PAPER

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.* 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

Manuscript received January 17, 2016.

Manuscript revised April 28, 2016.

Manuscript publicized June 3, 2016.

[†]The authors are with the Faculty of Engineering, King Mongkut's Institute of Technology Ladkraban (KMUTL), Chalongkrung road, Ladkrabang, Bangkok 10520, Thailand.

a) E-mail: chaiyaporn.pan@gmail.com

b) E-mail: chuchartpintavirooj@gmail.com

DOI: 10.1587/transinf.2016EDP7030

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, φ_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 φ_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 φ_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, φ_p , will be further investigated in various types of images, including images with low to high frequency. We concluded that the prediction with φ_p at sixth level is the best size for standard images, where the φ_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

x	$v_{i-1,j}$	x	•
$v_{i,j-1}$	$u_{i,j}$	$v_{i,j+1}$	x
x	$v_{i+1,j}$	x	•
•	x	•	x

Fig. 1 Cross and dot set in an image.

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 $u'_{i,j}$ be a predicting value. PE can be calculated by $d_{i,j} = u_{i,j} - u'_{i,j}$. 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 $U_{i,j} = u'_{i,j} + D_{i,j}$. Extracting the watermark and recovering the original image can be done by $b = D_{i,j} \bmod 2$ and $u_{i,j} = u'_{i,j} + \lfloor D_{i,j}/2 \rfloor$, where $\lfloor \cdot \rfloor$ 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 $u'_{i,j} = \lfloor (v_{i,j-1} + v_{i,j+1} + v_{i-1,j} + v_{i+1,j})/4 \rfloor$. 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 produced 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 σ_x and σ_y respectively. The Gaussian weight function can be represented by

$$w(x, y) = e^{-(x^2/2\sigma_x^2 + y^2/2\sigma_y^2)}. \quad (1)$$

An adaptive predicting function is

$$u'_{i,j} = \left[\frac{\sum_{k=1}^{\varphi_p} \sum_{l=-\varphi_p}^{\varphi_p-(2k-1)} \left(w(l, l+2k-1) v_{(i+l, j+l+2k-1)} \right)}{\sum_{k=1}^{\varphi_p} \sum_{l=-\varphi_p}^{\varphi_p-(2k-1)} \left(w(l, l+2k-1) \right)} \right]. \quad (2)$$

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 gray-scale images.

Predictor type	PE value ($d_{i,j}$)	Lena	Barbara	Baboon	Boat	Peppers	Sailboat
Four-neighbor predictor	-2	25050	20284	9396	17352	18957	14868
	-1	32106	24808	10736	19245	20576	17352
	0	33132	25386	10727	19838	20974	19284
	1	28806	22568	10646	18452	20073	17706
	2	22408	19084	9132	15968	18340	14310
	Total	141502	112130	50637	90855	98920	83520
Gaussian weight predictor (σ_x, σ_y)	$d_{i,j}$	$\sigma_x=1.575, \sigma_y=0.777$	$\sigma_x=0.707, \sigma_y=0.543$	$\sigma_x=0.486, \sigma_y=0.538$	$\sigma_x=0.313, \sigma_y=0.205$	$\sigma_x=1.617, \sigma_y=4.512$	$\sigma_x=0.622, \sigma_y=4.911$
	-2	26050	21284	10982	19830	20252	16755
	-1	33346	25554	11080	21938	21720	19470
	0	35816	27550	11394	22400	21800	20004
	1	31010	25082	11382	21354	21152	17351
	2	23408	20084	10728	19038	19108	14752
	Total	149630	119554	55566	104560	104032	88332

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

$$A_1 = \{\varphi_p + 2n - 1 | n = 1, 2, \dots, (N - (N+1) \bmod 2 + 1) / 2 - \varphi_p\}.$$

$$A_2 = \{\varphi_p + 2n - 1 | n = 1, 2, \dots, (M - (M+1) \bmod 2 + 1) / 2 - \varphi_p\}.$$

$$A_3 = \{\varphi_p + 2n | n = 1, 2, \dots, (N - N \bmod 2) / 2 - \varphi_p\}.$$

$$A_4 = \{\varphi_p + 2n | n = 1, 2, \dots, (M - M \bmod 2) / 2 - \varphi_p\}.$$

N is number of row, and M is number of column in an image. φ_p is a level of expansion for a predictor.

$$\varphi_p \in \{1, 2, \dots, \min(\lfloor (N-1)/2 \rfloor, \lfloor (M-1)/2 \rfloor)\}.$$

From Eq. (1), the weight function is defined by using σ_x and σ_y in $w(x, y)$ and the level of expansion for a predictor (φ_p).

