Reversible Data Embedding Scheme Using Relationships between Pixel-Values and Their Neighbors

Chia-Chen Lin, Chin-Chen Chang, and Yi-Hui Chen

Abstract—Reversible data embedding is becoming a very important issue in securing images transmitted over the Internet, especially in dealing with sensitive images such as those created for military data and medical data. Based on the relationships between pixels and their neighbors, we propose a reversible data embedding scheme to embed hidden messages into an original image. In our proposed scheme, a two-layer data embedding approach is used for our proposed data embedding phase. Layer-1 embedding is used to hide secret data. Layer-2 embedding, which is an embedding variant of the proposed layer-1 embedding, is used to hide side information, such as the parameters required to restore the marked image. In our layer-1 embedding, the value of an embedded pixel is determined according to a predetermined threshold and the relationship between the pixel and its neighbors. In our layer-2 embedding, the similar data embedding concept is expanded to the block-based. Experimental results provide supportive data to show that the proposed scheme can provide greater hiding capacity while preserving the image quality of a marked image in comparison with previous work.

Index Terms—Cover image, marked image, reversible data embedding, watermarking.

1. Introduction

Data embedding\cite{1,2,5,11,12,14,16,19,22,23} is a process for hiding data in cover media, and the cover media can be any kind of data such as images, animations, videos, electronic documents, or any similar digitized medium. Two sets of data are linked with the data embedding process: one set is the embedded message and the other is the cover media. The relationship between these two sets of data characterizes two applications: secure communications and authentication. In secure communications, the hidden messages usually are irrelevant to the cover medium, thus enhancing the security of any hidden message. The cover image is a decoy of no value to the recipient, and therefore the receiver should have no interest in the original cover image. As a result, there is no demand to restore the original cover image after any embedded message is extracted. In authentication, the hidden messages are closely related to the cover medium, and the hidden message can be an authentication code, a logo, a watermark, or any similar message.

In most cases, the embedded message damages the cover medium somewhat, making it difficult to restore the original medium even after the embedded message is extracted. However, in some applications, for example medical imagery and law enforcement, the original cover medium must be recovered after the embedded message is extracted to meet legal requirements. For other applications, such as remote sensing, astronomic research, and high-energy particle physical experimental investigation, the requirement for highly precise contents also makes it desirable for the marked medium to be inverted back to the original.

To respond to these demanding situations, reversible data embedding, which is also called lossless data embedding or distortion-free data embedding, has been proposed. Reversible data embedding embeds an invisible hidden message into a cover medium in a reversible fashion. Over the past decade, many reversible data embedding schemes have been proposed\cite{3,4,5,11,12,14,16,19,22,23}. The earliest reference to reversible data hiding can be traced back to Barton’s scheme\cite{2} in 1997. Barton’s scheme compresses the bits to be influenced by the embedding operation, then embeds the compressed data and the embedded message into the cover image. In the extracting and recovering phase, the original bits are decompressed and later used to restore the modified bits. In 2001, Honsinger et al.\cite{16} applied reversible data embedding to lossless authentication. However, their scheme suffers from the disturbing salt-and-pepper noise problem. To solve that problem, Vleeschouwer et al.\cite{23} used a circular
interpretation of bijective transformations. However, the hiding capacity and the visible distortion of that scheme are limited.

In 2001, Fridrich et al.\cite{3} presented an invertible watermarking scheme to authenticate digital images in the joint photographic experts group (JPEG) domain. They used an inverse order-2 function to modify the quantization table to enable lossless embedding of one bit per discrete cosine transform (DCT) coefficient. Unfortunately, the hiding capacity of this scheme is limited by the compressed results of the least significant bit (LSB) plane. Then in 2002, Fridrich et al. proposed an improved scheme for enhancing hiding capacity while reducing visible distortion\cite{4}. In this improved scheme, a cover image is first divided into several disjointed groups. Next, a flipping function and a discrimination function are used to categorize each group into one of the three sets: regular (R), singular (S), and unusable (U). The embedded message can only be embedded when a group is categorized into the R set or the S set. The R and S sets have their own group types, for example, the group type of an R set is 1 and that of an S set is 0. If the message data to be hidden in a group do not match the group type of the set to which the group is assigned, then the flipping function is used to flip the group into another set. To restore the cover image, Fridrich et al.’s scheme losslessly compresses the R set and the S set, then concatenates the compression results with the embedded message to be hidden in the cover image.

Later, Celik\cite{57}-\cite{59} devised a generalized LSB embedding scheme, called the G-LSB scheme, to enhance the hiding capacity of Fridrich’s scheme. Celik’s scheme first quantizes each pixel into $L$ levels, then compresses the quantization noise to reverse the cover image. Next, the message data are converted into a string of $L$-ary symbols. Then, the compressed results of the quantization noises and the string of $L$-ary symbols are appended as the payload. Experimental results show that the hiding capacity of Celik’s scheme depends on the number of $L$ levels.

Subsequently, Tian\cite{22} proposed a promising high-capacity reversible data embedding algorithm in which two techniques, difference expansion and G-LSB embedding, are adopted to achieve very high hiding capacity while keeping distortion low. Extra data are required in the restoring process in both Celik’s and Tian’s schemes. In other words, their restoring processes require at least two scans of the marked image. The extra data extracted during the first scan are used to extract the embedded message and restore the original pixels during the second scan.

To support the authentication application and allow the extraction and restoration process to be completed in a single scan, a pixel-based reversible data embedding scheme is proposed in this paper. The proposed scheme uses the relationship between a pixel and its neighbors to determine whether a pixel is embeddable. Therefore, low computation cost during the data embedding phase is an extra advantage of our proposed scheme. In addition, experimental results prove that the proposed scheme can offer higher embedding capacity and better image quality of marked images compared with previous schemes.

The rest of this paper is organized as follows. In Section 2, because the proposed scheme is inspired by Tian’s scheme, we briefly describe Tian’s scheme for later comparison. Our proposed reversible data embedding scheme with high payload capacity and satisfactory image quality of marked images is presented in Section 3. Several experimental results are illustrated and discussed in Section 4. Finally, concluding remarks are given in Section 5.

2. Review of Tian’s Lossless Data Embedding Scheme

In 2003, Tian proposed a high-payload lossless data embedding scheme in \cite{22}. In that scheme, the difference expansion between two neighboring pixels is used to hide one message bit. Because the computations of two pixels may result in underflow and overflow problems, the embedding pixel pairs are further classified into three categories: expandable pairs, changeable pairs, and unchangeable pairs. Two equations are used to categorize pixel pairs as follows.

\[
2\times d + b \leq \min[2(255-\text{avg}), 2\text{avg}+1] \tag{1}
\]

\[
2\times|d/2| + b \leq \min[2(255-\text{avg}), 2\text{avg}+1] \tag{2}
\]

where avg means the average of the pixel pairs, $d$ is the distance between the pixel pairs, and $b$ is the message bit to be embedded into these pixel pairs.

