A Novel LSB Data Hiding Scheme with the Lowest Distortion

Nan-I Wu
Department of Animation and Game Design System, TOKO University, Email: 100001@mail.toko.edu.tw
51 Sec. 2 Syuefu Rd., Puzih City, Chiayi County 61363, Taiwan

Min-Shiang Hwang
Department of Computer Science & Information Engineering Asia University
500, Lioufeng Rd., Wufeng, Taichung 41354, Taiwan
and
Department of Medical Research, China Medical University Hospital,
China Medical University, Taiwan
91 Xueshi Rd, North District, Taichung 404, Taiwan
Email: mshwang@asia.edu.tw

Abstract
In this paper, we present a novel LSB matching steganographic algorithm which provides a better stego image quality after embedding a secret message. The proposed scheme groups three pixels as an embedding unit to conceal three bits of secret data in which the possible modification for each pixel is at most +1 or −1. With our new scheme, the probability of altering one pixel by adding 1 or subtracting 1 is 6/8; it is 1/8 when altering two pixels, and 1/8 when preserving pixels without any modification. In conventional algorithms, the expected number of modification per pixel (ENMPP) mostly ranges from 0.375 to 0.5 on average when the embedding capacity of each pixel is one bit of secret message. In contrast, our new scheme groups three pixels together to convey three bits of secret message. The ENMPP is reduced at least by 0.333 on average. As a result, our algorithm offers higher visual quality for the stego image. The experimented results have verified that our new scheme is super to the previous algorithms.

Keywords: Data hiding, steganography, least-significant-bit substitution (LSBs), LSB matching revisited

1. Introduction
Due to the rapid advancement of digital technologies, the accessibility of Internet has made digital data exchange increasing in our common life. Hence, the security of the information exchanged on the public communication channels has becoming an important issue which is worth researching for.

While the traditional cryptography is able to protect the contents of the secret messages in transmission, the characteristic of the ciphertexts makes it an easy target for attackers. Malicious attacks frequently occur when the communication channels are monitored.

Data hiding/Steganography is a newly discovered data protection technique to protect the ciphertexts with the use of plaintexts. It is capable of hiding the existence of a secret message by utilizing an image, so it is implemented in covert communication. The existence of secret data in transmission is completely undetectable [1-4, 7].

The earliest concern of covert communication problems was first proposed by Simmons [10]. He sketches the prisoners problem: “Alice” and “Bob” are imprisoned in two separate places, and they plan to flee away together. All communication between them is, however, closely monitored by a prison
officer called “Wendy”. Under this circumstance, the traditional cryptography is ineffective since it will be easily detectable by Wendy. The only approach to secure their communication is to build a covert communication channel under the monitoring of “Wendy”. There, the transmission data has to be hidden inside a carrier (cover object), which often indicates a meaningful data (the plaintext), so as not to lose its generality.

To date, there is a variety of cover objects (e.g., images, videos, audios, and 3D models) that have been used as carriers to convey secret messages [2, 12-14, 17, 18, 19, 20, 21]. In this paper, we take a gray-level image as an example cover object to hide our secret data. An image before embedded with secret data is called a “cover image”, and “stego image” is the term we use to refer to the carrier (i.e. transmission media data) after the secret message is embedded.

There are two major concerns when selecting a gray-level image as the carrier to convey secret data. (1) The high tolerance of its pixel modification: the embedding capacity of a gray-level image is much higher than that of the other image formats, and the visual artifacts produced by the steganographic modification are hardly detectable by human eyes. Therefore, the payload and the stego image quality would be easier for users to control. In general, if the payload of each pixel for an image is less than 3 bpp (bits per pixel), the visual artifacts of the stego image would be undetectable by human eyes. (2) Its accessibility: Digital gray-level images are extremely common on the Internet, and they can be download easily and rapidly. The popularity and distribution of them makes their security important during the process of exchanging messages via Network.

Basically, we evaluate the performance of a data hiding scheme according to two metric. The first one is the metric of the perceptual distortion caused by data hiding. Since PSNR (Peak-Signal-to-Noise-Ratio) is popularly employed to evaluate the visual distortion of processed images and videos, we adopt PSNR to evaluate the perceptual distortion produced by our new scheme and the other existing schemes at the given embedding capacity. A higher PSNR value signifies that the visual distortion between the cover and the stego image is not able to be perceived by the human eyes. The second one is the metric of the embeddable payload. The embeddable capacity indicates the maximum amount of secret messages that can be embedded into the cover image without any noticable visual artifacts.

