Optimal Gaussian Weight Predictor and Sorting Using Genetic Algorithm for Reversible Watermarking Based on PEE and HS

Chaiyaporn PANYINDEE†[a], Nonmember and Chuchart PINTAVIROOJ†[b], Member

SUMMARY This paper introduces a reversible watermarking algorithm that exploits an adaptable predictor and sorting parameter customized for each image and each payload. Our proposed method relies on a well-known prediction-error expansion (PEE) technique. Using small PE values and a harmonious PE sorting parameter greatly decreases image distortion. In order to exploit adaptable tools, Gaussian weight predictor and expanded variance mean (EVM) are used as parameters in this work. A genetic algorithm is also introduced to optimize all parameters and produce the best results possible. Our results show an improvement in image quality when compared with previous conventional works.

key words: prediction-error expansion (PEE), histogram shifting (HS), reversible watermarking, Gaussian weight predictor, expanded variance mean (EVM)

1. Introduction

Watermarking techniques have been improved over the past 20 years, making it possible to embed watermarks into multimedia data so as to trace ownership or copyright protection. For some applications, such as medical images and military images, it is unacceptable for the watermarking technique to permanently distort the host image. In these cases, the original image must be recovered without any distortion after extracting the embedded watermark. The watermarking techniques that satisfy these requirements are referred to as reversible watermarking.

In earlier works, modulo-arithmetic-based additive spread-spectrum techniques were used by Macq [1] and Honsinger et al. [2]. However, using arithmetic modulation caused salt-and-pepper artifacts. Vleeschouwer et al. [3] presented a method to decrease artifacts by using histogram transformation of the circulation interpretation of bijective transformation. In 2003, Jun Tian [4] introduced an effective technique called difference expansion (DE), which is a modified high-pass and low-pass filter and bit-shifting process. This technique can embed data into an image and preserve the original image information. This technique relies on a lossless compression tool to reduce the size of hidden data in order to decrease image distortion, as done in previous work [14]–[16].

In the following year, Thodi and Rodriguez [5] presented a technique called prediction-error expansion, which improved the DE expansion in Tian’s work. They achieved this by using prediction error (PE) instead of the difference of two connected pixels, which produces less error and decreases image degradation significantly. Their technique is to expand the PE whose predicted value is calculated from the predictor; therefore, this technique can be improved by utilizing a high performance predictor such as the median edge detector predictor (MED) [5]–[8], the gradient-adjusted predictor (GAP) [23], the rhombus predictor [9], [10], [19] and improved rhombus predictor [22], partial differential equation (PDE) predictor [11], Gaussian weight predictor [12], [13], or non-uniform weight predictor [24]. Several researchers had tried to increase the prediction function efficiency as in ShaoWei et al.’s work’s [17]. However, increasing the prediction function efficiency did not improve the overall result of the algorithm because other components of the algorithm also needed modification.

A sorting technique was introduced in 2005 by Kamstra and Heijman [18]. This technique improved DE expansion, produced smaller size location maps and decreased the distortion of watermarked images. Their idea utilized efficient correlation between neighboring pixels to sort data in the image. In 2007, Thodi and Rodriguez [6] introduced another technique called histogram shifting (HS). It was designed to decrease image distortion by shifting the unused pixels instead of expanding them. Moreover, changing the proportion between the numbers “0” and “1” in a location map would produce a more effective compression tool and also decreases the size of location map. An effective method of data embedding seen in many recent works is to combine PEE with HS [7]. There are other different novel techniques such as using infinity norm rotation [20] to transform data into another domain for embedding data or using fast transformation techniques such as in Coltuc et al.’s work [21].

An important improvement of the method is Sachnev et al.’s work [19] which combines several techniques including prediction-error expansion, histogram shifting, sorting, two-pass testing, double embedding and appropriate threshold values. One alternate prediction scheme, used exclusively for sorting algorithms, was utilized in their work for achieving increased embedding capacity; the rhombus prediction scheme allows sorting to achieve a better performance. Another modification of their techniques is referred to as the two-pass testing which was used for testing each pixel before using the histogram shift method for embedding. This
method remarkably reduces the size of the location map. By this process, they have achieved the best performance reported so far. Li et al. [23] proposed two improvements. The first improvement is to embed two bits into smooth locations and one bit into rough locations. This method provides a better performance when the payload is high. The second improvement is pixel selection technique, which is similar to the sorting technique. Kotvicha et al. [25] also achieved better results by using expanded variance mean (EVM) method to sort data.

Although the rhombus prediction and sorting technique (i.e., local variance) as previously mentioned in [19] is outstanding in terms of reducing distortion and enhancing the embedding capacity, it was designed to use a fixed predictor and sorting configuration for every image and every payload size. Because of these limitations, Panyindee and Pintavirooj [12], [13] introduced the new modification scheme by adding a new Gaussian weighted predictor and the sorting technique using EVM. In their Gaussian weighted predictor, the Gaussian shape can be varied in two directions, namely x and y directions, in order to fit the watermarked image. Their works had shown that it was more efficiency in predicting pixels. Moreover, they also applied genetic algorithm to optimize Gaussian parameters for two-dimensional predictor and set of sorting pixel sequence, which will diminish PE value. Note that the scope of prediction, \( \varphi_p \), will not only affect with PSNR value, but also the running time. In [13], Panyindee and Pintavirooj proposed the advanced methods based on Gaussian weighted predictor. They used the maximum level of \( \varphi_p \) at fourth level in standard image, i.e. Barbara and Boat, which was sufficient to achieve the promising results and reduce running times.

Even though reducing the scope of \( \varphi_p \) in some standard images allows the number of operator to decrease, which in turn affects the processing time. However, in order to achieve the best possible PSNR, an application in other standard images still has to be explored. This paper is hence the continued research work of [12] and [13] in which the scope of prediction, \( \varphi_p \), will be further investigated in various types of images, including images with low to high frequency. We concluded that the prediction with \( \varphi_p \) at sixth level is the best size for standard images, where the \( \varphi_p \)-prediction at fourth level [13] is suitable for high-variation images.

The rest of the paper is organized as follows. Section 2 briefly describes prediction-error expansion. Section 3 introduces Gaussian weighted predictor. In Sects. 4, 5 and 6, we explain data sorting, histogram shifting and two-pass testing, respectively. Explanation of optimization using the genetic algorithm, threshold optimization and encoding and decoding algorithm are given in Sects. 7, 8 and 9 respectively and Sect. 10 provided experimental results. Finally, our conclusions are found in Sect. 11.

