A lossless robust data hiding scheme

Xian-Ting Zeng a,b,*, Ling-Di Ping a, Xue-Zeng Pan a

a College of Computer Science and Technology, Zhejiang University, 20 Yugu Road, Hangzhou 310027, PR China
b College of Information Engineering, China Jiliang University, 258 Xueyuan Street, Hangzhou 310018, PR China

Abstract

This paper presents a lossless robust data hiding scheme. The original cover image can be recovered without any distortion after the hidden data have been extracted if the stego-image remains intact, and on the other hand, the hidden data can still be extracted correctly if the stego-image goes through JPEG compression to some extent. The proposed scheme divides a cover image into a number of non-overlapping blocks and calculates the arithmetic difference of each block. Bits are embedded into blocks by shifting the arithmetic difference values. The shift quantity and shifting rule are fixed for all blocks, and reversibility is achieved. Furthermore, owing to the separation of bit-0-zone and bit-1-zone as well as the particularity of arithmetic difference, minor alteration applying to the stego-image generated by non-malicious attacks such as JPEG compression will not cause the bit-0-zone and the bit-1-zone to overlap, and robustness is achieved. Experimental results show that, compared with previous works, the performance of the proposed scheme is significantly improved.

1. Introduction

Data hiding have been extensively studied in the past, and different approaches and algorithms have been presented in the literature. 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 also been proposed, but they are fragile and most of them experience some permanent distortions as a result of data hiding [1–6].

In some applications, such as medical image system, law enforcement, military imagery and artwork preservation, where the images must be in their original states for some legal considerations or the images themselves are rare, it is desirable to reverse the stego-image back to the original cover image without any distortion. Some techniques have been published that satisfy this reversibility requirement [7–16]. 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 applying to the stego-image, such as image compression and sometimes unavoidable addition of random noise which is below a certain level and does not change the content of an image. That is, 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. Techniques with this property are referred to as robust lossless data hiding or lossless robust data hiding. The robustness of a lossless data hiding algorithm against image processing can be useful in the context of reversible data hiding. It might enlarge the application scope of lossless data hiding technique 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 [17].

1.1. Existing methods

To our knowledge, currently there are only three lossless robust data hiding schemes [17–20]. The scheme of De Vleeschouwer et al. [17] is based on the patchwork theory [21] and modulo-256 addition. By using a circular interpretation of bijective transformations, their scheme can achieve reversibility and robustness against high quality JPEG compression.

Zou et al. proposed a semi-fragile lossless digital watermarking scheme based on integer wavelet transform [18]. 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. If a bit 0 is to be embedded, this block remains unchanged.

* Corresponding author at: College of Information Engineering, China Jiliang University, 258 Xueyuan Street, Hangzhou 310018, PR China.

E-mail addresses: mico@cjlu.edu.cn (X.-T. Zeng), ldping@cs.zju.edu.cn (L.-D. Ping), xzpan@cs.zju.edu.cn (X.-Z. Pan).

0031-3203/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
Ni et al. proposed a robust lossless image data hiding scheme [19,20]. This scheme achieves greater robustness and higher PSNR values of stego-images than the scheme of De Vleeschouwer et al. After calculating the arithmetic average difference of block, a bit 1 is embedded by shifting the arithmetic average difference value away from 0 by a shift quantity. If a bit 0 is to be embedded, this block remains unchanged.

1.2. Comparison of relative methods

The comparison of three existing methods is presented in Table 1. Owing to a good difference pair pattern as shown in Fig. 1 as well as a powerful ECC code (i.e., BCH), the robustness of the scheme of Ni et al. is the best among three methods. By overcoming the drawback of the bit-embedding mechanism of Ni et al.’s scheme, the scheme of Zou et al. can offer the largest payload among the three methods when a low bit error rate (BER) is maintained. In the scheme of Zou et al., however, ECC is also applied to correct the introduced error, leading to a low embedding capacity.

In this paper, we enhanced the scheme of Ni et al. by introducing two thresholds and a new embedding mechanism. The addition of a threshold and a new embedding mechanism can enhance the capacity and make the algorithm more robust.

The rest of this paper is organized as follows. Previous works are reviewed in Section 2. The proposed scheme is presented in Section 3. Experimental results are given in Section 4. In Section 5, we discuss some factors that can improve the stego-image quality and reduce the bit error rate. Finally, conclusions are drawn.

2. Related works

Ni et al. proposed a robust lossless image data hiding scheme [19,20]. In this scheme, the original cover image is divided into a number of non-overlapping blocks sized $m \times n$ each. For an $m \times n$ 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 ‘−’, each set has $m \times n/2$ pixels. Fig. 1 shows such a pattern with size $8 \times 8$.

Then, the arithmetic average difference of block, denoted by $x$, is calculated by

$$x = \frac{1}{k} \sum_{i=1}^{k} (a_i - b_i),$$

where $k$ is equal to $m \times n/2$, $a_i \in A$, $b_i \in B$, and each pixel is used only once.

In order to overcome the overflow/underflow problem, Ni et al. classify the blocks into four different categories and use different bit-embedding schemes for each category. Four categories are shown in Fig. 2, which represent the pixel value distribution of block. For each category, two or three cases are considered according to the value of $x$.