The earliest and most famous data hiding technique is the simple LSBs (Least-Significant-Bit substitution) method [3]. It utilizes the variation of the least significant bits of a pixel which is insusceptible to human visual sensitivity when concealing a secret message. The concept of the simple LSBs embedding algorithm is applicable for other kinds of multimedia objects. We will introduce the embedding algorithm of the simple LSBs method in detail later in Section 2-1. Ever since the simple LSBs method was proposed, there have been many other new methods presented to improve the performance of the LSBs method in terms of visual quality or embedding capacity [1, 2, 11,15, 16]. For example, Wang et al. developed an effective optimal rightmost k LSBs of the cover image based on a genetic algorithm in order to reduce the possible distortion caused by the simple LSBs method [15]. And to improve the computation time of Wang et al.’s scheme [15], Chang et al. used the dynamic
programming strategy so as to get the optimal solution [2]. However, the cost and computation complexity are heavy, and the visual quality improvement is slight (PSNR value merely raised 0.1~0.3 dB ) when applying either Wang et al.’s or Chang et al.’s scheme. Therefore, three optimal LSBs methods [1, 11, 16] were presented. It significantly improved the visual quality of the simple LSBs method with very low computation complexity. Specifically, the PSNR value can be raised at least 3 dB after hiding a secret data with the three optimal schemes [1, 11, 16] when the embedding rate of each pixel is 2, 3 or 4 bpp (bits per pixel). We will also introduce the embedding algorithm of the optimal LSBs scheme in detail later in Section 2-2. Nevertheless, the contribution of the optimal LSBs scheme becomes ineffective when the embedding rate is 1 bpp. Namely, the visual distortion produced by the simple optimal LSBs methods [1, 2, 11, 15, 16] all turns identical when the probability of bit modification for each pixel is 0.5. The low probability of bit modification can reduce the visual distortion after hiding a secret message. For the sake of reducing the expected number of modification per pixel (ENMPP), Mielikainen [9] presented an LSB matching revisited method by using a pixel pair to conceal two bits of a secret data based on a binary function. Mielikainen’s method can ensure the pixel to be altered at most +1 or −1 in which the ENMPP is reduced at 0.375. The +1 or −1 operator represents the LSB of a pixel when it is modified to carry two bits of a secret message. Since the probability of bit modification is reduced, the visual distortion after hiding data becomes less than that in Wang et al.’s or Chang et al.’s scheme [1, 2, 11, 15, 16].

However, in Mielikainen’s method there is no any analysis or discussion about how to group the pixels as an embedding unit when the number is bigger than two. In fact, the Mielikainen’s method can not be applied for embedding three bits of secret data into three pixels. The LSB matching revisited method is merely feasible in a paired pixels which are capable of reducing the possible distortion itself.

In this paper, we propose a new LSB data hiding scheme by grouping three pixels as an embedding unit to show that the embedding effect can minimize perceptual distortions to decrease the probability of bit modification. The proposed scheme alters at most one of three pixels by +1 or −1, so that three bits of secret data can be hidden into a group consisting of three pixels, and the expected number of modification per pixel can be reduced at 0.333. It is much more advanced than that of 0.375 produced by an LSB matching revisited scheme. Furthermore, we analyze and demonstrate that the ENMPP will increase from 0.333 to 0.343 when three or more pixels are grouped into an embedding unit. That is to say, among the state-of-the-art LSB matching schemes the proposed scheme is an optimal solution with the least embedding effect after hiding secret data when grouping three pixels as an embedding unit. The remainder of this paper is organized as follows: In Section 2, we introduce some related works, including the simple LSB method, the optimal LSB method and the LSB matching revisited method, respectively. The proposed new scheme which includes the mathematical theory analysis is presented in Section 3. The experimental results and comparisons with some other LSB embedding schemes are given in Section 4. Finally, the conclusion is in Section 5.

2 Related Works
This section contains three subsections. The simple LSB (Least-Significant-Bit) substitution scheme is described in Section 2-1. Then we describe the optimal LSB substitution which is able to reduce the distortion of the the simple LSB substitution in Section 2-2. Last, in section 2-3, we introduce the LSB matching revisited method which is able to improve the quality of the stego images produced by both the simple LSB substitution and the optimal LSB substitution when the embedding capacity is 1 bpp.

2-1 Simple LSBs method

The earliest application of the simple Least-Significant-Bit substitution (LSBs) scheme was specifically for grayscale images. The feature of this scheme is its simple embedding process and its accessibility to implement. It is in fact why it has gotten its name as the simple LSBs method. The simple LSBs scheme carries messages by directly replacing the LSB plane of gray-level pixel value. This is possible because the messages based on changing the LSB plane of pixel value is invisible to human visual sensitivity. Suppose that the decimal value of a grayscale pixel is \( G_{(10)} \), then we can convert \( G_{(10)} \) into eight bits of binary, i.e., \( G_{(10)}=d_8d_7d_6d_5d_4d_3d_2d_1_{(2)} \). For example, when \( G_{(10)}=28_{(10)} \), then its binary value is \( 00011100_{(2)} \). If we want to embed four bits of a secret message, such as \( 0010_{(2)} \), we can directly replace the most right and the least significant bits \( 1100_{(2)} \) of the pixel with the secret message \( 0010_{(2)} \). The original binary representation \( 00011100_{(2)} \) is, therefore, changed into \( 00010010_{(2)} \), and its decimal value is \( 18_{(10)} \).