If (1) is satisfied, the embedding pixel pairs are labeled expandable pairs, called ENs. ENs are further classified into three disjoined subsets. If the distance between the pair equals 0 or −1, the ENs belong to a subset, which is called EZ subset. For the remaining ENs, a threshold $h$ is used to determine which ones can hide secrets $s$. The subsets of selected and unselected ENs for hiding secrets $s$ are denoted as EN1 and EN2, respectively.

If the pixel pairs do not satisfy (1) but satisfy (2), the pixel pairs are labeled changeable pairs, called CNs.

If the pixel pairs satisfy neither (1) nor (2), they are labeled unchangeable pairs. A sender embeds secrets $s$ into ENs in EZ and EN1 in a digital image. To make sure the decoder knows which pairs have been selected for hiding secrets $s$, a location map $l$ must be generated. Later, a lossless compression, for example, JBIG2 or arithmetic coding, is applied to reduce the location map size. For pairs in EZ and EN1, a value 1 is set in the location map $l$; for the remaining pairs, a value 0 is assigned. Due to the LSB of the changeable pixels, the expandable pixels in EN2 will be
modified during data embedding. Their original LSBs must be collected into a stream before data are embedded so that the original cover image can be successfully recovered later. A binary bitstream $b$ is generated by appending the original LSB stream $c$ to the end of compressed location map $l$, then appending the secrets stream $s$ to the end of $c$. Finally, the binary bitstream $b$ is embedded into the cover image using Tian’s embedding strategy. After receiving the marked image, receivers can extract the binary bitstream $b$ from the LSBs of all the changeable pixels in the marked image. The compressed location map $l$ is decompressed and the original pixels are restored according to the location map $l$ and Tian’s restoring strategy.

The two examples that followed provide detailed explanations of Tian’s difference expansion embedding and extracting strategies. The first is for expandable pairs and the second is for changeable pairs. For expandable pixel pairs, two neighboring pixels, $P_1$ and $P_2$, are applied to embed one message bit $s$, where the pixel values of $P_1$ and $P_2$ are between 0 and 255 and $s \in \{0, 1\}$. Let us assume that $P_1=40$, $P_2=45$, and $s=1$. A sender can hide a message bit $s$ during the embedding phase by using the following four steps. First, compute the difference $d$ between the two pixels $P_1$ and $P_2$ as $d=|P_2-P_1|=45-40=5$. Second, compute the average, avg, of the two pixels $P_1$ and $P_2$ as $\text{avg} = \left\lfloor \frac{(P_1 + P_2)}{2} \right\rfloor = \left\lfloor \frac{40 + 45}{2} \right\rfloor = 42$. Thirdly, a temporary $d'$ is computed as $d'=2 \times d + s = 2 \times 5 + 1 = 11$. Finally, replace pixels $P_1$ and $P_2$ with $P'_1$ and $P'_2$, respectively, to hide message bit $s$ by using the following formulas:

\[
P'_1 = \text{avg} - \left\lfloor \frac{d'}{2} \right\rfloor = 42 - \left\lfloor \frac{11}{2} \right\rfloor = 37
\]
\[
P'_2 = \text{avg} + \left\lfloor \frac{(d'+1)}{2} \right\rfloor = 42 + \left\lfloor \frac{(11+1)}{2} \right\rfloor = 48.
\]

In the extraction phase for expandable pixel pairs, first, the difference $d'$ and the integer average “avg” between $P'_1$ and $P'_2$ are obtained by computing $d' = P'_2 - P'_1 = 48 - 37 = 11$, and $\text{avg} = \left\lfloor \frac{(P'_1 + P'_2)}{2} \right\rfloor = \left\lfloor \frac{37+48}{2} \right\rfloor = 42$. Next, $d$ can be derived from $d'$ as $d = \left\lfloor \frac{d'}{2} \right\rfloor = 5$. Later, the embedded message bit $s$ can be extracted from $d'$ by computing $s = d' - \left\lfloor \frac{d'}{2} \right\rfloor \times 2 = 11 - 10 = 1$. Finally, the original pixels $P_1$ and $P_2$ are restored as

\[
P_1 = \text{avg} - \left\lfloor \frac{d}{2} \right\rfloor = 42 - \left\lfloor \frac{5}{2} \right\rfloor = 40
\]
\[
P_2 = \text{avg} + \left\lfloor \frac{(d+1)}{2} \right\rfloor = 42 + \left\lfloor \frac{(5+1)}{2} \right\rfloor = 45.
\]

For changeable pixel pairs, assume that $P_3=254$, $P_4=250$ and $s=1$. In the embedding phase, the difference $d$ between $P_3$ and $P_4$ is computed as $d=P_3-P_4=4$ and the integer average is computed as $\text{avg} = \left\lfloor \frac{(P_3 + P_4)}{2} \right\rfloor = \left\lfloor \frac{254+250}{2} \right\rfloor = 252$. Second, $d$ is converted to a binary stream such as “100” and uses the message bit $s$ “1” to substitute the LSB of “100” as “1”, then the original LSB “0” is recorded. Thirdly, the binary stream “101” is converted into a decimal digit such as $d'=5$. Finally, the sender replaces pixels $P_3$ and $P_4$ with $P'_3$ and $P'_4$, respectively, to embed hidden bit $s$ “1” by

\[
P'_3 = \text{avg} + \left\lfloor \frac{(d'+1)}{2} \right\rfloor = 252 + \left\lfloor \frac{(5+1)}{2} \right\rfloor = 255
\]
\[
P'_4 = \text{avg} - \left\lfloor \frac{d'}{2} \right\rfloor = 252 - \left\lfloor \frac{5}{2} \right\rfloor = 250.
\]

In the extraction phase for changeable pixel pairs, firstly, the difference $d$ and the integer average avg between $P'_3$ and $P'_4$ are obtained by computing $d' = P'_3 - P'_4 = 255 - 250 = 5$, and $\text{avg} = \left\lfloor \frac{(P'_3 + P'_4)}{2} \right\rfloor = \left\lfloor \frac{255+250}{2} \right\rfloor = 252$. Then, $d'=5$ is converted into a binary stream $d'=(101)_2$, and the message bit $s$ is retrieved as $s=1$ from the LSB of $d'$. Thirdly, the original LSB “0” is retrieved from extra information to substitute the LSB of $d'$, so that $d=(100)_2 = 4$. Finally, the original pixels $P_3$ and $P_4$ are restored as

\[
P3 = \text{avg} + \left\lfloor \frac{(d+1)}{2} \right\rfloor = 252 + \left\lfloor \frac{(4+1)}{2} \right\rfloor = 254
\]
\[
P4 = \text{avg} - \left\lfloor \frac{d}{2} \right\rfloor = 252 - \left\lfloor \frac{4}{2} \right\rfloor = 250.
\]

In this paper, we propose a lossless data embedding scheme that can conceal large amounts of embedded messages in a cover image with better image quality of the corresponding marked image in comparison with Tian’s scheme. A distinct feature of the proposed scheme is that the difference between one pixel and the mean value of its neighbors is considered to determine the degree of expansion required to embed message data. The proposed scheme does not need to record the LSB of a pixel and a location map, which is required by Tian’s scheme to offer high payload capacity; moreover, the proposed scheme outperforms both Tian’s and Celik’s schemes in payload and image quality of marked images.