2. Prediction Error Expansion

Prediction-error expansion is a technique to conceal data within prediction error. PE uses data from neighboring pixels to calculate predicted values. Let \( d_{i,j} \) be PE at position \((i, j)\), \( u_{i,j} \) be an original pixel’s value and \( u_i' \) be a predicting value. PE can be calculated by \( d_{i,j} = u_{i,j} - u_i' \). Bit shifting, also known as PE expansion, embeds watermark data into the last bit of expanded PE values in order to preserve original image information and allows encoder to recover watermark extracted image. Expanded PE value, \( D_{i,j} \), is defined by, \( D_{i,j} = 2d_{i,j} + b \), where \( b \) is a data to be hidden. The pixel’s modified value is \( U_{i,j} = u_i' + D_{i,j} \). Extracting the watermark and recovering the original image can be done by \( b = D_{i,j} \mod 2 \) and \( u_i' = u_i' + [D_{i,j}/2] \), where [ ] is the floor function. The predicted function has a major influence on every components of the reversible watermarking algorithm. The lower the PE is the lower the image distortion. Therefore, the prediction value should be close to the originals. Sachnev et al. [19] selected local mean to be their prediction value. Their process starts by separating pixels in an image into two sets called “Cross set” and “Dot set” as seen in Fig. 1. All cross set pixels are predicted using data from the dot set. Each pixel is predicted using four neighboring pixels from the dot set using the equation \( u_i' = [(u_{i+1,i+1} + v_{i+1,i-1} + v_{i+2,i+1})/4] \). When data from the cross set is adjusted and all values in the dot set remains unchanged, the decoder will then calculates exactly the same prediction value as the encoder.

3. Gaussian Weighted Predictor [12], [13]

A Gaussian function is an ideal type of function for determining the weight of a predictor because its shape produces extreme value at the middle and decline when farther away from the center. The shape of Gaussian function depends on only two parameters, standard deviation (SD) in x and y directions, denote as \( \sigma_x \) and \( \sigma_y \) respectively. The Gaussian weight function can be represented by

\[
w(x, y) = e^{-(x^2/2\sigma_x^2 + y^2/2\sigma_y^2)}.
\]

An adaptive predicting function is

\[
u_i' = \frac{\sum_{k=1}^{\varphi_p} \sum_{l=-\varphi_p}^{\varphi_p} w(l, l + 2k - 1)\psi_p(l+i+l+2k-1)}{\sum_{k=1}^{\varphi_p} \sum_{l=-\varphi_p}^{\varphi_p} w(l, l + 2k - 1)}.
\]

Fig. 1 Cross and dot set in an image.
Fig. 2  Gaussian shape weight for four test images with different payload sizes. From top down are Lena, Baboon, Peppers, and Sailboat. From left to right are 0.038, 0.153, 0.267, 0.381, 0.496, 0.610, 0.725 and 0.839 bpp payload.

Table 1  Comparisons for the number of the PEs \((d_{i,j})\) between the proposed predictor and the four-neighbor predictor of Sachnev et al. [19], payload = 40,000 bits, for six standard 512 × 512 sized grayscale images.

<table>
<thead>
<tr>
<th>Predictor type</th>
<th>PE value ((d_{i,j}))</th>
<th>Lena</th>
<th>Barbara</th>
<th>Baboon</th>
<th>Boat</th>
<th>Peppers</th>
<th>Sailboat</th>
</tr>
</thead>
<tbody>
<tr>
<td>Four-neighbor predictor</td>
<td>-2</td>
<td>25050</td>
<td>20284</td>
<td>9396</td>
<td>17552</td>
<td>18957</td>
<td>14868</td>
</tr>
<tr>
<td></td>
<td>-1</td>
<td>32106</td>
<td>24808</td>
<td>10727</td>
<td>19245</td>
<td>20576</td>
<td>17552</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>33132</td>
<td>25386</td>
<td>10727</td>
<td>19838</td>
<td>20974</td>
<td>19284</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>28806</td>
<td>22568</td>
<td>10646</td>
<td>18452</td>
<td>20073</td>
<td>17706</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>22408</td>
<td>19084</td>
<td>9132</td>
<td>15968</td>
<td>18340</td>
<td>14310</td>
</tr>
<tr>
<td>Total</td>
<td>141502</td>
<td>112130</td>
<td>50637</td>
<td>90855</td>
<td>98920</td>
<td>85320</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Gaussian weight predictor ((\sigma_x, \sigma_y))</th>
<th>(d_{i,j})</th>
<th>(\sigma_x=1.575)</th>
<th>(\sigma_x=0.707)</th>
<th>(\sigma_y=0.846)</th>
<th>(\sigma_y=0.313)</th>
<th>(\sigma_y=1.617)</th>
<th>(\sigma_y=0.622)</th>
</tr>
</thead>
<tbody>
<tr>
<td>-2</td>
<td>26500</td>
<td>21284</td>
<td>10982</td>
<td>19830</td>
<td>20252</td>
<td>16755</td>
<td></td>
</tr>
<tr>
<td>-1</td>
<td>33346</td>
<td>25554</td>
<td>11080</td>
<td>21938</td>
<td>21720</td>
<td>19470</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>35816</td>
<td>27550</td>
<td>11394</td>
<td>22400</td>
<td>21800</td>
<td>20004</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>31010</td>
<td>25082</td>
<td>11382</td>
<td>21554</td>
<td>21152</td>
<td>17351</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>23408</td>
<td>20084</td>
<td>10728</td>
<td>19038</td>
<td>19108</td>
<td>14752</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>149630</td>
<td>119554</td>
<td>55566</td>
<td>104560</td>
<td>104032</td>
<td>88332</td>
<td></td>
</tr>
</tbody>
</table>

Where \((i, j) \in A_1 \times A_2 \cup A_3 \times A_4\) for a cross set and \((i, j) \in A_1 \times A_2 \cup A_3 \times A_4\) for a dot set. \([\bullet]\) is a ceiling function and \(A_1, A_2, A_3, A_4\) are defined by

\[ A_1 = \{ \varphi_p + 2n - 1 | n = 1, 2, \ldots, (N - (N + 1) \mod 2 + 1) / 2 - \varphi_p \} \]

\[ A_2 = \{ \varphi_p + 2n - 1 | n = 1, 2, \ldots, (M - (M + 1) \mod 2 + 1) / 2 - \varphi_p \} \]

\[ A_3 = \{ \varphi_p + 2n | n = 1, 2, \ldots, (N - N \mod 2) / 2 - \varphi_p \} \]

\[ A_4 = \{ \varphi_p + 2n | n = 1, 2, \ldots, (M - M \mod 2) / 2 - \varphi_p \} \]

\(N\) is the number of rows, and \(M\) is the number of columns in an image. \(\varphi_p\) is a level of expansion for a predictor.

Data sorting is another tool used to decrease the distortion of watermarked images and produce smaller sized location maps. There are two advantages of this technique which can be explained as follows: The first is to decrease image distortion as PEs are used for embedding data that are rearranged based on the magnitude of its EVM values (for example, an image), i.e., in an increasing order of EVM value. Thus, when data is embedded using a histogram shifting scheme, a lower PE value would be more advantageous in terms of distortion. The second advantage is to reduce the size of the location map. High PE values cause more overflow/underflow problems (map) than low efficient (depending on the amount of \(d_{i,j} = 0\) as seen in Table 1). Our prediction errors (PEs) performs better in embedding more data when compared to the conventional predictor [19]. The correlation of Gaussian weight prediction is higher than that of the mean values of four neighboring pixels. Note that an effective predictor is the predictor that provides considerable number of which PEs \((d_{i,j})\) equals to zero. When \(d_{i,j} = 0\), distortion of watermarked image is minimal.