In a similar manner, if \( n \) bits of secret message are to be hidden, we can utilize \( n \) least significant bits of a pixel to cover \( n \) bits of a secret message. Also, we can extract the embedded secret messages by reading \( n \) least significant bits of a pixel if we have acquired the knowledge of how many bits of secret messages are embedded into the LSB plane of a pixel. In this paper, 1-LSB indicates the capacity of each pixel is only one bit, and 2-LSB indicates the capacity of each pixel are two bits of secret messages; and so \( n \)-LSB represents the capacity of each pixel which hides \( n \) bits of secret messages.

2-2 Optimal LSBs method

Normally, the distortion of an image is quite large when we embed a large amount of secret messages using the simple LSB substitution scheme. For example, the PSNR of the stego image is lower than 40 dB when using 3-LSB embedding algorithm. In order to reduce the distortion of the simple LSBs scheme, the optimal LSBs schemes are proposed [1, 11, 16]. When we embed secret messages into one pixel of an image, the simple LSBs scheme only generates one stego pixel value. However, the optimal LSBs scheme is capable of generating three stego pixel values, one of them with the least distortion. We favor this stego pixel when carrying secret messages, because it is able to improve the quality of a stego image significantly.

Next, we take an example to explain the process of the optimal LSBs method. Suppose that the original pixel value \( G_{(10)}=28_{(10)} \) and four bits (i.e, \( n=4 \)) of secret messages \( 0010_{(2)} \) are to be embedded into the pixel \( G_{(10)} \), we would first calculate the remainder \( R \) of \( G_{(10)} \) by
\[ R = G_{(10)} \mod 2^n. \]

In this case, we obtain \( R = 12 \), and the secret message \( 0010_{(2)} \) is equal to \( 2_{(10)} \). We also adjust \( G_{(10)} \) until the remainder is equal to the secret message \( 2_{(10)} \). The messages are directly embedded to the most right four bits of the original pixel value \( 28_{(10)} = 000101100_{(2)} \). After that, we obtain a stego pixel value \( 18_{(10)} = 00010010_{(2)} \). We ensure the remainder value of \( 18_{(10)} \) by a mod operation which is equal to the decimale value of the secret message \((18_{(10)} \mod 2^4 = 2)\). Furthermore, we adjust the stego pixel value \( 18_{(10)} \) by \(+2^n\) and \(-2^n\) (i.e. \( 18_{(10)} + 2^n \) and \( 18_{(10)} - 2^n \), respectively. We do the same way as to obtain the other two stego pixel values which are \( 34_{(10)} \) \((34 \mod 2^4 = 2)\) and \( 2_{(10)} \) \((2 \mod 2^4 = 2)\) accordingly. In the end, we have three candidates of stego pixels which are \( 18_{(10)}, 34_{(10)} \) and \( 2_{(10)} \) to replace the original pixel \( 28_{(10)} \) since their remaining values are the same by \( \mod 2^n \) operation.

Obviously in this example, the candidate \( 34_{(10)} \) must be chosen since it has the least distortion (\( 34 - 28 = 6 \)) among the three candidates. However, \( 18_{(10)} \) would become the only option if we use the simple LSBs scheme, and its distance (i.e. pixel distortion) would become 10 (\( 28 - 18 = 10 \)). It is clear that the optimal LSBs scheme is able to reduce the distortion of the simple LSBs scheme tremendously.

We present the PSNR comparison between the simple LSBs scheme and the optimal LSBs scheme under different embedding capacities as shown in Table 1. By examining Table 1, it clearly shows that the optimal LSBs method can significantly reduce the distortion of the simple LSBs method by 2 to 3 dB PSNR in all cases except 1-LSB (namely, the embedding capacity of each pixel is one bit). In order to improve the distortion produced by the simple and optimal 1-LSB substitution scheme, Mielikainen presented an LSB matching revisited approach [9], with which the expected number of the modification per pixel (ENMPP) can be preserved.

Table 1: Comparison on visual distortion between the simple LSBs method and the optimal LSBs scheme.