3. Proposed Scheme

In this section, we describe the global concept of the proposed scheme. Generally, only a minute difference exists between a pixel and its neighbors in the smooth area of an image, whereas a larger difference exists between a pixel and its neighbors in the complex area of an image. Therefore, the variations between pixels and their neighbors are considered during data embedding to ensure that high payload capacity and good image quality of the marked image can be achieved at the same time. The proposed scheme consists of two phases: data embedding and restoring. A two-layer data embedding approach is used for
our proposed data embedding phase. Layer-1 embedding is used to hide secret data. Layer-2 embedding, which is an embedding variant of the proposed layer-1 embedding, is used to hide side information, such as the parameters required to restore the marked image generated in layer-1 embedding. The following subsections describe the two phases in detail.

3.1 Embedding Phase

Fig. 1 shows a flowchart of the proposed reversible data embedding procedure, where EOF is the meaning of the end of a file. The procedure can be broken down into two layers: layer-1 and layer-2 embedding.

The secret data are embedded into the cover image by using layer-1 embedding and the parameters required for restoring the original pixels of the cover image are embedded into the marked image generated during layer-1 embedding by using layer-2 embedding. The following subsections describe layer-1 and layer-2 embedding in detail.

A. Layer-1 Embedding

The flowchart of layer-1 embedding is shown in Fig. 2. In the first stage, the proposed reversible data embedding procedure scans all pixels except the first pixel in the first line of an original cover image \( I \) in raster-scanning order to determine each pixel is embeddable or unembeddable according to our estimation policies.

The difference between a scanned pixel and its neighbors determines whether the scanned pixel is embeddable. The neighbors of a scanned pixel are determined as shown in Fig. 3. Basically, the number of neighbors for the current pixel, which is denoted as \( x_1 \) in the following paragraphs, is adaptive according to the value of \( N \), where \( N \) is the layer number.

In Fig. 3, the neighbors of pixel \( x_1 \) are \( \{ x_2, x_3, x_4, x_5 \} \) and \( \{ x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12}, x_{13}, x_{14} \} \) when \( N=1 \) and \( N=2 \), respectively. Not all neighbors in the \( N \)th layer are considered for pixel \( x_1 \) when pixel \( x_1 \) is located along the boundaries of an image. Take pixels \( x_{16} \) and \( x_{18} \) in Fig. 3 for example: pixels \( x_{16} \) and \( x_{18} \) are located in the first column and the first row in image \( I \), respectively; therefore, the neighbors of pixels \( x_{16} \) and \( x_{18} \) are only \( \{ x_{17}, x_{18} \} \) and \( \{ x_{13}, x_{14} \} \), respectively, when \( N=1 \). The pixel located in the left-most upper corner, such as \( x_{17} \) in Fig. 3, is not used for data embedding because there is no pixel located to its left according to our definition of a pixel’s neighbors.

Later, for each scanned pixel (except the pixel located in the left-most upper corner of an image), three neighbor parameters of the pixel, avg, var, and maxDiff, are generated to evaluate whether it is embeddable. The values avg and var of the scanned pixel are the mean and variance of its neighbors, respectively. The maxDiff of the scanned pixel is the maximum difference between avg and its neighbors. Taking pixel \( x_1 \) in Fig. 4 for example, its neighbors are \( x_2, x_3, \ldots, x_{13} \) when \( N=2 \). The mean and variance for pixels \( x_2, x_3, \ldots, x_{13} \) are 8 and 18.92, respectively, and maxDiff is the maximum difference between \( \text{avg} \) and \( \text{avg} \)’s neighbors is 12. In other words, the \( \text{avg} \), \( \text{var} \), and maxDiff of pixel \( x_1 \) when \( N=2 \) are 8, 18.92, and 12, respectively. Note that the neighbors of pixels \( x_1 \) are the original pixels in the original image rather than the modified pixels.

Before the layer-1 embedding procedure begins, the proposed scheme also considers that underflow and overflow may occur after data embedding. To prevent underflow and overflow from occurring, two important parameters, \( p \) and \( q_i \), are used to detect pixels that will have overflow/underflow problems after layer-1 data embedding, as shown in Lines 11 and 12 in the estimation algorithm in Fig. 5. Pixels are judged unembeddable if they will have the underflow or overflow problem after data embedding. Once a scanned pixel is determined to be embeddable, one of our designed embedding strategies will be adopted for layer-1 data embedding according to its corresponding maxDiff.
The proposed layer-1 embedding phase starts from the second pixel in the first row of the cover image and the scanned pixel is denoted as \(x_1\).

Step 1. Compute the neighbor parameters for the scanned pixel \(x_1\).

Compute \(\text{avg}(x_1)\), \(\text{var}(x_1)\), and \(\text{maxDiff}\) of the scanned pixel \(x_1\) according to (3):

\[
\text{avg}(x_1) = \left( \sum_{j=2}^{2^{j-1}+2^{j-1}} x_j \right)/\left[ 2N(N+1) \right], \tag{3a}
\]

\[
\text{var}(x_1) = \sqrt{\left( \sum_{j=2}^{2^{j-1}+2^{j-1}} \left[ \text{avg}(x_j) - x_j \right]^2 \right)/\left[ 2N(N+1) \right]} \tag{3b}
\]

\[
\text{maxDiff} = \max(\text{avg}(x_1) - x_j) \quad \text{(3c)}
\]

where \(j = 2, 3, \ldots, (2^N+2^N+1)\). \(N\) is the layer number for the scanned pixel \(x_1\), \(| \cdot |\) is the absolute value, and the Max() function returns the maximal difference between \(\text{avg}(x_1)\) and the neighbors of scanned pixel \(x_1\).

Step 2. Evaluate whether the scanned pixel \(x_1\) is embeddable.