Next, they introduce a threshold $K$. If $x$ is kept within a specified threshold $-K$ and $K$, a bit 0 is embedded, and the value of $x$ is shifted by a shift quantity $b$ beyond the threshold $-K$ or $K$ to embed a bit 1. Due to the bit-embedding mechanism, the shift quantity $b$ of this scheme is usually twice of the specified threshold $K$.

Because the shift quantity $b$ is fixed, the original arithmetic average difference $x$ can be restored. Furthermore, the arithmetic average difference in a block is a statistical quantity, and minor changes to the stego-image caused by JPEG compression will not cause the statistical quantity to change much, and robustness is achieved.

In many cases, however, only ‘0’ (sometimes ‘1’) is embedded mandatorily regardless of whether the bit to be embedded is ‘0’ or ‘1’, and ECC is applied to correct the introduced error, leading to a

![Fig. 1. Difference pair pattern.](image1)

![Fig. 2. Four categories of block.](image2)

<table>
<thead>
<tr>
<th>Methods</th>
<th>De Vleeschouwer et al. [17]</th>
<th>Zou et al. [18]</th>
<th>Ni et al. [19,20]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reversible</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Robust</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>PSNR</td>
<td>Low</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td>Payload</td>
<td>Low</td>
<td>Larger than [19,20]</td>
<td>Larger than [17]</td>
</tr>
<tr>
<td>Robustness</td>
<td>Good</td>
<td>Better than [17]</td>
<td>Better than [18]</td>
</tr>
<tr>
<td>Others</td>
<td>Exist salt-and-pepper noise</td>
<td>Introduce error bits, ECC is necessary to correct them.</td>
<td>Introduce error bits, ECC is necessary to correct them. Sometimes, reversibility is challenged.</td>
</tr>
</tbody>
</table>

Table 1

The comparison of relative methods.
low embedding capacity. In addition, in two cases, i.e., case 1 of category 2 and case 1 of category 3, they always shift \( x \) by the quantity \( \beta \) beyond the threshold \( K \) or \(-K\) in one direction to embed a bit 1. For example, in case 1 of category 2, they always shift \( x \) by the quantity \( \beta \) towards the right-hand side beyond the threshold \( K \). As the result, when the stego-image experiences compression, many \( x \) that are shifted from a negative to a positive or visa versa leave their locations and step into the wrong range more easily, leading to a low robustness. What is more, some blocks may change from one category to another in the embedding phase, and this implies that the reversibility of this algorithm is challenged. Here is an example in which a block will be identified as case 2 of category 4, the block remains unchanged and the bit ‘0’ is extracted, leading to an error.

3. The proposed scheme

In this section, the foundation of the proposed scheme is first presented. Then, the issue of how to prevent overflow/underflow is addressed. Next, data embedding algorithm and extraction algorithm are given. Finally, data extraction for compressed stego-images is given.

3.1. Foundation of the proposed scheme

To begin with, an 8-bit grayscale image, denoted by \( C \), is divided into a number of non-overlapping blocks sized \( m \times n \) each. Then, by introducing an \( m \times n \) matrix \( M \), we can calculate the arithmetic difference of block. The matrix \( M \) is given by

\[
M(i,j) = \begin{cases} 1 & \text{mod}(i,2) = \text{mod}(j,2), \\ -1 & \text{mod}(i,2) \neq \text{mod}(j,2), \end{cases}
\]

where \( i \in [1, m], j \in [1, n] \), and \( \text{mod}(x, 2) \) is a mod-2 function. As an example, a matrix \( M \) with size \( 4 \times 8 \) is shown in Fig. 4.

The arithmetic difference of block, denoted by \( \alpha \), is given by

\[
\alpha^{(k)} = \sum_{i=1}^{m} \sum_{j=1}^{n} C^{(k)}(i,j) \times M(i,j),
\]

where the superscript \( (k) \) indicates the \( k \)th block, and \( C^{(k)}(i,j) \) denotes the pixel value of the point \((i, j)\) in the \( k \)th block. A distribution of \( \alpha \) is shown in Fig. 5, where the X axis represents the value of \( \alpha \), while the Y axis is the number of \( \alpha \).

Next, we introduce two thresholds, denoted by \( T \) and \( G \), respectively, both of which are positive integers, and explore extra space to embed data. Without loss of generality, for any positive integer \( T \), we can explore the extra space by

\[
S^{(k)}(i,j) = \begin{cases} C^{(k)}(i,j) + \beta_1, & \alpha > T \text{ and } \text{mod}(i,2) = \text{mod}(j,2), \\ C^{(k)}(i,j) + \beta_2, & \alpha < -T \text{ and } \text{mod}(i,2) \neq \text{mod}(j,2), \\ C^{(k)}(i,j), & \text{otherwise}, \end{cases}
\]

where \( i \in [1, m], j \in [1, n] \),

\[
\beta_1 = \left\lfloor \frac{(2 \times G + T) \times 2}{m \times n} \right\rfloor.
\]

and the symbol \( \lfloor \cdot \rfloor \) is the ceil function meaning “to the nearest integer towards infinity.”

The result generated by Eq. (4) is shown in Fig. 6. It can be observed that there is no \( \alpha \) in the range of \([T+G, 2T+G] \) or \([-2T+G, -(T+G)] \); in other words, we can obtain extra space efficiently.

Now, we can embed bits into the blocks by utilizing the extra space. Scan each block and examine the arithmetic difference \( \alpha \). If the value of \( \alpha \) is in the range of \([-T, T] \), one bit can be embedded into the block. The embedding process is as follows.