4. Data Sorting [12]
PE values. Note that, the location map is part of the payload space and if the size of the location map decreases, the image distortion will be lower as well. The technique was first presented by Kamstra and Heijimans [18]. The concept is to select low PE values before high PE values in order to hide more data with less visual degradation and then use the correlation between a pair of neighboring pixels instead of using the embedded value directly. Kotvicha et al. [25] shows in his work that using, expanded variance mean (EVM) calculated from more numbers of neighboring pixels, could get a better sorting parameter than using local variance calculated by using four neighboring pixels like in Sachnev et al. work [19]. Figure 3 shows three different sequences of PE value of Lena and Baboon testing images calculated by a predictor of Sachnev et al.. In Fig. 3 (b), both images are sorted by using their local variances. In Fig. 3 (c), two sets of pixels are sorted by using their EVM. After comparing the sequence of PE sorted by local variance (Fig. 3 (b)) and by EVM (Fig. 3 (c)), the results show that the sequence of PE, sorted by EVM value, are better organized; i.e. close to ideality. EVM can be calculated as follows:

\[
\bar{\mu}_{i,j} = \frac{\sum_{L=-[\varphi_i/2]}^{[\varphi_j/2]} \sum_{k=1}^{[\varphi_j/2]} \mu_{i+j+l+2k} + \sum_{L=-[\varphi_i/2]}^{[\varphi_j/2]} \sum_{k=1}^{[\varphi_j/2]} \mu_{i+j+l+2k} + 1 - 4(\varphi_i,mod2)}{2[\varphi_i/2]^2 + 2[\varphi_j/2] + 1 - 4(\varphi_i,mod2)}
\]

where \((i, j) \in B_1 \times B_2 \cup B_3 \times B_4\) for a cross set, and \((i, j) \in B_1 \times B_3 \cup B_3 \times B_2\) for a dot set. The set \(B_1, B_2, B_3, B_4\) are defined by

\[
B_1 = \{[\varphi_i/2] + 2nn | n = 1, 2, \ldots, (N - (N + 1) mod 2 - 1)/2 - [\varphi_i/2]\}.
\]

\[
B_2 = \{[\varphi_i/2] + 2nn | n = 1, 2, \ldots, (M - (M + 1) mod 2 - 1)/2 - [\varphi_i/2]\}.
\]

\[
B_3 = \{[\varphi_i/2] + 2n + | n = 1, 2, \ldots, (N - N mod 2 - 2)/2 - [\varphi_i/2]\}.
\]

\[
B_4 = \{[\varphi_i/2] + 2n + 1 | n = 1, 2, \ldots, (M - M mod 2 - 2)/2 - [\varphi_i/2]\}.
\]

Let \(\mu_{i,j}\) be a local variance of the pixel \((i, j)\). It can be calculated by

\[
\mu_{i,j} = \frac{4}{\Delta v_k - \Delta u_k / 4},
\]

where \(\Delta v_1 = \left| v_{i+1,j} - v_{i-1,j} \right|, \Delta v_2 = \left| v_{i,j-1} - v_{i,j+1} \right|, \Delta v_3 = \left| v_{i+1,j} - v_{i-1,j} \right|, \Delta v_4 = \left| v_{i+1,j} - v_{i,j-1} \right|, \text{ and } \Delta \kappa = (\Delta v_1 + \Delta v_2 + \Delta v_3 + \Delta v_4)/4. \varphi_i\) is a level of variance expansion (VE).

One of the key steps of our algorithm is to sort the pixel according to the PE value. In practicality however, it is impossible to sort the pixel directly from the PE value because after embedding, the PE value will be changed. Since the magnitudes of EVM remain unchanged and closely related to magnitude of PE \(d_{i,j}\) of the pixel, it can be used as the sorting criterion in place of the PE value. Note, that if the level of variance expansion \(\varphi_i\) increases, the number of pixels used to calculate in (3) will also increase. The number of pixels involved in (3) equals to 

\[2[\varphi_i/2]^2 + 2[\varphi_i/2] + 1 - 4(\varphi_i,mod2).\]

Any suitable number of pixels for calculating EVM varies, depending on the image’s variation and payload size. We apply the level of variance expansion \(\varphi_i\) of EVM together with GA with more detail as described in Sect. 7. This tool helps sorting PE to be more accurate and gives better performance in decreasing image distortion, especially in small payloads. However, this was not significantly improved in high capacity because the high PE values were moved to the back of the sequence.
5. Histogram Shifting [12]

Thodi and Rodriquez [6] and [7] developed the histogram shifting method to decrease image distortion by using a threshold value associated with PE. Another advantage of this method is that it allows us to avoid overlapping problem caused by PE expansion. If PE values of some pixels are within the threshold \( t_p \) and \( t_n \), PE expansion will be used. In contrast, if the values are not in the threshold, shifting will be used instead. Shifting produces an adjusted value which lies outside the range of expanded values. An adjusted value can be evaluated using:

\[
D_{i,j} = \begin{cases} 
2d_{i,j} + b & \text{if } d_{i,j} \in [t_n, t_p] \\
 d_{i,j} + t_p + 1 & \text{if } d_{i,j} > t_p \\
 d_{i,j} + t_n & \text{if } d_{i,j} < t_n.
\end{cases}
\]

Expanded PE values range between \( 2t_n \) and \( 2t_p + 1 \), while the shifted PE are greater than \( 2t_p + 1 \) or less than \( 2t_n \). The original PE may be recovered with the equation:

\[
d_{i,j} = \begin{cases} 
\lfloor D_{i,j}/2 \rfloor & \text{if } D_{i,j} \in [2t_n, 2t_p + 1] \\
 D_{i,j} - t_p - 1 & \text{if } D_{i,j} > 2t_p + 1 \\
 D_{i,j} - t_n & \text{if } D_{i,j} < 2t_n.
\end{cases}
\]

Note that it is necessary to send a location map to the decoder for locating underflow/overflow pixels (unchanged pixels) which occur when an adjusted value is higher than 255 (for an 8 bit image) or lower than 0. These data can’t be presented as image since it is not in 8-bit unsigned integer.

6. Two Pass Testing

In reversible watermarking it is important to avoid underflow or overflow errors, where the modified pixels exceed the 8-bit range \([0, 255]\). This overflow/underflow problem with the histogram shift method cannot be avoided. The problematic pixels are those pixels which stay unchanged and cause overflow/underflow errors after data hiding. When a decoder calculates \( D_{i,j} \), if \( D_{i,j} \) is expanded or shifted and the result still remains in the range \([0, 255]\), then the pixel does not have an underflow/overflow problem and the location map is not required. If the result falls outside the range causing an underflow/overflow problem, then a location map will be developed. Location map size is the main problem in [4]–[7], [17] and [18]. The size is usually significant and needs to be compressed. Even after compressing, they still require a part of the payload space. Therefore, the size of the location map after compression will determine the method efficiency. In two-pass testing, location maps are very small or possibly not required.