Based on the difference between scanned pixel \(x_1\) and its \(\text{avg}(x_1)\), scanned pixel \(x_1\) is judged embeddable or unembeddable according to the proposed estimation policies (listed in Line 11 to Line 13) in Fig. 5. If the scanned pixel is embeddable, go to Step 3; otherwise, go to Step 4. According to the estimation algorithm in Fig. 5, the embeddable pixels located in the smooth area are determined to be case 1 and embeddable pixels located in the complex area are determined to be case 2. To ensure that the good quality of the marked image and the hiding capacity can be enhanced at the same time, we not only design the two embedding strategies demonstrated in (4) and (5) (shown in Step 3) for both case 1 and case 2, but also further classify each case into two scenarios. For embeddable pixels with a variance less than or equal to \(t_1\), the amount of hidden message \(n\) is set as 1 bit and the embedded message \(b\) is set as 0 or 1. For embeddable pixels that have a variance larger than \(t_1\) and less than \(t_2\), the amount of hidden message \(n\) is set as 2 bits and the embedded message \(b\) ranges between 0 and 3. Based on our experimental results, the best range of \(t_1\) is between 5 and 10, and that of \(t_2\) is between 10 and 20. In general, \(t_1\) and \(t_2\) are smaller for smooth images than for complex images. The best value of \(t_1\) is to let the condition listed in Line 8 in Fig. 5 be satisfied for embeddable pixels that belong to case 1 so that the distortion caused by data embedding can be limited. Details of our proposed embedding strategies are presented in Step 3.

In Fig. 5, \(n\) represents the amount of secret bits that can be embedded into an embeddable pixel. The difference between scanned pixel \(x_1\) and its avg is depicted as \(d\), and \(d\) usually is \(p\) (or \(q\)) times \(\text{maxDiff}\) or is equal to \(\text{maxDiff}\). Therefore, the parameters \(p\) and \(q\) serve as boundary controllers to prevent the embedded pixels from entering the overflow/underflow state (listed in Lines 11 and 12 in Fig. 5). Furthermore, \(p\) and \(q\) also can be treated as the image quality and hiding capacity controllers because larger \(p\) and \(q\) reduce the range of embeddable pixels. Typically, larger \(p\) and \(q\) lead to higher image quality but lower hiding capacity. Based on our experiments, the range of parameters \(p\) and \(q\) is between 11 and 20. Detailed discussions of the effects of parameters \(p\) and \(q\) on image quality and hiding capacity are given in Table 3 in Section 4 of this paper.

Generally, the value of \(d\) can be very similar to \(\text{maxDiff}\), especially when the scanned pixel is located in the smooth area. Conversely, when the scanned pixel is located in the complex area, its maxDiff can be large. A smaller maxDiff usually preserves good image quality. Therefore, the proposed estimation algorithm tends to judge a scanned pixel with a small maxDiff as an embeddable pixel and a scanned pixel with a large maxDiff as an unembeddable pixel. A predefined threshold \(T\) is used to decide whether an embeddable pixel is located in a smooth or complex area. Based on our experimental results, the best range of \(T\) is between 15 and 30, with \(T=20\) being the best choice.

Step 3. Embed data.

If scanned pixel \(x_1\) is determined to be case 1 of Step 2 according to the estimation algorithm shown in Fig. 5, the embedded pixel \(\hat{x}_1\) is computed by using (4) where \(b\) is a secret message ranging from 0 to \(w-1\), with \(w=2^n\) and \(n\) is secret bits. Generally, the difference \(d\) is small when it is in a smooth area. Based on (4), the difference between \(x_1\) and \(\hat{x}_1\) is \((d\pm b)\) when \(w\) is equal to 2. In addition, when \(d\) is small \(\hat{x}_1\) will be similar to \(x_1\).

\[
\hat{x}_1 = \begin{cases} 
\text{avg} + w \times d + b, & \text{if } x_1 > \text{avg} \\
\text{avg} - w \times d - b, & \text{otherwise.}
\end{cases} \tag{4}
\]
If the scanned pixel $x_i$ is determined to be case 2 in Step 2, the embedded pixel $\hat{x}_i$ is computed as follows:

$$\hat{x}_i = \begin{cases} 
    \text{avg} + T \times w - (d + w - 1) \times w + b, & \text{if } x_i > \text{avg}; \\
    \text{avg} - T \times w + (d + w - 1) \times w - b, & \text{otherwise}. 
\end{cases}$$

Whether the embeddable pixels belong to case 1 or case 2, the amount of hidden secret bits is set as 1 bit when the variance of the scanned pixel is less than or equal to $t_1$. Only when the embeddable pixels have a variance larger than $t_1$ and less than $t_2$, can the amount of hidden message be set as 2 bits, and ranges the hidden message $b$ between 0 and 3. There is one exceptional scenario that requires different processing. That exception is when the scanned pixel $x_i$ is near 0 or 255 and its maxDiff is zero but its $d$ is not equal to zero. In this case, $(\text{avg} - w \times \text{maxDiff} \times x_i - b) \geq 0$ or $(\text{avg} + w \times \text{maxDiff} \times x_i + b) \leq 255$ is always satisfied and the scanned pixel $x_i$ is judged embeddable. However, after data embedding, the embedded pixel $\hat{x}_i$ will suffer from the problem of underflow or overflow. Therefore, the location of scanned pixel $x_i$ must be recorded and the pixel must be marked unembeddable. In our experiments, only two pixels in the “Pepper” image fit the above scenario, while the others do not suffer from the problem. Because this situation occurs infrequently and the pixel locations are very small, these exceptional pixel locations can be treated as part of the parameters. Finally, the parameters can be embedded into the marked image, which is generated from layer-1 embedding, by using layer-2 embedding.

Step 4. Output the embedded pixel value.

Finally, output the embedded pixel to construct a marked image. Then, repeat Step 1 to Step 4 to process the next inputted pixel until all pixels of an image $I$ have been thoroughly scanned. The parameters used during the layer-1 embedding phase are required to extract the embedded message and restore the original pixels. Therefore, these parameters will be embedded into the marked image generated from layer-1 embedding by using layer-2 embedding.

B. Layer-2 Embedding

In essence, layer-2 embedding is a variant of layer-1 embedding. The differences between layer-1 and layer-2 embedding are: 1) layer-1 embedding is pixel-based and layer-2 embedding is block-based, and 2) layer-1 hides secret data and layer-2 hides the parameters required to restore the secret data hidden with layer-1. The layer-2 embedding phase is broken down into five steps as follows.

Step 1. Divide the image into non-overlapping blocks.

In layer-2 embedding, the marked image is first divided into several non-overlapping blocks sized 3 pixels × 3 pixels. For clarity, the pixels in a block are denoted as shown in Fig. 6 (a) and (b).

Step 2. Evaluate whether a block is smooth.