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...
data bits are embedded into the blocks. The shift quantity whether a fixed number is added to or subtracted from the absolute value among the values of \( b \) proposed scheme is and the block size is \( m \times C_2 \) are kept within specified thresholds \( (-T, T) \), \( T \) as a result of embedding 0s, and the range \( [-T, T] \) is called the bit-0-zone; the values of \( x \) are kept in the range of \( [T+G, 2T+G] \) or \( [-T, -T+G] \) as a result of embedding 1s, and the range \( [T+G, 2T+G] \) or \( [-2T+G, -(T+G)] \) is called the bit-1-zone. Note that the pixel values of some blocks need to be added or subtracted by \( \beta_2 \) when data bits are embedded into the blocks. The shift quantity \( \beta_2 \) is referred to as embedding level in this paper.

Fig. 7 shows that the bit-0-zone and the bit-1-zones are separated by a distance \( G \). Assuming that \( x_{\text{max}} \) is the largest absolute value among the values of \( x \), we can see that if \( T \) is less than \( x_{\text{max}} \), the blocks whose \( x \) are larger than \( T \) cannot embed bits, and the values of \( x \) of the blocks are shifted by the shifting quantity \( \beta_1 \) (\( \beta_1 > \beta_2 \)). Hence, in order to achieve higher payload and lower image degradation, we can let \( T \geq x_{\text{max}} \).

Clearly, if \( T \geq x_{\text{max}} \), and 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 \( [h/m] \times [w/n] \), where the symbol \( \lfloor \cdot \rfloor \) is the floor function meaning “the largest integer less than or equal”. For example, given a grayscale image with size 512 \( \times \) 512, the embedding capacity of the proposed scheme is 8192 B if the image is divided into \( 4 \times 8 \) blocks.

Owing to the particularity of \( x \), for any block, no matter whether a fixed number is added to or subtracted from the grayscale value of each pixel, the value of \( x \) of the block remains unaltered, i.e., the binary bit embedded into the block remains unaltered. Furthermore, since the bit-0-zone and the bit-1-zone are separated by a distance \( G \), minor alteration applied to the stego-image caused by non-malicious attacks such as JPEG compression will not cause the bit-0-zone and the bit-1-zones to overlap, i.e., the hidden data can be extracted correctly, thus, the hidden data are robust to non-malicious attacks such as JPEG compression.

Extraction is the reverse process. Scan the stego-image and calculate the value of \( x \) of each block in the same predefined order as that used in the embedding phase. If \( x \in [-T, T] \), a bit 0 is extracted, and if \( x \in [T, 2T+G] \) or \( x \in [-2T+G, -T] \), a bit 1 is extracted. In addition, if the stego-image has not been altered, the cover image can be recovered by

\[
\begin{align*}
S^{(b)}(i,j) &= \begin{cases} S_1^{(b)}(i,j) + \beta_2, & x \in [0, T] \text{ and } \text{mod}(i, 2) \neq \text{mod}(j, 2), \\
S_1^{(b)}(i,j), & x \in [-T, 0] \text{ and } \text{mod}(i, 2) \neq \text{mod}(j, 2), \\
S_1^{(b)}(i,j) + \beta_2, & \text{otherwise},
\end{cases} \\
R^{(b)}(i,j) &= \begin{cases} S_1^{(b)}(i,j) - \beta_2, & x \in (T+G, 2T+G) \text{ and } \text{mod}(i, 2) = \text{mod}(j, 2), \\
S_1^{(b)}(i,j), & x \in [-2T+G, -T) \text{ and } \text{mod}(i, 2) \neq \text{mod}(j, 2), \\
S_1^{(b)}(i,j) + \beta_1, & \text{otherwise},
\end{cases}
\end{align*}
\]

where \( \beta_1 = \left\lfloor \frac{2 \times G + T}{m \times n} \right\rfloor \) and \( \beta_2 = \left\lfloor \frac{T + G}{m \times n} \right\rfloor \).

3.2. Prevention of overflow/underflow

To embed bits into the cover image as mentioned above, we need to add a certain number, say, \( S_t \), to the grayscale values of some pixels of the cover image. For a grayscale image, the permitted range is \([0, 255]\). Therefore, if some pixel values of the cover image are larger than \((255 - S_t)\) and a number \( S_t \) needs to be added to the grayscale values of these pixels, overflow will occur.

The image histogram has four types as shown in Fig. 8(a)–(d). For type A, no pixel has a grayscale value smaller than \( S_t \) or larger than \((255 - S_t)\); for type B, no pixel has a grayscale value larger than \((255 - S_t)\); for type C, no pixel has a grayscale value smaller than \( S_t \); for type D, there are some pixels with value smaller than \( S_t \) and some larger than \((255 - S_t)\). To avoid overflow/underflow, we must take measures.

For types A and B, no pixel has a grayscale value larger than \((255 - S_t)\), overflow/underflow will not take place in these cases according to Eqs. (4) and (5).
For type C, overflow will take place according to Eqs. (4) and (5). To solve this issue, we introduce a parameter and some conversions. The parameter, denoted by $\delta$, is given by

$$
\delta = \begin{cases} 
1 & \text{type A or type B}, \\
-1 & \text{type C}, 
\end{cases}
$$

and the conversions are as follows:

1. $M(i,j)$ is replaced by $(\delta \times M(i,j))$ in Eq. (3);
2. $\beta_1$ is replaced by $(\delta \times \beta_1)$ in Eqs. (4) and (6);
3. $\beta_2$ is replaced by $(\delta \times \beta_2)$ in Eqs. (5) and (6).