Two-pass testing is the method presented by Sachnev et al. [19]. This technique is used to check if the modified pixel’s value will cause underflow or overflow problem by double expanding or shifting. Their methods divide the three possible cases that may arise as a result of the encoder testing (ET) as detailed below. ET(a) If the current pixel is modifiable twice, the corrective location map is not necessary. ET(b) If the current pixel is modifiable once owing to overflow/underflow errors during the second pass, this pixel is marked as “0” in the corrective location map. ET(c) If the current pixel is not modifiable even once, the pixel cannot be used in the embedding phase. This pixel is marked as “1” in the location map. From three cases mentioned above can be separated into seven sets as follows:

- **EE** is a set of the pixels of which the values are expandable twice without an underflow/overflow problem.
- **ES** is a set of the pixels of which the values are expandable in the first time and shiftable in the second time without an underflow/overflow problem.
- **SS** is a set of the pixels of which the values are shiftable twice without an underflow/overflow problem.
- **E** is a set of the pixels of which the values are expandable once and the second modification causes an underflow/overflow problem.
- **S** is a set of pixels of which their values are shiftable at the first shift but caused underflow or overflow at the second shift.
- **NE** is a set of the pixels of which their values are always unexpandable.
- **NS** is a set of the pixels of which their values cannot be shifted.

These seven sets cover all pixels that can be modified twice. Note that each set are named after their characters. Detailed explanation of two-pass testing can be found in appendix A. Data points in each set locate in different regions of a PE histogram as seen in Fig. 4.

All modified PE values are summarized as follows. The modified PE can be calculated by Eq. (4) and the modified pixel’s value is \( U_{i,j} = u'_{i,j} + D_{i,j} \).
The original PE can be recovered by:

A location map can be created using the equation:

\[
D_{i,j} = \begin{cases} 
2d_{i,j} + b & \text{if } (i, j) \in EE \cup ES \\
2d_{i,j} + 0 & \text{if } (i, j) \in E \cup d_{i,j} < 0 \\
2d_{i,j} + 1 & \text{if } (i, j) \in E \cup d_{i,j} \geq 0 \\
d_{i,j} + t_n & \text{if } (i, j) \in SS \cup S \cup d_{i,j} < 0 \\
d_{i,j} + t_p + 1 & \text{if } (i, j) \in SS \cup S \cup d_{i,j} \geq 0 \\
d_{i,j} & \text{if } (i, j) \in NE \cup NS.
\end{cases}
\]  

(4) A location map can be created using the equation:

\[
L_{i,j} = \begin{cases} 
0 & \text{if } (i, j) \in E \cup S \\
1 & \text{if } (i, j) \in NE \cup NS \\
\text{undefined} & \text{if } (i, j) \in EE \cup ES \cup SS.
\end{cases}
\]  

(5) The original PE can be recovered by:

\[
d_{i,j} = \begin{cases} 
[D_{i,j}/2] & \text{if } (i, j) \in EE \cup ES \cup E \\
D_{i,j} - t_n & \text{if } (i, j) \in SS \cup S \cup D_{i,j} < 0 \\
D_{i,j} - (t_p + 1) & \text{if } (i, j) \in SS \cup S \cup D_{i,j} \geq 0 \\
D_{i,j} & \text{if } (i, j) \in NE \cup NS.
\end{cases}
\]  

(6) Note that results of location map size for different payloads shows in Table 2 that our method produces smaller size of location map. Even if the location map is indispensable, its size is almost negligible compared to the payload. The biggest location map (4,406 bits) is used for peppers image when the payload is 250,000 bits.

### 7. Optimizations Using the Genetic Algorithm [12, 13]

Genetic algorithm (GA) is a well-known search heuristic optimization technique. The GA uses parameters called genes and chromosomes. In our application, “genes” are the parameters \(\sigma_x\), \(\sigma_y\) and \(\varphi\), in binary form that contains bits information, the longer genes will have better exchange potential. Our “chromosome” is the binary string composed of the three genes. Binary representation can be change into decimal by using the equation

\[
decimal = l_b + (u_b - l_b)(\sum_{i=1}^{m} 2^{-i} b_{i-1}/(2^m - 1))
\]  

(7) 

\(u_b, l_b \in R\) when \(u_b\) is a parameter’s upper bound. \(l_b\) is a parameter’s lower bound. \(m\) is a number of bits using for each gene and \(b_{i-1}\) is a binary value of \(i^{th}\) bits.

After generate \(n\) different binary string of chromosomes, they will be put into an objective function as show in Fig. 5. An objective function (sometime called fitness function) for GA is defined as mean square error (MSE) as follows:

\[
MSE = \frac{1}{N \times M} \sum_{i=1}^{N} \sum_{j=1}^{M} |u_{ij} - U_{i,j}|^2 / (N \times M)
\]  

(8) 

MSE is an average value for the square of the difference between an original image and an embedded image. We need to find values for \(\sigma_x\), \(\sigma_y\) and \(\varphi\), which give lowest MSE value.

To provide better GA performance, we would have to control the boundary of each parameter. Consider \(\sigma_x\) and \(\sigma_y\) as \(m\) bits which can represent up to \(2^m\) different values. In order to find an upper bound and a lower bound of the parameters, let \(w_{n}^{R}, w_{n}^{S}\) be the Gaussian weight, which located \(R\) unit away from the center along \(x\)-direction and \(y\)-direction respectively, so \(w_{n}^{R} = e^{-((R/2\sigma_x)^2)}\) and \(w_{n}^{S} = e^{-((R/2\sigma_y)^2)}\). Suppose \(w_{R=3}^{x} = \alpha w_{R=2}^{x}\). The Gaussian weight of a point which located 1 unit from the center is \(\alpha\) times bigger than the one that is 2 units from the center along \(x\)-direction. Suppose further that \(w_{R=1}^{x} = \beta w_{R=1}^{y}\) the weight, located 1 unit away from the center along \(x\)-direction, is \(\beta\) times weight along the \(y\)-direction. Thus, \(\sigma_x = \sqrt{3/(2\alpha \alpha)}\) and \(\sigma_y = \sqrt{3/(2\alpha \alpha + 6\alpha \beta)}\). If \(\alpha\) is very big, the curve will fade away quickly. If \(\beta \gg 1\), the weight along \(y\)-direction is relatively small compared to the other direction, so the curve expands mainly in \(x\)-direction. When defining \(\alpha = 10^4\) and \(\beta = 10^4\), we will get \(\sigma_x = 0.4\) and \(\sigma_y = 0.2\). For this reason, we set 0.2 to be a lower bound of SD values. Setting 5 as a parameter’s upper bound will give a region wide enough for the predicting function because the weight at radius 1 will be 2 times of the one at radius 6. Since SD is bounded, the domain of the parameter is limited. Therefore, the GA tool can perform faster. For \(\alpha = 10^4\) and \(\beta = 1\), \(\sigma_x = \sigma_y = 0.4\), meaning that all weight located at radius 1 have the same value and the weight at radius 2 is small enough to ignore. This case can be considered to be the same as Sachnev et al. [19] predicting function. The next step is updating all chromosomes using three basic steps of genetic operation, selection, crossover and mutation [26] as shown in Fig. 6. After all of the operations are done, the next generation will be produced. Chromosomes of new generation always give equal or better fitness function than previous generation even though the algorithm cannot guarantee that the best result will be found. However, our proximity to the best value depends on the number of generations.