The four masks shown in Fig. 6 (c), (d), (e), and (f), are used to evaluate whether the current block is smooth. Using four masks and (6), four temporary values $R_{SR}$, $R_{SS}$, $R_{ST}$, and $R_{SU}$ are derived. The smaller temporary values imply pixels $Q_1$, $Q_2$, $Q_4$, $Q_5$, $Q_6$, and $Q_8$ are similar to each other, and vice versa. A block is judged smooth if its $R_{SR}$, $R_{SS}$, $R_{ST}$, and $R_{SU}$ are equal to 0.

$$R_{SR} = \sum_{i,j}^{3 \times 3} (Q' \times SR)_{i,j}$$

$$R_{SS} = \sum_{i,j}^{3 \times 3} (Q' \times SS)_{i,j}$$

$$R_{ST} = \sum_{i,j}^{3 \times 3} (Q' \times ST)_{i,j}$$

$$R_{SU} = \sum_{i,j}^{3 \times 3} (Q' \times SU)_{i,j}$$

Step 3. Generate candidate pool.

If a block is judged as a smooth block in Step 2, its corresponding block ID is inserted into the candidate pool. Steps 2 and 3 are repeated until all blocks are processed.

Step 4. Generate block pairs based on blocks in the candidate pool.

First, compute mean values of blocks in the candidate pool. Sort all blocks in ascending order according to their mean values. Note that the initial mean value is 128, and ranges from 128–128(=0) to 128+128(=256). Two nearest blocks with the same mean value are treated as a pair. Step 4 is repeated until all blocks in the candidate pool are processed.

Step 5. Embed data.

The value of parameter $XP$ determines how many block pairs are used to embed data, which is determined according to (7):

$$XP = 40 \text{bits} + 10 \text{bits} + 2 \times 9 \times n$$

For clarity, the pixels in a block are denoted as shown in Fig. 6 (a) and (b).

Fig. 6. Example of pixels in a block and four evaluating masks: (a) example of pixels in a block, (b) example of pixels for block filtering, (c) matrix SR, (d) matrix SS, (e) matrix ST, and (f) matrix SU.
where 40 bits are used to represent five parameters \( t_1, T, t_2, p, q \), and each parameter requires 8 bits to represent itself. Ten bits is used to represent the number of exceptional pixels. Basically, the number of exceptional pixels in an image is less than 10, and \( n \) is equal to the number of exceptional pixels. Once the number of exceptional pixels is known, the number of bits for recoding the coordination of exceptional pixels (2×9×\( n \)) can be determined.

Select the first block pair to embed parameter data by using (8):

\[
Q_{5}^{(2)} = \begin{cases} 
Q_{5}^{(1)} + w \times d + b, & \text{if } Q_{5}^{(2)} > Q_{5}^{(1)} \\
Q_{5}^{(1)} - w \times d - b, & \text{otherwise.}
\end{cases}
\]  

where \( Q_{5}^{(1)} \) is \( Q_{5} \) located in the first block in the block pair and \( Q_{5}^{(2)} \) is \( Q_{5} \) located in the second block in the same block pair. \( Q_{5}^{(2)} \) is the embedded \( Q_{5} \) located in the second block in the block pair. \( d = |Q_{5}^{(2)} - Q_{5}^{(1)}|, w=2, \) and \( b \) is secret data.

### 3.2 Restoring Phase

Fig. 7 shows the flowchart of our proposed restoring phase, from Fig. 7 we can see that layer-1 restoring and layer-2 restoring are the inverse procedures of layer-1 data embedding and layer-2 data embedding, respectively. In this subsection, we describe the proposed restoring phase, in which the parameters are extracted by using layer-2 restoring. Embedded secret messages are then extracted and the original cover image is restored by using layer-1 restoring.

#### A. Layer-2 Restoring

Layer-2 restoring is the inverse of the layer-2 embedding. Layer-2 is broken down into five steps. Step 1 to Step 4 are the same for layer-2 embedding. Only Step 5 is different from that in layer-2 embedding. Therefore, Step 1 to Step 4 are skipped in the following subsection, while Step 5 is discussed in greater detail.

Step 5. Extract hidden parameters and restore the original \( Q_{5}^{(1)} \).

Extract hidden data \( b \) by computing \( b = |Q_{5}^{(1)} - Q_{5}^{(2)}| \mod w, \) where \( w \) is equal to 2. Later, the original pixel \( Q_{5}^{(1)} \) is reversed by using

\[
Q_{5}^{(2)} = \begin{cases} 
Q_{5}^{(1)} + d, & \text{if } Q_{5}^{(2)} > Q_{5}^{(1)} \\
Q_{5}^{(1)} - d, & \text{otherwise.}
\end{cases}
\]  

where \( d = (|Q_{5}^{(1)} - Q_{5}^{(2)}| - b)/w \) and \( w \) is equal to 2.

After the receiver extracts the parameters, the receiver can perform the layer-1 restoring procedure to extract hidden data from the marked image. As discussed in Section 3.2, while the original image is recovered, the hidden data are extracted.

#### B. Layer-1 Restoring

First, the proposed scheme scans the marked image in raster-scanning order except for the pixel located in the left-most upper corner of the marked image, and decides whether the embedded messages are hidden in the current scanning pixel according to its neighbors. Next, the scheme uses its extracting strategies to simultaneously extract the embedded messages and reverse the original pixel value. Layer-1 restoring requires only those parameters that are extracted during layer-2 restoring; no extra data are required during the layer-1 restoring procedure. Note that layer-1 restoring starts from the second pixel in the first row of the marked image, just as with the layer-1 embedding procedure because our scheme uses the neighbors’ original pixel values instead of their modified values to decide whether the currently scanned pixel is embeddable and what kind of embedding strategy is to be used. The first pixel is not processed during layer-1 embedding; however, it can be used to extract the hidden bit from the second pixel and restore the second pixel when the second pixel is judged embeddable. After the second pixel is restored, it can be used to extract the hidden bit from the next embeddable pixel and restore it. The same process continues until all pixels in the marked image are processed.

The layer-1 restoring procedure can be broken into the four steps shown in Fig. 8. The details of each step are described below.

Step 1. Compute the neighbor parameters for scanned pixel \( x_1 \).

Compute \( \text{avg}(x_1), \text{var}(x_1), \) and \( \text{maxDiff}(x_1) \) for the scanned pixel \( x_1 \) according to (3).

Step 2. Evaluate whether the scanned pixel \( x_1 \) is embeddable.

The scanned pixel \( x_1 \) is judged embeddable or unembeddable according to the proposed estimation policies in Fig. 5, which are described in subsection A. If the scanned pixel is embeddable, go to Step 3; otherwise, go to Step 4.

![Fig. 8. Flowchart showing the four steps in our proposed layer-1 restoring phase.](image)
Step 3. Extract the embedded message.