For instance, according to these conversions Eq. (5) will become

$$
S^{(k)}(i,j) = \begin{cases} 
S^{(k)}(i,j) + (\delta \times \beta_2), & \alpha \in [0, T] \text{ and } \text{mod}(i, 2) = \text{mod}(j, 2), \\
S^{(k)}(i,j) + (\delta \times \beta_2), & \alpha \in [-T, 0) \text{ and } \text{mod}(i, 2) \neq \text{mod}(j, 2), \\
S^{(k)}(i,j) & \text{otherwise.}
\end{cases}
$$

In this way, when $\delta$ is equal to 1, i.e., for type A or type B, Eq. (5) is the same as Eq. (5). When $\delta$ is equal to $-1$, i.e., for type C, addition is turned into subtraction in the new forms of Eq. (5).

Fig. 8(c) shows this case. Hence, overflow/underflow will not occur whether $\delta=1$ or $\delta=-1$ in Eq. (5), which means that overflow/underflow will not occur for type A or type B or type C.

For type D, overflow/underflow will take place inevitably. In our proposed scheme, we can change this type into type B or type C by preprocessing the cover image. The steps are as follows.

If the number of pixels $(< S_0)$ is less than that of $(> (255 - S_0))$, we can reset the pixel values of less than $S_0$ to $S_0$. As the result, type D becomes type C, and we let $\delta=1$. Otherwise, we can reset the pixel values of larger than $(255 - S_0)$ to $(255 - S_0)$. In this way, type D becomes type B, and we let $\delta=-1$.

Then, the original values and the coordinates of the pixels, whose original values are less than $S_0$ (or larger than $(255 - S_0)$), are saved as overhead information in a predefined format. Next, the overhead information can be compressed losslessly, and the compressed overhead information and secret data can be concatenated and be embedded into the cover image together.

Finally, we can obtain a preprocessed cover image in which the grayscale values of pixels are kept in the range $[0, 255 - S_0]$ or $[S_0, 255]$, and the preprocessed image will act as a cover image to embed data.

In this way, our algorithm can work for any type of image.

Fortunately, there are few images of type D.

3.3. Data embedding algorithm

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. Embed_Bits(C, B, G, m, n, T, $\delta$, S)**

/* Inputs: C, the cover image; B, bit stream to be embedded into the cover image; G, a threshold, which is used to space the different zones, such as the bit-0-zone, the bit-1-zone, etc.; m \times n, the block size; T, a threshold, which is usually set to $T = \frac{S_{\text{max}}}{S} \cdot \delta$, its value is 1 or $-1$ depending on the type of the cover image */

/* Output: S, the stego-image */

Generate the matrix $M$: $M \leftarrow -\delta \times M$; Num_of_Blk $\leftarrow \text{height}(C)/m \times \text{width}(C)/n$; $S \leftarrow C$

For $k \leftarrow 1$ to Num_of_Blk

Compute the value of $x$ of the $k$th block

While $(x > T \text{ and } \text{mod}(i, 2) = \text{mod}(j, 2))$ do

$(R^{(k)}(i,j) \leftarrow R^{(k)}(i,j) - \beta_1)$

While $(x < -(2T+G) \text{ and } \text{mod}(i,2) \neq \text{mod}(j,2))$ do

$(R^{(k)}(i,j) \leftarrow R^{(k)}(i,j) - \beta_1)$

End if

If $(x = -T)$ then

$B(p) \leftarrow 1$; $p \leftarrow p + 1$

End if

If $(x = (2T+G))$ then

$B(p) \leftarrow 1$; $p \leftarrow p + 1$

End if

While $(\text{mod}(i,2) \neq \text{mod}(j,2))$ do

$(R^{(k)}(i,j) \leftarrow R^{(k)}(i,j) - \beta_2)$

End if

End for

The computational load of the proposed scheme is very light. Let $h$ and $w$ denote the height and width of an image, respectively. In the Embed_Bits function, the required processing mainly lies on calculating $x$ and shifting $x$. The computational complexity is $O(hw)$. With a computer of Intel T1350 1.86 GHz and the software Matlab R2008b, the total embedding time needed for the Lena image $512 \times 512 \times 8$ is about 80 ms.

3.4. Data extraction and original image recovery

With the unchanged stego-image, we can scan the stego-image and calculate the value of $x$ of each block in the same predefined order as that used in the embedding phase. If $x$ lies in the bit-0-zone, i.e., $[-T, T]$, a bit 0 is extracted, and if $x$ 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 which are used in the embedding phase, such as the block size $m \times n$, the thresholds $T$ and $G$, and the thresholds are also needed for the Extract_Bits function.