### 8. Threshold Optimization

Threshold values affect a performance of an algorithm di-
rectly. Their values are used to identify which PE would be expanded or shifted. There are two thresholds; \( t_p \), for positive PE values, and \( t_n \), for negative PE values. Since, a distortion depends on a magnitude of PE, \( t_p \) and \( t_n \) should have a close magnitude to PE in order to select as small PE as possible for data embedding. However the magnitude of \( t_p \) and \( t_n \) have to be large enough to leave proper space for all payload. To select a proper set of \( t_p \) and \( t_n \), we will choose thresholds which optimize the lowest distortion. A proper set of thresholds depends on image’s properties and a payload size. Let \( \tau \), where \( \tau = 0, 1, \ldots, 254 \), be an index represents a set of thresholds. Define a set of corresponding thresholds for each \( \tau \) as \( t_p(\tau) = \lfloor \tau/2 \rfloor \), and \( t_n(\tau) = -\lceil \tau/2 \rceil \). When \( \tau \) increases, a range of \( [t_p(\tau), t_n(\tau)] \) will increase. Consequently, there are more available pixels for embedding data. To archive the proper thresholds, \( \tau \) is done, the dot embedding process is then started similarly. For the decoding process, it must be started on the dot sets first and then on the cross. Use \( \sigma_x \) and \( \sigma_y \) from each chromosome to calculate Gaussian weight values as in Eq. (1) and use them in a predicting function (2). Predict the cross set by using data from the dot set and vice versa. Evaluate PE of every pixel in the set. Use a level of sorting parameter, EVM, of every pixels in the set as in Eq. (3). Preserve the first \( h \) pixels in the set for header embedding. Reorder the rest of pixels in the set in ascending order by sorting the data using their EVM values. Check if each set of parameters gives enough space for embedding data. Start by checking the type of set that corresponds to each pixel as defined in definition 1 (see appendix A for detail). At the \( k^{th} \) order pixel in the sorted sequence, count the number of the embeddable pixels; \( BE(t_p, t_n, k) \), which we have found since the first pixel in the sequence. Define the set \( LC(t_p, t_n, k) \) to be the set of points that their order are less than or equal to \( k \) and these points need to be collected to form a location map. The member of this set belongs to the set \( E, S, NE \) and \( NS \). These two sets are defined by

\[
BE(t_p, t_n, k) = \{(i, j)_{t_p} \in EE \cup ES, 1 \leq l \leq k\}
\]

\[
LC(t_p, t_n, k) = \{(i, j)_{t_n} \in E \cup S \cup NE \cup NS, 1 \leq l \leq k\}
\]

where \((i, j)_{t_p}\) is the \( l^{th}\) order pixel in the sorted sequence. At the \( k^{th}\) order of the sequence, if the number of embeddable pixels is equal to the location map size plus the payload size as seen in Eq.(9), meaning that there is enough space for embedding data. In this case, we will modify the first \( k \) pixels of the sequence for data embedding.

9. Encoding and Decoding [12], [13]

9.1 Data Encoding

As discussed above, our proposed method exploit GA to optimize parameters used in predicting and sorting. An overall scheme of the encoding algorithm is depicted in block diagram as shown in Fig. 6. In order to use the encoding algorithm, first, set up the lowest possible thresholds for \( t_p, t_n \). Next, define initial binary values for \( n \) chromosomes which are used as an initial population for GA. Transform each chromosome to decimal values represent by \( \sigma_x, \sigma_y \) and \( \varphi_x \). Separate image data into a cross set and a dot set. Separate the payload into two equal parts. One part will be put into the cross set; the other will be put into the dot set. Note that the two sets (the cross set and dot set) are independent of each other. Independence means changes in one set do not affect the other set, and vice versa. Pixels from the dot set are used for computing predicted values \( u'_j \), whereas pixels from the cross set \( u_{t_{i,j}} \) are used for embedding data. The embedding process starts with the cross set. After embedding, the pixel in the cross set is then transformed while the dot set remains unchanged. Once the cross embedding process is done, the dot embedding process is then started similarly. For the decoding process, it must be started on the dot sets first and then on the cross. Use \( \sigma_x \) and \( \sigma_y \) from each chromosome to calculate Gaussian weight values as in Eq. (1) and use them in a predicting function (2). Predict the cross set by using data from the dot set and vice versa. Evaluate PE of every pixel in the set. Use a level of sorting parameter, EVM, of every pixels in the set as in Eq. (3). Preserve the first \( h \) pixels in the set for header embedding. Reorder the rest of pixels in the set in ascending order by sorting the data using their EVM values. Check if each set of parameters gives enough space for embedding data. Start by checking the type of set that corresponds to each pixel as defined in definition 1 (see appendix A for detail). At the \( k^{th} \) order pixel in the sorted sequence, count the number of the embeddable pixels; \( BE(t_p, t_n, k) \), which we have found since the first pixel in the sequence. Define the set \( LC(t_p, t_n, k) \) to be the set of points that their order are less than or equal to \( k \) and these points need to be collected to form a location map. The member of this set belongs to the set \( E, S, NE \) and \( NS \). These two sets are defined by

\[
BE(t_p, t_n, k) = \{(i, j)_{t_p} \in EE \cup ES, 1 \leq l \leq k\}
\]

\[
LC(t_p, t_n, k) = \{(i, j)_{t_n} \in E \cup S \cup NE \cup NS, 1 \leq l \leq k\}
\]

where \((i, j)_{t_p}\) is the \( l^{th}\) order pixel in the sorted sequence. At the \( k^{th}\) order of the sequence, if the number of embeddable pixels is equal to the location map size plus the payload size as seen in Eq.(9), meaning that there is enough space for embedding data. In this case, we will modify the first \( k \) pixels of the sequence for data embedding.
\[ n(EB(t_p, t_n, k)) = n(LC(t_p, t_n, k)) + n(Payload) \quad (9) \]

If \( k = n(\text{Cross set}) - n(\text{Header}) \) and \( n(EB(t_p, t_n, k)) < n(LC(t_p, t_n, k)) + n(\text{Payload}) \), the number of embeddable pixels corresponding to the thresholds are not enough for embedding all of the data. In other words, if every set of parameters in GA do not have enough space for embedding data, then GA will be terminated and threshold values will be raised (see step 5.4.3 in the encoding algorithm). If there are some sets of parameters that have enough space for embedding data, expanding or shifting will be applied to each pixel using Eq. (4). The embedding algorithm will be applied into the dot set to embed the rest of the payload and evaluate MSE. GA will also be terminated if its iterations reach the maximum generation, or GA will adjust the chromosomes until the fitness value remains unchanged. Keep increasing thresholds and repeat all steps until we get the one that produce lowest image distortion as detail in the previous section. Use the optimal thresholds for data embedding. The full encoding algorithm is shown below.