If the scanned pixel is determined to be case 1 in Step 2, the scheme first retrieves the embedded message \( b \) by computing \( b = [\text{avg} - \hat{x}_i] \mod w \) where \( w = 2^t \). If \( b \) is less than or equal to \( t_1 \), \( n \) is set at 1; otherwise, \( n \) is set at 2. The difference \( d \) between avg and embedded pixel \( \hat{x}_i \) is obtained by computing \( d = ([\text{avg} - \hat{x}_i] \mod w) \). Later, the original pixel \( x_i \) is reversed by using

\[
x_i = \begin{cases} \text{avg} - d, & \text{if } \hat{x}_i > \text{avg} \\ \text{avg} + d, & \text{otherwise.} \end{cases}
\]  

If the scanned pixel is determined to be case 2 in Step 2, the original pixel \( x_i \) is restored by using

\[
x_i = \begin{cases} \text{avg} - d, & \text{if } \hat{x}_i > \text{avg} \\ \text{avg} + d, & \text{if } \hat{x}_i \leq \text{avg} \\ \text{avg} - \text{avg} - b, & \text{if } \hat{x}_i < \text{avg} - d \end{cases}
\]

Step 4. Output the pixel value.

Finally, output the restored pixel to construct the original cover image. Then, repeat Step 1 to Step 4 to process the next scanned pixel until all pixels have been thoroughly scanned and processed.

3.3 Examples of Data Embedding and Restoring Phases

Because layer-2 embedding is a variant of layer-1 embedding, this subsection contains only examples of our proposed layer-1 embedding and layer-1 restoring. To give a better explanation of our proposed layer-1 embedding and layer-1 restoring procedures, we describe four embedding instances with two scenarios, case 1 and case 2 with \( \text{var} \leq t_1 \), in Table 1. Instances \( x_1 \) and \( x_2 \) presented in case 1 demonstrate the embedding results when two times \( \text{maxDiff} \) is less than threshold \( T \). Instances \( x_3 \) and \( x_4 \) listed in case 2 present the embedding results when two times \( \text{maxDiff} \) is greater than or equal to \( T \) and its var is less than or equal to \( t_1 \). The corresponding restoring results are listed in Table 2. In this subsection, we use only instance \( x_1 \) to demonstrate our embedding strategy; the others simply follow the same embedding rule to hide the embedded message.

Assume that the \( \text{maxDiff} \) of \( x_1 \) is 5, \( x_1 \) is 42, the embedded message bit is 0, \( T \) is 15, \( w \) is 2, \( p \) is 1, and the value of avg is 45. Because \( x_1 \) is less than avg (42<45), the encoder further estimates whether it satisfies the condition \( (\text{avg} - w \times \text{maxDiff} \neq p) \). In this case, the condition is satisfied. In addition, because two times the \( \text{maxDiff} \) of \( x_1 \) is less than \( T \) (10<15), the case 1 embedding strategy is applied to embed message data into the current pixel. Furthermore, \( x_1 \) is less than avg so the value of \( x_1 \) is replaced with \( \hat{x}_1 \) by using (4) and the embedded pixel is computed as \( \text{avg} - w \times d - b = 45 - 2 \times 3 - 0 = 39 \). To provide a clearer demonstration, eight embedding examples are listed in Table 1 and their corresponding extracting results are shown in Table 2.

To demonstrate simply how our proposed layer-1 restoring extracts the hidden bit and restores the original pixels, we later use the embedded results of the example above to explain our extracting strategies. We assume the value of \( \hat{x}_1 \) is 39. By using pixel \( \hat{x}_1 \)'s neighbors, the encoder computes the values of avg and \( \text{maxDiff} \), and we assume these two values are 45 and 5, respectively. Then, we apply avg and \( \text{maxDiff} \) to estimate whether pixel \( \hat{x}_1 \) is embedded with the embedded message bit. Because the condition \( (\text{avg} - w \times \text{maxDiff} < p) \) is satisfied, the decoder estimates that the current pixel contains embedded data. In addition, two times its \( \text{maxDiff} \) is less than \( T \), so pixel \( \hat{x}_1 \) is judged to be case 1. Because pixel \( \hat{x}_1 \) is less than avg, the hidden data \( b \) can be derived by computing \( b = [\text{avg} - \hat{x}_1] \mod w \). As a result, when the value of \( \hat{x}_1 \) is 39, its corresponding hidden bit is derived as 0. Later, we can obtain the value of its difference \( d \) by computing \( d = ([\text{avg} - \hat{x}_1] \mod w) \) and then restore the original pixel value \( x_1 \). Because the value of \( d \) is 3, the original pixel value is restored as 42 by using (10). Eight restoring examples are shown in Table 2.
In our experiments, nine images served as test images: “Lena”, “Pepper”, “F16”, “Boat”, “Splash”, “Head”, “Watch”, “Zelda”, and “Cornfield”. Six of these test images are 512 pixels grayscale images. For portability, we implemented the proposed scheme and Tian’s scheme with Java. To simulate Tian’s data embedding strategies, we used arithmetic coding to compress the location map generated by Tian’s scheme. The simulation platform is Microsoft Windows XP, Pentium III with 512 MB memory. Two performance criteria were used to measure the performance of our proposed hiding scheme in hiding capacity and image quality of the marked image. The hiding capacity denotes the volume of the message embedded in a marked image. The image quality of the marked image is evaluated by peak signal-to-noise ratio (PSNR), which is defined as

$$\text{PSNR} = 10 \log \left( \frac{255^2}{\text{MSE}} \right) \text{dB}. \quad (8)$$

where 255 represents the maximum value of each pixel and the mean square error (MSE) for an image is defined as

$$\text{MSE} = \frac{1}{HW} \sum_{i=1}^{H} \sum_{j=1}^{W} (x_{i,j} - \hat{x}_{i,j})^2 \quad (9)$$

where $H$ and $W$ represent the height and width of an image, $x_{i,j}$ is the pixel value of coordinate $(i, j)$ in the original image, and $\hat{x}_{i,j}$ is the pixel value after the embedding process.