**Algorithm. Extract_Bits(S, m, n, T, G, $\delta$, B, R)**

/* Inputs: S, the stego-image; m \times n, the block size; T and G, the thresholds; $\delta$ */

/* Output: B, the hidden data (in bits); R, the recovered image */

Generate the matrix $M$: $M \leftarrow -\delta \times M$; B = NULL

Num_of_Blk $\leftarrow \text{height}(S)/m \times \text{width}(S)/n$; $R \leftarrow S$

For $k \leftarrow 1$ to Num_of_Blk

Compute the value of $x$ of the $k$th block

While $(x > (2T+G) \text{ and } \text{mod}(i,2) = \text{mod}(j,2))$ do

$(R^{(k)}(i,j) \leftarrow R^{(k)}(i,j) - \beta_1)$

While $(x < -(2T+G) \text{ and } \text{mod}(i,2) \neq \text{mod}(j,2))$ do

$(R^{(k)}(i,j) \leftarrow R^{(k)}(i,j) - \beta_1)$

If $(x = -T)$ then

$B(p) \leftarrow 0$; $p \leftarrow p + 1$

End if

If $(x = (2T+G))$ then

$B(p) \leftarrow 1$; $p \leftarrow p + 1$

End if

While $(\text{mod}(i,2) \neq \text{mod}(j,2))$ do

$(R^{(k)}(i,j) \leftarrow R^{(k)}(i,j) - \beta_2)$

End if

End for

// i \in [1, m], j \in [1, n], $p \leftarrow 1$
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 overhead information, which is in the extracted data, can be used to recover the original cover image.

3.5. Data extraction for compressed stego-images

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

The distribution of $x$ will change as a result of JPEG compression. One such example is shown in Fig. 9, where parts of the bit-0-zone and the bit-1-zone are overlapping. Fig. 9 shows that many $x$ 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.

In this case, after obtaining the distribution of $x$ of the compressed stego-image, we can choose the new bit-0-zone and bit-1-zone by using the numbers of 0s and 1s in the hidden data. Hence, the numbers of 0s and 1s, denoted by N0 and N1, respectively, are needed in this case.

As shown in Fig. 9, we can obtain an Adj_0 such that the number of $x$ in the range of $[-\text{Adj}_0, \text{Adj}_0]$ is equal to N0, and can obtain an Adj_1 such that the number of $x$ in the range of $[-\text{Adj}_1, \text{Adj}_1]$ is equal to (N0+N1), i.e., we can obtain Adj_0 and Adj_1 by using N0 and N1. And with Adj_0 and Adj_1, we can obtain new thresholds $T$ and $G$ by

$$\begin{align*}
T &= \text{Adj}_0, \\
G &= \text{Adj}_1 - 2\text{Adj}_0.
\end{align*}$$

Finally, we can call the Extract_Bits function described above with new thresholds $T$ and $G$ generated by Eq. (8), and the hidden data can be extracted correctly even if the stego-image has gone through JPEG compression to some extent.

4. Experimental results

In our experiments, eight commonly used grayscale images, $512 \times 512$ each, as shown in Fig. 10(a)–(h), are used to evaluate the performance of the proposed scheme. The secret data used in our experiments are generated by a pseudo-random number generator. In robustness testing, all the stego-images are compressed by JPEG2000, and the compression tool is LuraWave JP2 Command Line Tool, which is based on LuraTech’s implementation of the JPEG2000 image compression standard. The JPEG compression level used in our experiments is measured by one of the two parameters, i.e., the surviving bit rate in the unit of bpp (bits per pixel) and the JPEG compression quality factor. The robustness against JPEG compression is measured by the

![Fig. 9. The distribution of $x$ of a stego-image that has gone through JPEG compression to some extent.](image)

![Fig. 10. Test images: (a) Lena; (b) Boat; (c) Zelda; (d) GoldHill; (e) Baboon; (f) Peppers; (g) Airplane; (h) Barbara.](image)
surviving bit rate or compression quality factor at a given BER, meaning that when the resultant bit rate (or quality factor) after compression is above or equal to the bit rate (or quality factor), the hidden data can be reliably extracted, and BER (bit error rate) denotes the percentage of bits that have errors relative to the total bits of the hidden data. In general, given a BER, the lower the surviving bit rate (bpp) (or the lower the compression quality factor) is, the better the robustness will be. On the other hand, given a compression level, the lower the BER is, the better the robustness will be. The measurement of image quality used in the experiments is the peak signal to noise ratio (PSNR). The PSNR is defined as

$$\text{PSNR} = 10 \times \log_{10} \left( \frac{255 \times 255 \times w \times h}{\sum_{i=1}^{m} \sum_{j=1}^{n} (C(i,j) - S(i,j))^2} \right) \text{dB},$$

where \(C(i,j)\) and \(S(i,j)\) denote the pixel values in row \(i\) and column \(j\) of the original cover image and the stego-image, respectively, and \(w\) and \(h\) stand for the width and height of the cover image, respectively. 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 \(\beta_2\), as a result of data bits embedding, the embedding level (i.e., \(\beta_2\)) will influence the image visual quality directly. Assuming that the maximum payload is embedded into the cover image and the threshold \(T\) is set to an integer that satisfies \(T \geq \alpha_{max}\), the relationship between the value of PSNR and the embedding level is shown in Fig. 11. An observation can be made that the larger is the value of \(\beta_2\), the lower is the value of PSNR. Considering that the value of PSNR will still be larger than 30 dB when the embedding level is 15, we let the maximum embedding level be 15 in our experiments. Hence, to prevent overflow/underflow, we set \(S_0=15\) in our experiments. That is, images of type D mean that there are some pixels with value \(<15\) and some larger than \(240\). According to this assumption, all the test images except image ‘Peppers’ are not type D, so it is not necessary to preprocess them. For image ‘Peppers’, as the number of pixels in the range of \([241,255]\) is less than that in the range of \([0,14]\), we can keep the pixel values in the range of \([0,240]\) by preprocessing it, i.e., we can reset the pixel values of larger than \(240\) to \(240\) and save their coordinates and the original grayscale values as overhead information. The overhead information of size 166 byte is generated as a result of preprocessing, and type of image ‘Peppers’ becomes type B.