**Encoding algorithm**
1. Set \( \tau = 0 \) and set \( \varphi_p \).
2. Start GA by defining initial binary values for \( n \) chromosomes \( \sigma_x, \sigma_y, \) and \( \varphi_x \).
3. Transform all binary from parameters to decimal numbers using (7).
4. For each position in a cross set and a dot set.
5. For each set of parameters from each chromosome, apply it to:
   5.1 Calculate Gaussian weight using (1).
   5.2 For each position in the cross set compute:
      5.2.1 Prediction value using (2) and its PE.
      5.2.2 The sorting parameter, EVM, using (3).
6. Preserve the first \( h \) pixels for header and sort the rest of pixels in the Cross set in ascending order using EVM.
7. Check a space for embedding algorithm:
   7.1 Set up \( k \) as an index represents order in the sorted sequence and start from 1.
   7.2 Increase \( k \) one by one and count number of set \( EB \) and \( LC \).
   7.3 Find the order \( k \) where Eq. (9) satisfied. If such a position is found, mark this chromosome as available, or mark the chromosome unavailable otherwise.
8. For each chromosome marked as available, perform embedding and evaluate the result by the following steps 6.1-6.5. If no chromosomes are marked as available, terminate GA here, increase \( \tau \) and go to step 2.
   6.1. Collect the first \( h \) preserved pixels’ LSB and replace them by the header.
   6.2. Collect location map using (5) and put both LSB from the previous step and location map into payload.
   6.3. Modify the first \( k \) pixels and embed payload using Eq. (4) respectively and the modified pixel’s value is \( U_{i,j} = u'_{i,j} + D_{i,j}. \)
   6.4. Apply the steps 5.2-6.3 to the dot set.
   6.5. Evaluate MSE using (8).
7. If GA is converged or the number of iterations reaches the maximum generation, the lowest MSE obtained in the current generation is taken as the best fit value for the current threshold values. In this case, terminate GA and go to the next step. Otherwise, apply GA operations to chromosomes and go to step 3.
8. If five sets of threshold values and chromosomes are obtained, the one that gives the lowest MSE is selected and embedding is performed under the parameters specified by the set. Otherwise, increase \( \tau \) and go to step 2.

To decode a hiding data and recover an original image we need to have the parameters that were sent from an encoder. This information is collected in the area called header. A header has 51 pixels which involves 10 bits for \( \sigma_x \), another 10 bits for \( \sigma_y \), 3 bits for \( \varphi_x \) (for \( \varphi_x = 1, \ldots, 8 \)), 8 bits for \( r \) and the last 20 bits for a payload size. Since the parameter \( \varphi_x \), has to be known in order to sort the pixels, the header has to be embedded outside a sorting sequence. The header is embedded in the first 51 pixels of each set by replacing each corresponding pixel Least Significant Bit (LSB). However, pixel LSBs still need to be collected and put into the payload.

**9.2 Data Decoding**

A decoding algorithm from a host image has two major steps, watermark extracting and image recovering. During the encoding process, the cross set will be encoded before the dot set, which is converse to the decoding process. The decoding algorithm starts by extracting the header from the LSBs of the first \( h \) pixels of the dot set. Now we have all sufficient parameters and payload size. Then, calculate EVM and sort the rest of pixels in dot set using this set of parameter. Then, we calculate the prediction values and evaluate modified PE. Next, define the set that each pixel corresponds to using definition 2 (see appendix A for detail). Then, extract the payload from the pixels belonging to the sets \( EE \) and \( ES \). Identify the watermark, the header LSB and location map. The size of location map can be known by counting the pixels which do not belong to \( EE, ES \) or \( SS \). Now, the watermark is known. Next, recover the original picture. Place the header’s LSB back into the first \( h \) pixels of the set. All modified pixels in the sorted sequence can be recovered by using (6). After recovering the dot set, repeat all steps with the cross set.

**Decoding algorithm**
1. Separate all positions in an image into a cross set and a dot set.
2. Extract the header from the LSB of the first \( h \) pixels of dot set and convert all binary parameters to decimal using (7). The parameters \( \sigma_x, \sigma_y, \varphi_x, t_p, t_n, \) and payload size are obtained.
3. Calculate the EVM of the pixels starts from the \((h + 1)^{th}\) position in dot set and sort them in ascending order.
4. Calculate the prediction value and modified PE values of all pixels in the sequence using (2).
5. Define the set corresponding to each pixel in the sorted sequence using the definition 2 and the thresholds from the
header.
6. Extract the payload from the pixels which belong to the sets $EE$ and $ES$ sequentially.
7. Identify the watermark, header’s LSB and location map from payload.
8. Replace the header’s LSBs back to the first $h$ pixels.
Table 3  Parameters values obtained from genetic algorithm apply to six tested images with different payload sizes.

<table>
<thead>
<tr>
<th>Payload Size (bpp)</th>
<th>Lena</th>
<th>Peppers</th>
<th>Baboon</th>
<th>Barbara</th>
<th>Boat</th>
<th>Sailboat</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.038</td>
<td>0.1186, 0.6278</td>
<td>0.1188, 0.4958</td>
<td>0.1048, 0.5336</td>
<td>0.1110, 0.7397</td>
<td>0.1148, 0.3503</td>
<td>0.1632, 0.9307</td>
</tr>
<tr>
<td>0.076</td>
<td>0.1185, 0.7028</td>
<td>0.1192, 0.4930</td>
<td>0.1048, 0.6742</td>
<td>0.1110, 0.7397</td>
<td>0.1148, 0.3503</td>
<td>0.1632, 0.9307</td>
</tr>
<tr>
<td>0.114</td>
<td>0.1185, 0.7558</td>
<td>0.1192, 0.4930</td>
<td>0.1048, 0.5316</td>
<td>0.1110, 0.7114</td>
<td>0.1148, 0.3503</td>
<td>0.1632, 0.9307</td>
</tr>
<tr>
<td>0.153</td>
<td>0.1176, 0.7577</td>
<td>0.1161, 0.4952</td>
<td>0.1048, 0.6824</td>
<td>0.1110, 0.5433</td>
<td>0.1033, 0.2051</td>
<td>0.1262, 0.9118</td>
</tr>
<tr>
<td>0.191</td>
<td>0.1176, 0.7574</td>
<td>0.1170, 0.4952</td>
<td>0.1048, 0.6824</td>
<td>0.1110, 0.5433</td>
<td>0.1033, 0.2051</td>
<td>0.1262, 0.9118</td>
</tr>
<tr>
<td>0.229</td>
<td>0.1176, 0.7268</td>
<td>0.1170, 0.4952</td>
<td>0.1048, 0.6824</td>
<td>0.1110, 0.5433</td>
<td>0.1033, 0.2051</td>
<td>0.1262, 0.9118</td>
</tr>
<tr>
<td>0.267</td>
<td>0.1176, 0.7605</td>
<td>0.1170, 0.4952</td>
<td>0.1048, 0.6824</td>
<td>0.1110, 0.5433</td>
<td>0.1033, 0.2051</td>
<td>0.1262, 0.9118</td>
</tr>
<tr>
<td>0.305</td>
<td>0.1176, 0.7829</td>
<td>0.1170, 0.4952</td>
<td>0.1048, 0.6824</td>
<td>0.1110, 0.5433</td>
<td>0.1033, 0.2051</td>
<td>0.1262, 0.9118</td>
</tr>
<tr>
<td>0.343</td>
<td>0.1176, 0.8077</td>
<td>0.1170, 0.4952</td>
<td>0.1048, 0.6824</td>
<td>0.1110, 0.5433</td>
<td>0.1033, 0.2051</td>
<td>0.1262, 0.9118</td>
</tr>
<tr>
<td>0.381</td>
<td>0.1176, 0.8329</td>
<td>0.1170, 0.4952</td>
<td>0.1048, 0.6824</td>
<td>0.1110, 0.5433</td>
<td>0.1033, 0.2051</td>
<td>0.1262, 0.9118</td>
</tr>
</tbody>
</table>

9. Recover all modified pixels in sorting sequence using (6) and the location map extracted from payload.
10. Repeat step 2-9 with cross set.