In general, the higher the PSNR value, the better the image quality will be. Conversely, once the image quality becomes worse, its PSNR value also becomes small. In addition, a larger hiding capacity may result in low image quality, and vice versa. It is always a challenge to maintain high image quality of a marked image while providing large hiding capacity. The experimental results listed in Table 3 for the “Lena”, “Pepper”, “F16”, and “Boat” images demonstrate the relationship between PSNR and different $p$ and $q$, and between hiding capacity and different $p$ and $q$ when using the proposed scheme without layer-2 embedding. The function of $p$ and $q$ is to make sure any potential underflow or overflow problem can be prevented and the property of any modified pixels, which is listed in Line 11 and Line 12 of the estimation algorithm, can be maintained so that the extraction and restoring procedures can be performed correctly. For example, if an original pixel is less than its avg, its modified pixel value must be still less than its avg after the secret message is embedded.
Therefore, the number of embeddable pixels is gradually increased when the corresponding $p$ and $q$ are decreased. As Table 3 shows, although decreasing parameters $p$ and $q$ gradually result in a decrease of PSNR, the pure hiding capacity is enhanced gradually. In “Lena”, for example, the largest pure hiding capacity is up to 167057 bits; and for “F16”, the largest pure hiding capacity is up to 181810 bits when parameters $p$ and $q$ are set at 11. Furthermore, in the worst case, the average PSNR of the marked images still is higher than 39.38 dB. The reason each test image has different $p$ and $q$ is to make sure the hiding capacity of each image can be as large as possible while maintaining a satisfactory image quality of the marked image. That is why the initial values of $p$ and $q$ for “Pepper” are from 13 instead of 11; overflow/underflow results occur when $p$ and $q$ are set at 11 or 12. The $p$ and $q$ are derived by our estimation policies and experiments based on the individual characteristics of each image. Note that $p$ and $q$ can be different values, and can always be justified to find the best combination to provide higher hiding capacity with slight image distortion.

![Image 3](http://example.com/image3.png)

Table 3: PSNRs and hiding capacity of our proposed scheme based on $T=20$, $t_1=10$, and $t_2=20$ for four test images.

<table>
<thead>
<tr>
<th>Images</th>
<th>$p$</th>
<th>$q$</th>
<th>PSNR (dB)</th>
<th>Hiding capacity (bits)</th>
<th>Pure hiding capacity (bits)</th>
<th>Hiding ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lena</td>
<td>11</td>
<td>11</td>
<td>39.46</td>
<td>167107</td>
<td>167057</td>
<td>0.637</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>12</td>
<td>39.85</td>
<td>166962</td>
<td>160642</td>
<td>0.612</td>
</tr>
<tr>
<td></td>
<td>13</td>
<td>13</td>
<td>40.23</td>
<td>154527</td>
<td>154477</td>
<td>0.589</td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>14</td>
<td>40.53</td>
<td>149086</td>
<td>149036</td>
<td>0.568</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>15</td>
<td>40.84</td>
<td>144109</td>
<td>144059</td>
<td>0.549</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>16</td>
<td>41.13</td>
<td>138999</td>
<td>138949</td>
<td>0.530</td>
</tr>
<tr>
<td></td>
<td>17</td>
<td>17</td>
<td>41.46</td>
<td>133793</td>
<td>133743</td>
<td>0.510</td>
</tr>
<tr>
<td></td>
<td>18</td>
<td>18</td>
<td>41.73</td>
<td>129141</td>
<td>129091</td>
<td>0.492</td>
</tr>
<tr>
<td></td>
<td>19</td>
<td>19</td>
<td>41.97</td>
<td>125280</td>
<td>125230</td>
<td>0.477</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>20</td>
<td>42.20</td>
<td>121385</td>
<td>121335</td>
<td>0.463</td>
</tr>
<tr>
<td>Pepper</td>
<td>13</td>
<td>13</td>
<td>39.45</td>
<td>142327</td>
<td>142241</td>
<td>0.542</td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>14</td>
<td>39.76</td>
<td>149151</td>
<td>149065</td>
<td>0.568</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>15</td>
<td>40.09</td>
<td>142048</td>
<td>141962</td>
<td>0.541</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>16</td>
<td>40.41</td>
<td>135121</td>
<td>135035</td>
<td>0.515</td>
</tr>
<tr>
<td></td>
<td>17</td>
<td>17</td>
<td>40.71</td>
<td>128859</td>
<td>128773</td>
<td>0.491</td>
</tr>
<tr>
<td></td>
<td>18</td>
<td>18</td>
<td>40.98</td>
<td>123326</td>
<td>123240</td>
<td>0.470</td>
</tr>
<tr>
<td></td>
<td>19</td>
<td>19</td>
<td>41.25</td>
<td>118392</td>
<td>118306</td>
<td>0.451</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>20</td>
<td>41.55</td>
<td>113085</td>
<td>112999</td>
<td>0.431</td>
</tr>
<tr>
<td>F16</td>
<td>11</td>
<td>11</td>
<td>39.54</td>
<td>181878</td>
<td>181810</td>
<td>0.693</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>12</td>
<td>39.96</td>
<td>175996</td>
<td>175928</td>
<td>0.671</td>
</tr>
<tr>
<td></td>
<td>13</td>
<td>13</td>
<td>40.35</td>
<td>170765</td>
<td>170697</td>
<td>0.651</td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>14</td>
<td>40.73</td>
<td>166013</td>
<td>165945</td>
<td>0.633</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>15</td>
<td>41.02</td>
<td>160298</td>
<td>160230</td>
<td>0.611</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>16</td>
<td>41.37</td>
<td>154477</td>
<td>154409</td>
<td>0.589</td>
</tr>
<tr>
<td></td>
<td>17</td>
<td>17</td>
<td>41.69</td>
<td>148921</td>
<td>148853</td>
<td>0.568</td>
</tr>
<tr>
<td></td>
<td>18</td>
<td>18</td>
<td>41.97</td>
<td>144962</td>
<td>144894</td>
<td>0.552</td>
</tr>
<tr>
<td></td>
<td>19</td>
<td>19</td>
<td>42.27</td>
<td>141472</td>
<td>141404</td>
<td>0.539</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>20</td>
<td>42.57</td>
<td>138764</td>
<td>138716</td>
<td>0.529</td>
</tr>
<tr>
<td>Boat</td>
<td>12</td>
<td>12</td>
<td>39.38</td>
<td>147511</td>
<td>147461</td>
<td>0.562</td>
</tr>
<tr>
<td></td>
<td>13</td>
<td>13</td>
<td>39.83</td>
<td>142791</td>
<td>142241</td>
<td>0.542</td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>14</td>
<td>40.27</td>
<td>137532</td>
<td>137482</td>
<td>0.524</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>15</td>
<td>40.66</td>
<td>133221</td>
<td>133171</td>
<td>0.508</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>16</td>
<td>40.99</td>
<td>129727</td>
<td>129677</td>
<td>0.494</td>
</tr>
<tr>
<td></td>
<td>17</td>
<td>17</td>
<td>41.24</td>
<td>126646</td>
<td>126596</td>
<td>0.483</td>
</tr>
<tr>
<td></td>
<td>18</td>
<td>18</td>
<td>41.54</td>
<td>122889</td>
<td>122839</td>
<td>0.468</td>
</tr>
<tr>
<td></td>
<td>19</td>
<td>19</td>
<td>41.84</td>
<td>118662</td>
<td>118612</td>
<td>0.452</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>20</td>
<td>42.09</td>
<td>114671</td>
<td>114621</td>
<td>0.437</td>
</tr>
</tbody>
</table>

![Image 4](http://example.com/image4.png)

Table 4: PSNRs and hiding capacity of our proposed scheme with/without layer-2 based on $T=20$, $t_1=10$, $t_2=20$ and $p=q=13$.