To test the robustness of the proposed scheme, we first divide the test images into blocks of size \(8 \times 8\) and let \(T \geq \alpha_{max}\). For images ‘Lena’, ‘Boat’, ‘Zelda’ and ‘GoldHill’, their \(\alpha_{max}\) are close to and less than 120, and considering that 128 is larger than 120 and is a multiple of \(8 \times 8\), we let \(T=128\). Also, for image ‘Baboon’, its \(\alpha_{max}\) is 388, we let \(T=416\); for image ‘Peppers’, its \(\alpha_{max}\) is 211, we let \(T=224\); for images ‘Airplane’ and ‘Barbara’, their \(\alpha_{max}\) are close to and \(<160\), we let \(T=160\). As a general acceptance, a marked image whose PSNR value is over \(38\) dB is acceptable for the imperceptibility requirement [4], so the embedding level (i.e., \(\beta_2\)) should be \(<7\) as shown in Fig. 11. Hence, we let \(\beta_2=6\). In this way, for images ‘Lena’, ‘Boat’, ‘Zelda’ and ‘GoldHill’, their \(G\) should be set to 64; for images ‘Airplane’ and ‘Barbara’, their \(G\) should be set to 32. For images ‘Baboon’ and ‘Peppers’, however, because their \(T\) are larger than or equal to 224, which means

$$\beta_2 = \frac{(T+G) \times 2}{m \times n} \geq \frac{224 \times 2 + G \times 2}{8 \times 8} > 6$$

even though their \(G\) are set to 0, so their \(G\) are set to 0. In this set of experiments, we embed the maximum payload into the test images, and this means that a message of length 4096 B can be embedded into every test image. After embedding data, we compress all the stego-images with increasing compression ratios, where the surviving bit rate (bpp) is selected as the compression level. Finally, we extract the hidden data from the compressed stego-images, and the results that satisfy BER \(<1\%\) are presented in Table 2. The robustness (bpp) in Table 2 is the lowest surviving bit rate at which the test images can resist JPEG compression at a given BER.

Table 2 shows that the test images can resist the JPEG compression with the surviving bit rate ranging from 1.72 to 0.56 bpp (bits per pixel) when BER \(<1\%\), and pure payload of each image except image ‘Peppers’ can be up to 4096 B. Note that the PSNR values of images ‘Baboon’ and ‘Peppers’ are \(<38\) dB as their embedding levels are not \(<7\). An observation can be made from Table 2 that the same is the embedding level \(\beta_2\), the value of PSNR is almost the same.

Next, we select the compression quality factor as the compression level to compress the stego-images and evaluate the robustness of the proposed scheme. Fig. 12 illustrates the results of a given \(T, G\) and block size. It can be observed that when JPEG compression quality factor is \(\geq 65\), more than 99% of the hidden data can be extracted correctly for all the test images, and when JPEG compression quality factor is greater than or equal to \(80\), the hidden data can be extracted with no error. This implies that our scheme is very robust. Because image ‘baboon’ has a higher embedding level, the lowest compression quality factor at

---

**Table 2**

The performance of the proposed scheme with block size \(8 \times 8\) for BER \(<1\%\).

<table>
<thead>
<tr>
<th>Images</th>
<th>PSNR (dB)</th>
<th>Payload (B)</th>
<th>Pure payload (B)</th>
<th>(T)</th>
<th>(G)</th>
<th>Embedding level (\beta_2)</th>
<th>Robustness (bpp)</th>
<th>Bit error rate (BER) (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lena</td>
<td>38.50</td>
<td>4096</td>
<td>4096</td>
<td>128</td>
<td>64</td>
<td>6</td>
<td>1.04</td>
<td>0.69</td>
</tr>
<tr>
<td>Boat</td>
<td>38.59</td>
<td>4096</td>
<td>4096</td>
<td>128</td>
<td>64</td>
<td>6</td>
<td>1.56</td>
<td>0.77</td>
</tr>
<tr>
<td>Zelda</td>
<td>38.58</td>
<td>4096</td>
<td>4096</td>
<td>128</td>
<td>64</td>
<td>6</td>
<td>0.56</td>
<td>0.60</td>
</tr>
<tr>
<td>GoldHill</td>
<td>38.58</td>
<td>4096</td>
<td>4096</td>
<td>128</td>
<td>64</td>
<td>6</td>
<td>1.72</td>
<td>0.90</td>
</tr>
<tr>
<td>Baboon</td>
<td>31.87</td>
<td>4096</td>
<td>4096</td>
<td>160</td>
<td>32</td>
<td>6</td>
<td>1.05</td>
<td>0.80</td>
</tr>
<tr>
<td>Airplane</td>
<td>38.60</td>
<td>4096</td>
<td>4096</td>
<td>160</td>
<td>32</td>
<td>6</td>
<td>1.72</td>
<td>0.89</td>
</tr>
</tbody>
</table>

* Average BER = 0.79%.

* Pure payload = 4096 – 166 × 8 = 2768 B.
which it can resist JPEG compression at a given BER is much lower than those of other images. It can also be observed that when the compression quality factor is less than a certain value, the robustness will weaken quickly.