<table>
<thead>
<tr>
<th>Cover image (512×512)</th>
<th>( n )-LSBs (Capacity: 512×512×( n ))</th>
<th>( PSNR ) (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Simple LSBs scheme</td>
<td>Optimal LSBs scheme</td>
</tr>
<tr>
<td>Lena</td>
<td>( n=1 )</td>
<td>51.12</td>
</tr>
<tr>
<td></td>
<td>( n=2 )</td>
<td>44.02</td>
</tr>
<tr>
<td></td>
<td>( n=3 )</td>
<td>37.86</td>
</tr>
<tr>
<td></td>
<td>( n=4 )</td>
<td>31.28</td>
</tr>
<tr>
<td>Baboon</td>
<td>( n=1 )</td>
<td>51.14</td>
</tr>
<tr>
<td></td>
<td>( n=2 )</td>
<td>44.02</td>
</tr>
<tr>
<td></td>
<td>( n=3 )</td>
<td>37.86</td>
</tr>
<tr>
<td></td>
<td>( n=4 )</td>
<td>31.33</td>
</tr>
</tbody>
</table>

### 2.3 LSB matching revisited method

No matter which scheme, the simple LSBs or the optimal LSBs scheme, an embedding unit consists only one pixel in the embedding process. That is to say, one bit of a secret message is hidden into one
The main concept of the LSB matching revisited approach is to divide a gray-level image into non-overlapped pixel pairs, and then utilizes a binary function to embed two bits into each pixel pair. The LSB matching revisited scheme can hide two bits into one pixel pair by changing the pixel +1 or −1 only. In the traditional data embedding process, we can at most add one bit to a pixel or substruct one bit from a pixel of a pixel pair after data hiding. Therefore, to change more than one pixel is nearly impossible. And since the probability of the pixel modification is now reduced, the expected number of the modification per pixel is also reduced from 0.5 to 0.375. This result shows that the visual distortion can be preserved after the data is hidden. The embedding algorithm of the LSB matching revisited scheme is given as follows:

Define \( m_i \) and \( m_{i+1} \) are two bits of a secret message, and \( y_i \) and \( y_{i+1} \) are the cover pixel pairs, respectively. After data hiding, the bit \( m_i \) is hidden into the LSB plane of the stego pixel \( y_i \), and the bit \( m_{i+1} \) is hidden into the function of a pixel pair \( y_i \) and \( y_{i+1} \). The equation of the binary function is given below:

\[
\begin{align*}
\text{LSB}(y_i) &= m_i, \\
m_{i+1} &= f(y_i, y_{i+1}) = \text{LSB}\left(\frac{y_i}{2} + y_{i+1}\right)
\end{align*}
\]

Herein, we take an example to illustrate the operation of the LSB matching revisited approach. Suppose that the decimal values of an original pixel pair are 100 \(_{(10)}\) and 51 \(_{(10)}\), respectively, we then convert them into the binary 01100100 \(_{(2)}\) and 00110011 \(_{(2)}\). And suppose that two bits of secret messages are \( m_0=0 \) and \( m_{i}=0 \), respectively, we then hide \((m_0=0, m_{i}=0)\) into \((100, 51)\) with the above binary function. In the end, we learn that no matter how we change the number from 51, 50 to 52, the number 100 does not change. If \( m_0=0 \) and \( m_1=1 \), the two pixel values do not need any modification. If \( m_0=1 \) and \( m_1=0 \), we then need to change 100 into 99, and the number 51 does not change. And if \( m_0=1 \) and \( m_1=1 \), we then need to change 100 into 101, and the number 51 does not change. Table 2 shows all the cases for an embedding unit by the LSB matching revisited.

<table>
<thead>
<tr>
<th>Original Pixel Pair</th>
<th>Secret Message ((m_i, m_{i+1}))</th>
<th>Pixel Variation</th>
</tr>
</thead>
<tbody>
<tr>
<td>( f(100,51) )</td>
<td>(0,0)</td>
<td>( f(100,51-1) ) or ( f(100,51+1) )</td>
</tr>
<tr>
<td>( f(100,51) )</td>
<td>(0,1)</td>
<td>No modify</td>
</tr>
<tr>
<td>( f(100,51) )</td>
<td>(1,0)</td>
<td>( f(100-1,51) )</td>
</tr>
<tr>
<td>( f(100,51) )</td>
<td>(1,1)</td>
<td>( f(100+1,51) )</td>
</tr>
</tbody>
</table>

As seen from the example above, one of the most important factors of the LSB matching revisited approach is that maximally, there can only be one pixel at a time to be modified by +1 or −1 when...
carrying two bits of secret messages. The situation of changing both two pixels of a pair at the same time does not exist. Thus, the probability of a pixel pair that needs to be modified is $3/4$, and the probability of the preserved original pixel pair is $1/4$. The expected number of the modification per pixel (ENMPP) of a pixel pair produced by LSB matching revisited is $\frac{1}{4}\times 0 + \frac{3}{4}\times 1 = \frac{3}{4}$. We can further calculate the ENMPP of each pixel with $(3/4)/2 \approx 0.375$ on average.