10. Experimental Results

In our experiment, we tested our proposed method using several 512 × 512 standard test images including Lena, Ba-
boon, Boat, Peppers, and Sailboat, which have both low and high variation. Those images can be download from [27], while Barbara is obtained from [28]. The method effectiveness is measured by using the peak signal to noise ratio (PSNR). It is evaluated by $PSNR = 10 \log_{10}(255^2/ME)$ where MSE is the mean square error defined by Eq. (8). The higher quality image will have a higher PSNR value. The implementing results show that our proposed method gives better results than the previous works. Figure 7 shows the implementing results of the proposed method compared to Sachnev et al. [19] and Li et al. [23] method for all images. The tested payloads start from 10,000 bits and increasing by 10,000 bits. The PSNR of the proposed method is higher than both comparing method in every test images and in almost every size of payloads. This success is largely due to the fact that the coefficients which are used in our predicting function can be adapted to best suit each image while the predictor of Sachnev et al. method is static. Sachnev et al. method does not consider how far a local relation expands. The graph of Li et al. method is lower than the proposed one except at the tail of the graph in some images. When payload is larger, Li et al. method performs better due to two bit embedding strategy. Table 3 presents $t_p$, $t_\alpha$, $\sigma_x$, $\sigma_p$, and $\varphi_p$ parameters of each image in a different size of payload. The weight functions are different in each payload even in the same hosted image because the difference of their payload size. As is already stated, the proposed method is based on the method of Sachnev et al. [19]. By introducing the Gaussian weighted predictor and genetic algorithm for parameter optimization, the proposed method has achieved better results as shown in Fig. 7.

In addition, the proposed method has significant improvements over the Panyindee and Pintavirooj method [13]. Note from Table 4 that the scope of $\varphi_p = 6$ in our work achieves the average gain PSNR higher than $\varphi_p = 4$ obtained from [13] around 0.91 dB for Sailboat, and around 1 dB for Peppers. Because $\varphi_p = 4$ is insufficient for the area of accurate prediction, the appropriate maximum value of $\varphi_p$ should be equal to sixth level to meet the best possible result for these standard images.

11. Conclusion

Our proposed reversible watermarking method produces less image distortion than previous methods. Its adaptive models combine the predictor with the sorting parameter as well as optimization techniques. We used a Gaussian weight function for the predictor because it can be modified for specific parameter values by changing only two variables. The PE value cannot be used to sort data because hiding data causes sorting errors when the decoder attempts to reinterpret the data. Thus, instead of relying on the PE value, the EVM value is used to represent those values. We use a genetic algorithm to obtain a different sorting parameter. The proposed adaptive sorting tool and the predictor were proven to significantly increase image quality by minimizing image distortion.

Acknowledgments

This work was supported by the Office of the Higher Education Commission, Ministry of Education, Thailand, under the research project entitled “Higher Education Research Promotion” The authors would also like to thank all the anonymous reviewer for their useful comments and suggestions, which help to improve the quality of this manuscript.

References

(D = 3) > t_p; U = u′ + D + t_p + 1 = 250 + 3 + 1 + 1 = 255
(pixel after second shifted) (no map).

- Sample SS set \( t_p = 1; u = 251; u′ = 249; d = 2 \)
  \( d > t_p; U = u′ + d + t_p + 1 = 249 + 2 + 1 + 1 = 254 \)
  (pixel after first shifted)
  \((d + t_p + 1 = 4) > t_p; U = u′ + (d + t_p + 1) + t_p + 1 = 249 + 4 + 1 + 1 = 255 \)
  (pixel after second shifted) (no map).

- Sample E set \( t_p = 1; u = 252; u′ = 251; d = 1 \)
  \( d = t_p; U = u′ + D + t_p + 1 = 251 + 2(1) + 1 = 254 \)
  (pixel after first expanded)
  \((D = 3) > t_p; U = u′ + D + t_p + 1 = 251 + 3 + 1 + 1 = 256 \)
  (Overflow) (map = “0”).

- Sample S set \( t_p = 1; u = 253; u′ = 251; d = 2 \)
  \( d > t_p; U = u′ + d + t_p + 1 = 251 + 2 + 1 + 1 = 255 \)
  (pixel after first shifted)
  \((d + t_p + 1 = 4) > t_p; U = u′ + (d + t_p + 1) + t_p + 1 = 251 + 4 + 1 + 1 = 257 \)
  (Overflow) (map = “0”).

- Sample NE set \( t_p = 1; u = 254; u′ = 253; d = 1 \)
  \( d = t_p; U = u′ + D + t_p + 1 = 252 + 2 + 1 + 1 = 256 \)
  (Overflow) (map = “1”).

- Sample NS set \( t_p = 1; u = 254; u′ = 252; d = 2 \)
  \( d > t_p; U = u′ + d + t_p + 1 = 252 + 2 + 1 + 1 = 256 \)
  (Overflow) (map = “1”).

**Definition 1** The defining sets for an encoder:

\[
E(t_u, t_p) = \{(i, j) \mid 0 \leq 2d_{ij} + 1 \leq t_p \} \cap \left( u_{ij}′ + 4d_{ij} + 3 \leq 255 \right) \}
\cup \{(i, j) \mid J(t_u, 2d_{ij} < 0) \cap (u_{ij}′ + 4d_{ij} + 3 \geq 0) \}
\]

- Sample EE set \( t_p = 1; u = 252; u′ = 252; d = 0 \)
  \( d < t_p; U = u′ + D = 252 + 2(0) + 1 = 253 \)
  (pixel after first expanded)
  \((D = 1) > t_p; U = u′ + D + t_p + 1 = 252 + 2(1) + 1 = 255 \)
  (no map).

- Sample ES set \( t_p = 1; u = 252; u′ = 249; d = 2 \)
  \( d > t_p; U = u′ + d + t_p + 1 = 249 + 2 + 1 + 1 = 254 \)
  (pixel after first shifted)
  \((d + t_p + 1 = 4) > t_p; U = u′ + (d + t_p + 1) + t_p + 1 = 249 + 4 + 1 + 1 = 255 \)
  (pixel after second shifted) (no map).