<table>
<thead>
<tr>
<th>Images</th>
<th>With layer-2</th>
<th>Without layer-2</th>
<th>PSNR (dB)</th>
<th>Hiding capacity (bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lena</td>
<td>With</td>
<td>Without</td>
<td>40.23</td>
<td>154526</td>
</tr>
<tr>
<td>Pepper</td>
<td>With</td>
<td>Without</td>
<td>39.45</td>
<td>142327</td>
</tr>
<tr>
<td>Boat</td>
<td>With</td>
<td>Without</td>
<td>39.83</td>
<td>142241</td>
</tr>
<tr>
<td>F16</td>
<td>With</td>
<td>Without</td>
<td>40.35</td>
<td>170765</td>
</tr>
<tr>
<td>Splash</td>
<td>With</td>
<td>Without</td>
<td>38.39</td>
<td>183258</td>
</tr>
<tr>
<td>Head</td>
<td>With</td>
<td>Without</td>
<td>41.91</td>
<td>185965</td>
</tr>
<tr>
<td>Watch</td>
<td>With</td>
<td>Without</td>
<td>41.95</td>
<td>172001</td>
</tr>
<tr>
<td>Zelda</td>
<td>With</td>
<td>Without</td>
<td>40.40</td>
<td>177771</td>
</tr>
<tr>
<td>Cornfield</td>
<td>With</td>
<td>Without</td>
<td>41.58</td>
<td>129442</td>
</tr>
</tbody>
</table>

Although “Lena” and “F16” can offer the largest hiding capacity when $p$ and $q$ are set at 11, Fig. 10 shows four marked images, “Lena”, “Pepper”, “F16”, and “Boat”, with $p=13$, $q=13$, and $T=20$. By setting parameters $p$ and $q$ at 13, we can ensure that the overflow/underflow does not occur in any test image. Fig. 10 shows that these marked images are still similar enough to the original images that the human vision system is not conscious of any distortion caused by the embedding operation.

To further prove the layer-2 embedding procedure does not affect the PSNRs but increases the hiding capacity of marked images. Table 4 lists the PSNRs and hiding capacity of our proposed scheme with and without layer-2 based on $T=20$, $t_1=10$, $t_2=20$, and $p=q=13$. We set $p=q=13$ because it can ensure that no overflow and underflow occurs for each test image.
In Tian’s scheme, a compressed location map, the original LSBs stream, and the secret bits are concatenated and embedded into a cover image using his embedding strategies. Because the pure hiding capacity includes only the number of secret bits but omits the compressed location map and the original LSBs stream, the hiding capacity of Tian’s scheme is limited by the size of the location map and the original LSBs stream. In addition, not every pixel pair can be used for data embedding, and even an embeddable pixel pair can hide only one secret bit. Thus, the hiding capacity of Tian’s scheme is smaller than 0.5 bpp (bits per pixel) per single scan of the marked image rather than with the two scans required by Celik’s and Tian’s schemes.

Because the layer-2 embedding only hides the parameters, which are very small, Fig. 11 only compares the performance of Tian’s, Celik’s, and our proposed schemes without layer-2 in image quality of the marked image for test images “Lena” and “Pepper” with different hiding capacities. Based on the same image quality, the hiding capacity of our proposed scheme is larger than that can be achieved with Tian’s and Celik’s schemes, respectively. Even when the hiding capacity is 0.5 bpp, the PSNRs of the marked “Lena” and “Pepper” images with our proposed scheme are still as much as 41.47 dB and 40.42 dB, respectively, which is significantly higher than that can be obtained with Tian’s and Celik’s schemes, respectively.

5. Conclusions

This paper proposes a reversible data hiding scheme that is based on the relationships between pixels and their neighbors and requires only a few parameters to extract hidden data but requires no extra data to restore the original pixels. Our experimental results confirm that the proposed scheme’s hiding capacity is larger than that can be achieved with either Tian’s or Celik’s scheme. On average, the PSNR of the marked images provided by the proposed scheme is as much as 39.46 dB when achieving the largest hiding capacity, which is larger than 0.5 bpp. In addition, the proposed restoring procedure can be completed with a single scan of the marked image rather than with the two scans required by Celik’s and Tian’s schemes.

Although the proposed scheme provides larger hiding capacity than either Tian’s or Celik’s scheme while maintaining satisfactory image quality of marked images, the volume of any embedded message might still be increased. Therefore, in the future we hope to extend the proposed scheme to hide more embedded message data in each pixel and increase the volume of embeddable pixels in a cover image, thus increase hiding capacity while maintaining low distortion of marked images and requiring only a few parameters for data extraction and restoration of the original pixels.

References


Chia-Chen Lin (also known as Min-Hui Lin) received her Ph.D. degree in 1998 from National Chiao Tung University, HsinChu. Dr. Lin is now an associate professor with the Department of Computer Science and Information Management, Providence University, Taichung. She is currently doing research on subjects of image processing, data hiding, mobile agent, and electronic commerce.

Chia-Chen Chang received his B.S. degree in applied mathematics in 1977 and the M.S. degree in computer and decision sciences in 1979, both from the National Tsing Hua University, Hsinchu. He received his Ph.D. degree in computer engineering in 1982 from the National Chiao Tung University, Hsinchu. During the academic years of 1980 to 1983, he was with the Department of Computer Engineering at the National Chiao Tung University. From 1983 to 1989, he was with the Institute of Applied Mathematics, National Chung Hsing University Taichung. From August 1989 to July 1992, he was a professor and the Head of the Institute of Computer Science and Information Engineering at the National Chung Cheng University, Chiayi. From August 1992 to July 1995, he was the Dean of the College of Engineering at the same university. From August 1995 to October 1997, he was the Provost at the National Chung Cheng University. From September 1996 to October 1997, Dr. Chang was the Acting President at the National Chung Cheng University. From 2002 to 2005, he was a chair professor with National Chung Cheng University. Since February 2005, he has been a chair professor with Feng Chia University, Taichung. In addition, he has served as a consultant to several research institutes and government departments. His current research interests include database design, computer cryptography, image compression, and data structures.

Yi-Hui Chen received her Ph.D. degree in computer science and information engineering from the National Chung Cheng University. After that, she worked with Academia Sinica as a postdoctoral fellow. Later on, she served for IBM Taiwan Collaboratory Research Center as a research scientist. She is now an assistant professor with the Department of Applied Informatics and Multimedia, Asia University, Taichung. Her research interests include image processing, watermarking, steganography, and XML techniques.