Consequently, the LSB matching revisited approach indeed improves the performance of the simple and the optimal LSBs scheme in view of the ENMPP. Table 3 shows the experimental results of the LSB matching revisited approach compared to the simple LSBs and the optimal LSBs approach. As shown in Table 3, the PSNR of the LSB matching revisited is actually better than that of the simple LSB and the optimal LSB scheme.

### Table 3: PSNR comparison among the LSB matching revisited, simple and optimal LSBs schemes.

<table>
<thead>
<tr>
<th>Cover image (512x512)</th>
<th>Capacity (bits)</th>
<th>PSNR (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Simple LSB</td>
<td>Optimal LSB</td>
</tr>
<tr>
<td>Lena</td>
<td>262144</td>
<td>51.12</td>
</tr>
<tr>
<td>Baboon</td>
<td>262144</td>
<td>51.14</td>
</tr>
</tbody>
</table>

#### 3. The proposed scheme

In this section, we propose a new algorithm to improve the performance of the LSB matching revisited by grouping three pixels as an embedding unit instead of a pixel pair. We utilize the $+1$ or $-1$ operator to control the second least significant bit when recording the status of a secret message. Six least significant bits are assembled by the XOR operation in order to yield three original bits from the embedding unit. We present a novel embedding algorithm to ensure that at most one pixel can be altered by $+1$ or $-1$. That is, the maximum variation of the pixel value before and after hiding secret data is limited to $+1$ or $-1$ pixel difference value. With the proposed scheme, we also discovered that using the bit modification to show an embedding unit which consists three pixels is the best optimal combination in terms of quality degradation. Now we introduce our new scheme in detail.

We define the three pixels as $p_i$, $p_{i+1}$, $p_{i+2}$, and group them into an embedding unit. We also choose three bits of secret data, $m_0$, $m_1$, $m_2$, and hide them into $p_i$, $p_{i+1}$, $p_{i+2}$, respectively. The embedding algorithms are described as below:

**Step 1:** Convert the decimal value $p_i$, $p_{i+1}$, $p_{i+2}$ into the binary value, which is $p_i(10) = a_5 a_7 a_6 a_5 a_4 a_3 a_2 a_1(2)$, $p_{i+1}(10) = b_3 b_5 b_4 b_3 b_2 b_1(2)$, and $p_{i+2}(10) = c_3 c_5 c_4 c_3 c_2 c_1(2)$, respectively. Let $a_1$ be the first most right bit of $p_i$, and $a_2$ be the second most right bit of $p_i$, respectively. In the same manner, we define $b_1$ and $c_1$ as the first most right bit of $p_{i+1}$ and $p_{i+2}$, respectively, and define $b_2$ and $c_2$ as the second most right bit of $p_{i+1}$ and $p_{i+2}$, respectively. We will control the variation of the first and second most right bits by modifying three pixels by $+1$ or $-1$. For example, when $p_i = 10(10) = a_5 a_7 a_6 a_3 a_4 a_3 a_2 a_1(2) = 000001010(2)$, $a_1 = 1$ with $p_i +1$ or $p_i -1$. In this case, however, if $a_2$ is 0, we can only modify $p_i$ by $-1$. In our new scheme, the original feature of a group which consists three pixels can be derived from the XOR operation. Also, since we define bits $A$, $B$ and $C$ as the
original feature values of an embedding unit, they will be changed into three bits of the secret data after applying the proposed embedding algorithm. Specifically, the significance of bits $A$, $B$ and $C$ will be used to conceal the status of a secret message in the form of three binary bits. In Step 2, the rule of generating bits $A$, $B$, and $C$ is described in detail.

**Step 2:** According to Eq. (1), compute bits $A$, $B$, and $C$ by XORing six most right significant bits.

$$
\begin{align*}
A &= a_1 \oplus a_2 \oplus b_1 \\
B &= b_1 \oplus b_2 \oplus c_1 \\
C &= c_1 \oplus c_2 \oplus a_1
\end{align*}
$$

As shown in Eq. (1), we can simultaneously control bits $A$ and $C$ by changing the least significant bit $a_i$ of $p_i$. In the same manner, bit $A$ can be controlled by changing the second least significant bit $a_2$ of $p_i$. Specifically, when the pixel value $p_i$ is an odd number, we only need to control it by modifying bit $a_i$ by $p_i - 1$. And when the pixel value $p_i$ is an even number, we also only need to control it by modifying bit $a_i$ by $p_i + 1$. In a similar manner, we only need to control bit $a_2$ by $p_i + 1$ if the pixel value $p_i$ is an even number. On the contrary, if pixel value $p_i$ is an odd number, we only need to control bit $a_2$ by $p_i - 1$. Next, we can simultaneously modify the status of bits $A$ and $B$ by changing the least significant bit $b_1$ of $p_i$. We also alter the second least significant bit $b_2$ of $p_i$ for the sake of controlling bit $B$. Finally, bits $B$ and $C$ can be simultaneously changed by altering the least significant bit $c_1$ of $p_i$. And bit $C$ can be changed by modifying the second least significant bit $c_2$ of $p_i$.