Since the threshold $G$ is used to separate the bit-0-zone and the bit-1-zone, so the robustness against JPEG compression will strengthen as the threshold $G$ increases. The experimental results (Figs. 13 and 14) support this observation, where the threshold $T$ is set to an integer that satisfies $T \geq \alpha_{\text{max}}$, an $8 \times 8$ block size is used in this set of experiments and a message of length 4096 B is embedded into the cover image. The surviving bit rate (bpp) or the compression quality factor in Figs. 13 and 14 is the lowest value at which the test images can resist the JPEG compression when the BER is $< 1\%$. Figs. 13 and 14 show that when $T \geq \alpha_{\text{max}}$, as the threshold G increases, the robustness against JPEG compression will strengthen. This implies that to obtain higher robustness we can choose a larger $G$.

To illustrate how the block size influences the performance of the proposed scheme, we conduct a set of experiments on image ‘Lena’. In this set of experiments, the threshold $T$ is set to an integer that satisfies $T \geq \alpha_{\text{max}}$ and the threshold $G$ is set to an integer that gives an embedding level of 8 for each block size, except for block size 4 where the embedding level is 15. We embed the maximum payload into the cover image ‘Lena’, and compress the stego-image to some extent. We extract the hidden data from the compressed stego-image, and the results are presented in Table 3. The robustness (bpp) in Table 3 is the lowest surviving bit rate at which the test image can resist JPEG compression at a given BER. We can observe from Table 3 that the smaller the block size is, the larger the payload can be. This implies that the proposed scheme can offer higher payload by selecting a smaller block size. However, the value of $\alpha_{\text{max}}$ of a small block is usually large, which means that the threshold $T$ must be a large integer because $T$ is usually set to an integer that satisfies $T \geq \alpha_{\text{max}}$. For example, for block size 4, $\alpha_{\text{max}}$ is 120, which means $\beta_2 = \frac{\left(T + G\right) \times 2}{m \times n} \geq \frac{120 \times 2 + G \times 2}{4 \times 4} \geq 15$.

Hence, although our proposed scheme has the maximum payload (16384 B) when block size is 4, the corresponding stego-image

---

**Fig. 12.** Robustness illustration with block size $8 \times 8$. The graph presents the BER as a function of the JPEG compression level.

**Fig. 13.** The relationship between the surviving bit rate (bpp) and the threshold $G$ with block size $8 \times 8$ for a BER of $< 1\%$. 

---
Fig. 14. The relationship between the JPEG compression quality factor and the threshold G with block size 8 × 8 for a BER of < 1%.

Fig. 15. The relationship between the robustness against JPEG compression and the embedding level with block size 8 × 8 for a BER of < 1%: (a) the robustness in terms of surviving bit rate (bpp); and (b) the robustness in terms of compression quality factor.

Table 3
The performance of the proposed scheme with different block size on image 'Lena' for BER < 1%.

<table>
<thead>
<tr>
<th>Block size</th>
<th>Pure payload (B)</th>
<th>PSNR (dB)</th>
<th>T</th>
<th>G</th>
<th>Embedding level β</th>
<th>Robustness (bpp)</th>
<th>Bit error rate (BER) (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 × 4</td>
<td>16384</td>
<td>30.63</td>
<td>120</td>
<td>0</td>
<td>15</td>
<td>0.74</td>
<td>0.89</td>
</tr>
<tr>
<td>6 × 6</td>
<td>7225</td>
<td>36.12</td>
<td>108</td>
<td>36</td>
<td>8</td>
<td>0.84</td>
<td>0.71</td>
</tr>
<tr>
<td>4 × 8</td>
<td>8192</td>
<td>36.09</td>
<td>128</td>
<td>0</td>
<td>8</td>
<td>0.88</td>
<td>0.81</td>
</tr>
<tr>
<td>8 × 4</td>
<td>8192</td>
<td>36.11</td>
<td>128</td>
<td>0</td>
<td>8</td>
<td>0.88</td>
<td>0.83</td>
</tr>
<tr>
<td>8 × 8</td>
<td>4096</td>
<td>36.08</td>
<td>128</td>
<td>128</td>
<td>8</td>
<td>0.80</td>
<td>0.79</td>
</tr>
<tr>
<td>12 × 12</td>
<td>1764</td>
<td>36.22</td>
<td>216</td>
<td>360</td>
<td>8</td>
<td>0.79</td>
<td>0.77</td>
</tr>
<tr>
<td>16 × 16</td>
<td>1024</td>
<td>36.09</td>
<td>256</td>
<td>768</td>
<td>8</td>
<td>0.77</td>
<td>0.51</td>
</tr>
</tbody>
</table>
has the worst visual quality. In general, block size $8 \times 8$ 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 $4 \times 8$ (or $8 \times 4$) is also a good candidate.

When $T < z_{max}$ as mentioned above, some blocks cannot embed bits, so payload will not achieve the maximum embedding capacity. In addition, the values of $z$, which are larger than $T$, will be shifted by a shifting quantity $\beta_1$. This implies that the PSNR value will decrease more quickly because $\beta_1$ is larger than $\beta_2$. We conduct a set of experiments to observe how the threshold $T$ influences the robustness of the proposed scheme, and the results are illustrated in Figs. 16–18. In this set of experiments, we also use $8 \times 8$ block size.

