintaining a BER of <1%. The opposite was true when the block size was 8×8. This implies that a block size of 8×8 is a good candidate to be used in a lossy environment. In addition, if we want to embed more bits into the cover image while maintaining strong robustness, a block size of 4×8 (or 8×4) is also a good candidate.

Table 2 Performance of the proposed scheme with different block sizes on image Lena

Block size	PSNR (dB)	Payload (bits)	T	G	EL	Rbn (bpp)	BER (%)
4×4	33.08	16 384	120	8	8	0.74	0.832
6×6	33.12	7225	100	188	8	0.53	0.490
4×8	33.07	8192	96	160	8	0.53	0.699
8×4	33.08	8192	128	128	8	0.52	0.558
8×8	33.07	4096	118	394	8	0.49	0.482

EL=Embedding level, Rbn=Robustness

Finally, to compare the performance between Ni *et al*. (2008)'s algorithm and our proposed algorithm in a more direct way, we conducted a set of experiments on three images: Lena, Boat and Baboon. In this set of experiments, we used a block size of 8×8. A comparison of the results for these images is shown in Table 3. Although Ni *et al*. (2008)'s scheme is capable

of providing higher PSNR values, their payload is quite limited. The embedding capacity of our proposed scheme is about five times higher than that of Ni *et al*. (2008)'s algorithm, in robustness against JPEG compression our scheme outperforms theirs, and the PSNR values of larger than 37.1 dB showed that the visual quality of the stego-image from our scheme is good.

Table 3 Performance comparison between the algorithm of Ni *et al*. (2008)'s and our proposed scheme on three commonly used images*

Scheme	Image	PSNR (dB)	Capacity (bits)	Robustness (bpp)	BER (%)
Ni *et al.* (2008)	Lena	40.2	792	0.8	0
	Boat	40.5	560	1.0	0
	Baboon	38.7	585	1.6	0
Proposed	Lena	37.16	4096	0.70	0.477
	Boat	37.16	4096	0.92	0.273
	Baboon	37.21	2000	1.60	0.396

* Image size 512×512

5 Conclusion

A robust lossless data hiding scheme is proposed in this paper. The exact original cover image can be restored after data extraction if the stego-image has not been altered. The hidden data are robust against non-malicious attacks such as JPEG compression to some extent. In the proposed scheme, no error bits are introduced and the addition of a threshold made the algorithm more robust and a new embedding mechanism enhanced the embedding capacity. Experimental results showed that the proposed scheme does not suffer from salt-and-pepper noise, and gives a significant improvement with respect to previous schemes in terms of embedding capacity and robustness.

References

Alattar, A.M., 2004. Reversible watermark using the difference expansion of a generalized integer transform. *IEEE Trans. Image Process.*, **13**(8):1147-1156. [doi:10.1109/TIP.2004.828418]

Bender, W., Gruhl, D., Morimoto, N., Lu, A., 1996. Techniques for data hiding. *IBM Syst. J.*, **35**(3-4):313-336.

Celik, M.U., Sharma, G., Tekalp, A.M., Saber, E., 2005. Lossless generalized-LSB data embedding. *IEEE Trans. Image Process.*, **14**(2):253-266. [doi:10.1109/TIP.2004.840686]

Celik, M.U., Sharma, G., Tekalp, A.M., 2006. Lossless watermarking for image authentication: a new framework and an implementation. *IEEE Trans. Image Process.*, **15**(4):1042-1049. [doi:10.1109/TIP.2005.863053]

Fridrich, J., Goljan, M., Du, R., 2001. Invertible authentication. *Proc. SPIE*, **4314**:197-208. [doi:10.1117/12.435400]

Fridrich, J., Goljan, M., Du, R., 2002. Lossless data embedding: new paradigm in digital watermarking. *EURASIP J. Appl. Signal Process.*, **2002**(2):185-196. [doi:10.1155/S1110865702000537]

Hartung, F., Kutter, M., 1999. Multimedia watermarking techniques. *Proc. IEEE*, **87**(7):1079-1107. [doi:10.1109/5.771066]

Honsinger, C.W., Jones, P., Rabbani, M., Stoffel, J.C., 2001. Lossless Recovery of an Original Image Containing Embedded Data. US Patent 6 278 791.