**Step 3:** We compare the extracted original feature value bits $A$, $B$, $C$ with three bits of the secret message $m_0$, $m_1$, $m_2$ to see whether they are the same or not. If the condition $(A, B, C) = (m_0, m_1, m_2)$ is satisfied, the pixels $p_i$, $p_{i+1}$, $p_{i+2}$ are unnecessary to be altered in the data embedding process. Otherwise, we must modify pixels $p_i$, $p_{i+1}$ or $p_{i+2}$ until the condition $(A, B, C) = (m_0, m_1, m_2)$ is satisfied. We propose the modification algorithm to ensure that the variation of pixels $p_i$, $p_{i+1}$, $p_{i+2}$ is restricted to $\pm 1$. In the following, we describe the algorithm in detail.

If $(A = m_0)$ & $(B = m_1)$ & $(C = m_2)$

No Modify

ElseIf $(A \neq m_0)$ & $(B = m_1)$ & $(C = m_2)$

If $p_{i+1} \mod 2 = 0$ Then $p_{i+1} = p_{i+1} - 1$

Else $p_{i+1} = p_{i+1} + 1$

End

ElseIf $(A = m_0)$ & $(B \neq m_1)$ & $(C = m_2)$

If $p_{i+2} \mod 2 = 0$ Then $p_{i+2} = p_{i+2} - 1$

Else $p_{i+2} = p_{i+2} + 1$

End

ElseIf $(A = m_0)$ & $(B = m_1)$ & $(C \neq m_2)$

If $p_i \mod 2 = 0$ Then $p_i = p_i - 1$

Else $p_i = p_i + 1$
According to the algorithm above, three bits of the secret data $m_0, m_1, m_2$ are ensured to be embedded into the three pixels $p_i, p_{i+1}, p_{i+2}$, respectively. After the stego image is received, the hidden message can be extracted with Eq. (1), without any knowledge of the cover image information.

For example, as shown in Table 4, when $p_i=51, p_{i+1}=99$ and $p_{i+2}=33$, we have $a_1=1, a_2=1, b_1=1, b_2=1, c_1=1$ and $c_2=0$. Calculate the original feature bit $A$ via $A = a_1 \oplus a_2 \oplus b_1$ based on Eq. (1), and we obtain bit $A=1$. Next, we can also derive bit $B=1$ via $B = b_1 \oplus b_2 \oplus c_1$ based on Eq. (1). Finally, the original feature bit $C=0$ can be computed by Eq. (1). If the status of the secret data is $m_0m_1m_2=110_2$, it is unnecessary to alter any pixels among pixels $p_i, p_{i+1}, p_{i+2}$ since $m_0m_1m_2=ABC=110_2$ is satisfied. Otherwise, we must alter one or two pixels by $+1$ or $-1$ until $m_0m_1m_2=ABC$ is satisfied. Table 4 shows seven examples of altering the pixels so that bit $A, B, C$ are to be identical to $m_0m_1m_2$.

### 3.1. Analysis

The example in Table 4 shows that in hiding secret data, there is only 1/8 of the probability when altering two pixels simultaneously if the modification variation of the cover pixels is restrained to $+1$ or
−1. The probability from modifying one pixel +1 or −1 for hiding a secret data is at most 6/8. And three pixels kept unaltered after data hiding has 1/8 probability. It is clear that with the proposed scheme, the total pixel variation rate for the steganographic modification of an embedding unit is \((1/8)\times2\times1+(1/8)\times0\times1+(1/8)\times1\times6=1\). The expected number of modification per pixel (ENMPP) is \(1/3=0.333\) in average. This result demonstrates that the proposed scheme effectively prevents the pixel distortion after the data hiding. It reduces the ENMPP of the LSB matching revisited [9] from 0.375 to 0.333.

Table 4: An example to illustrate how to embed three bits \(m_0, m_1, m_2\) into a group that consists three pixels \(p_i, p_{i+1}, p_{i+2}\) by changing at most +1 or −1.