- Sample SS set \( t_p = 1; u = 251; u′ = 249; d = 2 \)
  \( d > t_p; U = u′ + d + t_p + 1 = 249 + 2 + 1 + 1 = 254 \)
  (pixel after first shifted)
  \((d + t_p + 1 = 4) > t_p; U = u′ + (d + t_p + 1) + t_p + 1 = 249 + 4 + 1 + 1 = 255 \)
  (pixel after second shifted) (no map).

- Sample E set \( t_p = 1; u = 252; u′ = 251; d = 1 \)
  \( d = t_p; U = u′ + D + t_p + 1 = 251 + 2(1) + 1 = 254 \)
  (pixel after first expanded)
  \((D = 3) > t_p; U = u′ + D + t_p + 1 = 250 + 3 + 1 + 1 = 255 \)
  (pixel after second shifted) (no map).

**Appendix: Two Pass Testing Supplement**

Two-pass testing is a technique that is used to check pixel status before embedding. Pixel status is classified into 7 sets including EE, ES, SS, E, S, NE and NS. To check pixel status, double expanding or shifting is employed prior to the pixel embedding process using the histogram shift method. Provided here will be the simulated situation of the classification process of the 7 sets.

In reference to the rules of the histogram shift method, each pixel is checked using the two-pass testing, where each modification is an attempt to embed a test bit. Instead of using a random bit value, the test bit should be the hardest and extreme for embedding, i.e., “1” for positive d values and “0” for negatives (note that the output signal has maximum distortion under these cases). Let the threshold value be 1. There are seven samples:

- Sample EE set \( t_p = 1; u = 252; u′ = 252; d = 0 \)
  \( d < t_p; U = u′ + D = 252 + 2(0) + 1 = 253 \)
  (pixel after first expanded)
  \((D = 1) > t_p; U = u′ + D + t_p + 1 = 252 + 2(1) + 1 = 255 \)
  (no map).

- Sample ES set \( t_p = 1; u = 252; u′ = 252; d = 1 \)
  \( d = t_p; U = u′ + D + t_p + 1 = 250 + 2(1) + 1 = 253 \)
  (pixel after first expanded)
Definition 2 The defining sets for a decoder:

- \( EE(t_n, t_p) \in \{(i, j) \in J \mid (0 \leq D_{ij} < t_p) \land (u_{ij}' + 2D_{ij} + 1 + 255) \land (u_{ij}' + 2D_{ij} + 1 + 255) \} \)
- \( ES(t_n, t_p) \in \{(i, j) \in J \mid (t_p < D_{ij} < 2t_p + 1) \land (u_{ij}' + D_{ij} + t_p + 1 + 255) \} \)
- \( SS(t_n, t_p) \in \{(i, j) \in J \mid (2t_p \leq D_{ij} < t_n) \land (u_{ij}' + D_{ij} + t_n \geq 0) \} \)
- \( NE(t_n, t_p) \in \{(i, j) \in J \mid (0 \leq D_{ij} < t_p) \land (u_{ij}' + 2D_{ij} + 1 + 255) \land (u_{ij}' + 2D_{ij} + 1 + 255) \) \)
- \( NE(t_n, t_p) \in \{(i, j) \in J \mid (0 \leq D_{ij} < t_p) \land (u_{ij}' + 2D_{ij} + 1 + 255) \land (u_{ij}' + 2D_{ij} + 1 + 255) \)
- \( NE(t_n, t_p) \in \{(i, j) \in J \mid (0 \leq D_{ij} < t_p) \land (u_{ij}' + 2D_{ij} + 1 + 255) \land (u_{ij}' + 2D_{ij} + 1 + 255) \)
- \( NE(t_n, t_p) \in \{(i, j) \in J \mid (t_p < D_{ij} < 2t_p + 1) \land (u_{ij}' + D_{ij} + t_p + 1 + 255) \land (D_{ij} < t_n) \land (u_{ij}' + D_{ij} + t_n \geq 0) \) \)

For better understanding, we have shown the pathway status of \( d_{ij} \) in each set is shown in Fig. A.1 (for encoder) and Fig. A.2 (for decoder). Green arrows indicate the status that has no problem (\( EE, ES, SS \)), orange arrows indicate the status that seem to be good for embedding data but cannot be used (\( E \) and \( S \)) and the red arrows show the status that has problem from the beginning (\( NE \) and \( NS \)). We can see that each set can be identified both by an encoder and a decoder according to the rules of Definition 1 and Definition 2. Pixels in \( EE, ES, SS \) can be checked by an encoder because their values don’t have underflow/overflow problem like sets \( E, NE, S \) and \( NS \) do. To separate these pixels into their correct sets, we need to use a location map. Pixels that have a code “0” in a location map indicates “modified”, (expanded or shifted) and “1” means “unmodified”.

Consider an example for testing possible overflow errors (see *Sample E set). Embedding certain bit values (0 for positive \( d \) and 1 for negative \( d \)) to these pixels can change them from \( E \) set (map = “0”) to \( ES \) set (no map). As a result of this misclassification, the location map would be skewed.

- Sample E set when \( b = 0; t_p = 1; u = 252; u' = 251; d = 1; d = t_p; D = 251 + 2(1) + 0 = 253 \)

Assume that from Sample E set in the previous example will be used for embedding bit “0” (see *Sample E set). The pixel from *Sample E set would be expanded to 253. This
pixel belongs to the $E$ set and will be marked in the location map. During the decoder testing procedure this pixel would be classified as $ES$ set (instead of the correct $E$ set) and the location map would not be consulted. This would result in a cascade of incorrect decoding from that point onward. In order to resolve such ambiguity at the decoder, during the encoding process all expandable pixels classified as the $E$ set should have bit “1” embedded for all positive $d$ values and bit “0” for all negative $d$. This is the reason that $E$ set will not be used for embedding data.

Chaiyaporn Panyindee received B.S.Ind. Ed. (Electrical Engineering) from King Mongkut’s University of Technology Thonburi (KMUTT), Bangkok, Thailand in 2007 and M.Eng. (Electrical Engineering) from King Mongkut’s University of Technology Thonburi (KMUTT), Bangkok, Thailand in 2009. He is currently a D.Eng student in Department of Electronics, Faculty of Engineering, KMITL. His research interests include multimedia security, image and video processing, video compression and information hiding.

Chuchart Pintavirooj was born in Bangkok, Thailand in 1962. He received B.Sc. (Radiation Techniques) and M.Sc. (Biomedical Instrumentation) from Mahidol University, Bangkok, Thailand in 1985 and 1989, respectively. In 1995, he received another Master’s degree in Biomedical Engineering from Worcester Polytechnic Institute, MA, USA. In 2000, he earned a Ph.D. in Biomedical Engineering from Drexel University, Philadelphia, PA. After working as a research scientist at Biomedical Instrumentation Department, Mahidol University, he joined Department of Electronics, Faculty of Engineering, King Mongkut’s Institute of Technology Ladkrabang, Bangkok, where he is currently an associate professor. His current research is in Biomedical Image/Signal Processing majoring in Image Reconstruction, Image Classification, and Image Restoration. Dr. Pintavirooj is the acting chairman of Biomedical Engineering Society of Thailand affiliated with IFBME.