Katzenbeisser, S., Petitcolas, A.P., 2000. Information Hiding Techniques for Steganography and Digital Watermarking. Artech House Inc., Norwood, MA, USA.

Langelaar, G.C., Setyawan, I., Lagendijk, R.L., 2000. Watermarking digital image and video data. *IEEE Signal Process. Mag.*, **17**(5):20-46. [doi:10.1109/79.879337]

Lee, C.C., Wu, H.C., Tsai, C.S., Chu, Y.P., 2008. Adaptive lossless steganographic scheme with centralized difference expansion. *Pattern Recogn.*, **41**(6):2097-2106. [doi:10.1016/j.patcog.2007.11.018]

Li, Z., Chen, X.P., Pan, X.Z., Zeng, X.T., 2009. Lossless Data Hiding Scheme Based on Adjacent Pixel Difference. Proc. Int. Conf. on Computer Engineering and Technology, **1**:588-592. [doi:10.1109/ICCET.2009.40]

Lin, C.C., Hsueh, N.L., 2008. A lossless data hiding scheme based on three-pixel block differences. *Pattern Recogn.*, **41**(4):1415-1425. [doi:10.1016/j.patcog.2007.09.005]

Maniccam, S.S., Bourbakis, N., 2004. Lossless compression and information hiding in images. *Pattern Recogn.*, **37**(3):475-486. [doi:10.1016/j.patcog.2003.08.010]

Marvel, L.M., Boncelet, C.G.Jr., Retter, C.T., 1999. Spread spectrum image steganography. *IEEE Trans. Image Process.*, **8**(8):1075-1083. [doi:10.1109/83.777088]

Ni, Z., Shi, Y.Q., Ansari, N., Su, W., Sun, Q., Lin, X., 2004. Robust Lossless Image Data Hiding. IEEE Int. Conf. on Multimedia and Expo, **3**:2199-2202.

Ni, Z., Shi, Y.Q., Ansari, N., Su, W., 2006. Reversible data hiding. *IEEE Trans. Circ. Syst. Video Technol.*, **16**(3):354-362. [doi:10.1109/TCSVT.2006.869964]

Ni, Z., Shi, Y.Q., Ansari, N., Su, W., Sun, Q., Lin, X., 2008. Robust lossless image data hiding designed for semi-fragile image authentication. *IEEE Trans. Circ. Syst. Video Technol.*, **18**(4):497-509. [doi:10.1109/TCSVT.2008.918761]

Petitcolas, F.A.P., Anderson, R.J., Kuhn, M.G., 1999. Information hiding: a survey. *Proc. IEEE*, **87**(7):1062-1078. [doi:10.1109/5.771065]

Swanson, M.D., Kobayashi, M., Tewfik, A.H., 1998. Multimedia data embedding and watermarking technologies. *Proc. IEEE*, **86**(6):1064-1087. [doi:10.1109/5.687830]

Tian, J., 2003. Reversible data embedding using a difference expansion. *IEEE Trans. Circ. Syst. Video Technol.*, **13**(8): 890-896. [doi:10.1109/TCSVT.2003.815962]

Vleeschouwer, C.D., Delaigle, J.F., Macq, B., 2003. Circular interpretation of bijective transformations in lossless watermarking for media asset management. *IEEE Trans. Multimedia*, **5**(1):97-105. [doi:10.1109/TMM.2003.809729]

Zou, D., Shi, Y.Q., Ni, Z., Su, W., 2006. A semi-fragile lossless digital watermarking scheme based on integer wavelet transform. *IEEE Trans. Circ. Syst. Video Technol.*, **16**(10):1294-1300. [doi:10.1109/TCSVT.2006.881857]