<table>
<thead>
<tr>
<th>Status of Secret Data</th>
<th>Three cover pixels (p_i=51, p_{i+1}=99, p_{i+2}=33). The original feature (A=1, B=1, C=0).</th>
</tr>
</thead>
<tbody>
<tr>
<td>(m_i,m_j,m_k=000_2)</td>
<td>(p_i=51) (p_{i+1}=99) (p_{i+2}=33).</td>
</tr>
<tr>
<td>(m_i,m_j,m_k=001_2)</td>
<td>(51−1=50) (99) (33−1=32)</td>
</tr>
<tr>
<td>(m_i,m_j,m_k=010_2)</td>
<td>(51) (99+1=100) (33)</td>
</tr>
<tr>
<td>(m_i,m_j,m_k=011_2)</td>
<td>(51−1=50) (99) (33−1=32)</td>
</tr>
<tr>
<td>(m_i,m_j,m_k=100_2)</td>
<td>(51) (99) (33)</td>
</tr>
<tr>
<td>(m_i,m_j,m_k=101_2)</td>
<td>(51) (99) (33−1=32)</td>
</tr>
<tr>
<td>(m_i,m_j,m_k=110_2)</td>
<td>(51) (99) (33)</td>
</tr>
<tr>
<td>(m_i,m_j,m_k=111_2)</td>
<td>(51+1=52) (99) (33)</td>
</tr>
</tbody>
</table>

The proposed scheme can record the status by modifying one pixel +1 or −1 operation, and it can also record the status while two pixels are modified simultaneously by +1 or −1. In the same way, we can obtain the status when three pixels are modified at the same time. Without any loss of generality, a great number of statuses can be obtained when more than one pixel is modified simultaneously. However, it does also result in greater visual distortion after the secret message hiding process. Therefore, we only analyze the condition by modifying one pixel or two pixels simultaneously when an embedding unit consists more than 3 pixels. We also demonstrate the optimal solution when the proposed scheme groups three pixels into an embedding unit, as well as how it is capable of preventing the image distortion after the data hiding.

Now, we investigate the lower bound of the proposed scheme, which represents the least modification to an embedding unit. First, given an embedding unit which consists \(n\) pixels, we have \(2n=2C^n_1\) statuses if we only modify one pixel by +1 or −1. The original status can be obtained via Eq. (1), which is \(C^n_0=1\), to indicate an embedding unit with the secret data without any modification. On the other hand, for an embedding unit which consists \(n\) pixels, there are at least \(2n+1\) statuses to be utilized when hiding secret data, during which one of \(n\) pixel is to be altered by +1 or −1. If we want to hide \(n\) bits into \(n\) pixels, there are at least \(2^n\) statuses that need to be provided so that an embedding unit contributing to generate \(2^n\) statuses can be modified (+1/−1/or remain unaltered) to carry \(n\) bits of the secret messages. Accordingly, the condition \(2n+1>2^n\) is dissatisfied when \(n=3\) since it lacks one status.
To overcome this problem, we must modify two pixels simultaneously to replenish one status. Based on these assumptions, the expected number of the modification of an embedding unit which consists
3≤n≤4 pixels can be expressed as

$$P = \frac{1}{2^n} \times 0 \times 1 + \frac{1}{2^n} \times 1 \times 2n + \frac{1}{2^n} \times 2 \times \left[2^n - (2n + 1)\right]$$

(2)

Herein, the ENMPP is larger than 0.333 when n=4 pixels are grouped into an embedding unit. When 4 pixels are selected to hide 4 bits of secret messages, the probability of the remaining original pixels is 1/16, and the probability of modifying one of the pixels by +1 or −1 is 8/16. The other statuses can be surely obtained by modifying two pixels simultaneously, and its probability is 7/16. Therefore, we can derive the expected number of the modification of an embedding unit which consists 4 pixels by

$$\frac{1}{2^4} \times 0 \times 1 + \frac{1}{2^4} \times 1 \times 8 + \frac{1}{2^4} \times 2 \times \left[2^4 - (8 + 1)\right] = \frac{22}{16} = 1.375$$

The expected number of the modification is 1.375/4=0.34375 per pixel on average. This demonstrates that the ENMPP of four pixels combination is larger than that of three pixels combination. It is expected that the ENMPP will be larger than 0.34375 when five pixels are grouped into an embedding unit since there are at least three pixels to be modified simultaneously in order to produce all statuses to hide secret messages.

4. Experimental Results and Comparisons

To evaluate the performance of our proposed LSB data hiding algorithm, we take two hundred cover images in this experiment. Herein, we present the experimental results with three typical images of 512×512 pixels size; “Lena” with moderate image complexity, “Baboon” with high-texture characteristic and “Jets” with smoother characteristic as shown in Fig. 1. According to the proposed embedding algorithm, we group three pixels as an embedding unit to conceal three bits of secret message. The stream of the secret message is produced by a pseudo-random generator where the probability distribution of bit ‘1’ and ‘0’ is identical. We use PSNR metric to evaluate the visual quality degradation of the stego image produced by the proposed scheme and the LSB matching revisited scheme. In these experiments, the high PSNR value indicates that the visual artifacts of the stego image is imperceptible to human visual sensitivity, and vice versa when producing more distortion after the secret data is hidden.