Fig. 16 shows the robustness comparison between $T < z_{max}$ and $T \geq z_{max}$ on image ‘Lena’. Clearly, although the embedding levels of two cases are the same, i.e., $\beta_2=6$, the BER are not the same. The same thing happens at the situations of images ‘Barbara’ and ‘Baboon’, as shown in Figs. 17 and 18, respectively. An observation can be made from Figs. 16–18 that the case of $T < z_{max}$ will weaken the robustness drastically. This implies that $T < z_{max}$ is not a good candidate to be used in the data embedding process. Therefore, $T$ is usually set to an integer that satisfies $T \geq z_{max}$.

Finally, in order to compare the performance between Ni et al.’s algorithm [19,20] and the proposed algorithm in a more direct way, we conduct a set of experiments on all the test images, and comparison results on the test images are presented in Table 4. Although Ni et al.’s scheme is capable of providing higher
ARTICLE IN PRESS

As mentioned above, the overall performance including payload, image quality and robustness will degrade when $T < \alpha_{\text{max}}$. Therefore, $T$ is usually set to an integer that satisfies $T \geq \alpha_{\text{max}}$. Furthermore, to aim at better robustness, we can choose a larger $G$. However, sometimes $\alpha_{\text{max}}$ is very large, e.g., for image ‘Baboon’, the value of $\alpha_{\text{max}}$ is 388, leading to a very large $T$. In this way, the value of PSNR is not high enough even if $G$ is set to zero. Hence, if the value of $\alpha_{\text{max}}$ can be reduced, the value of the threshold $T$ satisfying $T \geq \alpha_{\text{max}}$ can be reduced accordingly, and the threshold $G$ can be set to an integer in a larger range. This means that a better balance between the hidden data robustness and the image quality can be achieved.

Fortunately, in most cases modifying only one pixel value can reduce the value of $\alpha$ of a block. Fig. 19 shows such an example, where the original value of $\alpha$ is $-388$. If the original pixel value of 181 (the shaded area shown in Fig. 19(a)) is subtracted by 100, the new value of $\alpha$ will be $-288$, as shown in Fig. 19(b).

For example, the four largest absolute values of $\alpha$ of image ‘Baboon’ are 388, 384, 365 and 313, respectively. If we modify

$$\begin{array}{cccc}
\text{Images} & \text{Ni et al. [19,20]} & \text{Proposed} \\
\hline
\text{Lena} & 40.19 & 792 & 0.80 & 38.07 & 2048 & 0.80 & 0.015 \\
\text{Boat} & 40.47 & 560 & 1.00 & 38.09 & 2048 & 1.00 & 0.182 \\
\text{Zelda} & 40.47 & 560 & 0.47 & 38.18 & 2532 & 0.47 & 0.082 \\
\text{Goldhill} & 40.38 & 792 & 1.08 & 38.10 & 2048 & 1.08 & 0.157 \\
\text{Baboon} & 38.68 & 585 & 1.60 & 38.05 & 850 & 1.60 & 0.243 \\
\text{Peppers} & 40.48 & 560 & 0.61 & 38.12 & 2568^a & 0.61 & 0.047 \\
\text{Airplane} & 40.17 & 792 & 0.80 & 38.09 & 2048 & 0.80 & 0.013 \\
\text{Barbara} & 40.19 & 792 & 1.16 & 38.07 & 2048 & 1.16 & 0.663 \\
\end{array}$$

* Pure payload=2568 – 166 × 8 = 1240 B.
these values to a new value 313 by altering the pixel values of corresponding blocks, the new $x_{\text{max}}$ of image ‘Baboon’, denoted by $x_{\text{max}}$, will become 313. Therefore, we can let $T=320$ ($T \geq x_{\text{max}}$) instead of 416. In this way, the threshold $G$ can be set to 0, 32, 64 or 96. If we want to offer a higher image quality, we can let $G=0$, and if we want to offer a better robustness, we can let $G=96$. Fig. 20(b) shows new robustness on image ‘Baboon’. Clearly, compared with the original BER shown in Fig. 20(a), a lower BER is achieved.

6. Conclusion

A lossless robust data hiding scheme is proposed in this paper. Data are embedded into the spatial domain. The original cover image can be restored with no distortion after the hidden data have been extracted if the stego-image has not been altered. The hidden data are robust against non-malicious attacks such as JPEG compression to some extent. Experimental results have shown that the addition of a threshold and a new embedding mechanism can enhance the capacity and make the algorithm more robust. The proposed scheme does not suffer from salt-and-pepper noise and has a significant improvement with respect to previous works in terms of the embedding capacity and robustness. Furthermore, the proposed scheme is simple and efficient, and can be applied to various images as shown in the test images.

Acknowledgments

The authors would like to thank the anonymous referees and the editor for their valuable suggestions and many constructive comments that resulted in the improvement and readability of this paper.

References


About the Author—XIAN-TING ZENG is an associate professor in College of Information Engineering, China Jiliang University. Currently, he is a Ph.D. candidate in the College of Computer Science and Technology, Zhejiang University. His research interests include image processing and data hiding.

About the Author—LING-DI PING is a professor in College of Computer Science and Technology, Zhejiang University. Her research interests include digital watermarking, signal and image processing.

About the Author—XUE-ZENG PAN is a professor in College of Computer Science and Technology, Zhejiang University. His research interests include network information security.