From Fig. 1, we can see that the visual artifacts between the cover image and the stego image are nearly invisible to human eyes. The embedding capacity of the stego image in Figs. 1(b)(d)(f) is floor{((512×512)/3)×3}=262143 bits. The PSNR values shown in Fig. 1 are average values when running 1000 times with the proposed scheme and the pseudo-random generator. Therefore, the PSNR values of Fig. 1. is not only accurate but objective, which shows the outstanding performance of the proposed scheme.

As for the imperceptibility among different LSB embedding schemes, we compare the proposed scheme with the existing LSBs schemes [1, 6, 8, 9, 11, 16]. Table 5 shows the PSNRs companion results under the same embedding capacity. A well-known LSB matching revisited algorithm proposed by
Mielikainen[9] was compared with the previous related LSB scheme in terms of visual quality degradation. Table 5 reveals that visual metric $PSNR$ of LSB scheme [1, 6, 8, 11, 16] is increased up to 1.247 (dB), and improved by Mielikainen’s scheme[9]. However, our new scheme can improve the performance of Mielikainen’s scheme by increasing the $PSNR$ up to 0.516 (dB). This is indeed a significant improvement as the performance of our new scheme is extremely optimal with the most minimal distortion among all LSB-based schemes. As mentioned in Section 3.1, our analysis shows that an embedding unit with a group which consists three pixels is a better optimal combination than that of a group which consists four pixels.

![Lena Cover image](image1)
![Lena Stego image](image2) $PSNR=52.916$ dB

![Baboon Cover image](image3)
![Baboon Stego image](image4) $PSNR=52.917$ dB

![Jets Cover image](image5)
![Jets Stego image](image6) $PSNR=52.925$ dB

Figure 1: (a)(c)(e) Cover images of sized $512 \times 512$, (b)(d)(f) stego images produced by the proposed scheme.
Table 5: Comparison between our new scheme and some other existing methods [1, 6, 8, 9, 11, 16].

<table>
<thead>
<tr>
<th>Cover Image</th>
<th>PSNR</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LSB-based</td>
<td>LSB matching</td>
<td>The proposed</td>
</tr>
<tr>
<td>Image</td>
<td>scheme [1,6,8,11,16]</td>
<td>revisited scheme [9]</td>
<td>scheme</td>
</tr>
<tr>
<td>Lena (512×512)</td>
<td>51.156 dB</td>
<td>52.404 dB</td>
<td>52.916 dB</td>
</tr>
<tr>
<td>Jets (512×512)</td>
<td>51.143 dB</td>
<td>52.4 dB</td>
<td>52.925 dB</td>
</tr>
<tr>
<td>Baboon (512×512)</td>
<td>51.161 dB</td>
<td>52.405 dB</td>
<td>52.917 dB</td>
</tr>
<tr>
<td>Elaine (512×512)</td>
<td>51.17 dB</td>
<td>52.407 dB</td>
<td>52.914 dB</td>
</tr>
<tr>
<td>Man (512×512)</td>
<td>51.138 dB</td>
<td>52.39 dB</td>
<td>52.911 dB</td>
</tr>
<tr>
<td>Average</td>
<td>51.154 dB</td>
<td>52.401 dB</td>
<td>52.917 dB</td>
</tr>
</tbody>
</table>

Now we compare the computational complexity and the overhead in the data embedding process. First, the schemes [1, 6, 8, 11, 16] need at least one logistic operation, and the total cost of the computational complexity is $O(n)$ when hiding $n$ bits into $n$ pixels. In Mielikainen’s scheme [9], there are four logistic operations which need to perform for each embedding process in which the average cost of one pixel is $O(2^n)$; that is to say, it is equal to $O(n)$. On the other hand, the proposed scheme needs 8 logistic operations each time in order to perform in the secret data hiding process. The average cost of one pixel is $O(2.66^n)$, which equals $O(n)$. Based on the discussion above, the total cost of the computational complexity with the proposed scheme is equal to the existing schemes [1, 9, 11, 16], as all the secret data embedding/extracting steps of the proposed scheme are performed on a linear operation.

5. Conclusions

In this paper, we propose a novel LSB data hiding scheme which is capable of minimizing the image quality distortion produced by the LSB matching revisited scheme. The proposed scheme groups three pixels into an embedding unit to convey three bits of secret message data and replaces a pixel pair used in the LSB matching revisited scheme. For each embedding unit, we extract two most right bits of a pixel to combine six bits in order to introduce the original status of secret data in a binary form of 3 bits by using the XOR operation. In the embedding algorithm, there are only 6/8 probability required to alter one pixel by adding 1 or subtracting 1, and 1/8 probability required to alter two pixels by adding 1 or subtracting 1 simultaneously; and it does not alter any pixel. The proposed scheme shows that the expected number of the modification per pixel can be improved. In addition, we also prove that an
embedding unit with three pixels is the most optimal combination which is capable of minimizing the total visual distortion resulted from the embedding effect.

Reference