Higher levels of expansion will require more pixels to calculate a predicting value. The number of pixels corresponding to the level of expansion φ_p is equal to $2\varphi_p^2 + 2\varphi_p$. Figure 2 shows the shapes of the Gaussian function for four standard test images. The shapes of Gaussian are varies on different images and payload size. For example, the Gaussian function of the Sailboat image expands more in y -direction than in x -direction. The Gaussian function for the Peppers image expands wider than other pictures. In the opposite way, Baboon has a very narrow Gaussian shape. Therefore, using a proper shape of predictor will increase the prediction's

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]

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 low PE value would be more advantageous than a high PE value 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

Fig. 3 Graphs of PE show sequences of pixels sorted by different parameters. The first three images are sequences of the Lena image. The last three images are sequences of the Baboon image. (a) The unsorted PE sequence (b) The sequence of sorted PE by using local variance (c) The sequence of sorted PE by using EVM with $\varphi_s = 2$.

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 Heijmans [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:

$$\bar{\mu}_{i,j}^{\varphi_s} = \frac{\sum_{l=-\lceil\varphi_s/2\rceil}^{\lceil\varphi_s/2\rceil} \mu_{i+l,j+l} + \sum_{k=1}^{\lceil\varphi_s/2\rceil} \sum_{l=-\lceil\varphi_s/2\rceil}^{\lceil\varphi_s/2\rceil-2k} \left(\mu_{i+l,j+l+2k} + \mu_{i+l+2k,j+l} \right)}{2\lceil\varphi_s/2\rceil^2 + 2\lceil\varphi_s/2\rceil + 1 - 4(\varphi_s \bmod 2)} \quad (3)$$

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

$B_1 = \{\lceil\varphi_s/2\rceil + 2n | n = 1, 2, \dots, (N - (N+1) \bmod 2 - 1)/2 - \lceil\varphi_s/2\rceil\}$.

$B_2 = \{\lceil\varphi_s/2\rceil + 2n | n = 1, 2, \dots, (M - (M+1) \bmod 2 - 1)/2 - \lceil\varphi_s/2\rceil\}$.

$B_3 = \{\lceil\varphi_s/2\rceil + 2n + 1 | n = 1, 2, \dots, (N - N \bmod 2 - 2)/2 - \lceil\varphi_s/2\rceil\}$.

$B_4 = \{\lceil\varphi_s/2\rceil + 2n + 1 | n = 1, 2, \dots, (M - M \bmod 2 - 2)/2 - \lceil\varphi_s/2\rceil\}$.

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

$$\mu_{i,j} = \sum_{k=1}^4 (\Delta v_k - \Delta \bar{v}_k)^2 / 4,$$

where $\Delta v_1 = |v_{i,j-1} - v_{i-1,j}|$, $\Delta v_2 = |v_{i-1,j} - v_{i,j+1}|$, $\Delta v_3 = |v_{i,j+1} - v_{i+1,j}|$, $\Delta v_4 = |v_{i+1,j} - v_{i,j-1}|$, and $\Delta \bar{v}_k = (\Delta v_1 + \Delta v_2 + \Delta v_3 + \Delta v_4) / 4$. φ_s is a level of variance expansion (VE).

$\varphi_s \in \{1, 2, \dots, \min(2\lceil(N+1)/2\rceil - 2, 2\lceil(M+1)/2\rceil - 2)\}$.

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 φ_s increases, the number of pixels used to calculate in (3) will also increase. The number of pixels involved in (3) equals to, $2\lceil\varphi_s/2\rceil^2 + 2\lceil\varphi_s/2\rceil + 1 - 4(\varphi_s \bmod 2)$. 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 φ_s 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 Rodriguez [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:

$$D_{i,j} = \begin{cases} 2d_{i,j} + b & \text{if } d_{i,j} \in [t_n, t_p] \\ d_{i,j} + t_p + 1 & \text{if } d_{i,j} > t_p \geq 0 \\ d_{i,j} + t_n & \text{if } d_{i,j} < t_n < 0. \end{cases}$$

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

Fig. 4 All possible locations for PEs from 7 sets. (a) Histogram of original PE values before modification. (b) Histogram of modified PE values.

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 $U_{i,j} = u'_{i,j} + D_{i,j}$.

Table 2 Location map size vs. payload for tested images.

Pay Load (bpp)	Location map (bits)					
	Lena	Peppers	Baboon	Barbara	Boat	Sailboat
0.191	0	0	6	0	0	0
0.381	0	0	11	0	0	0
0.572	0	76	274	1	0	0
0.763	0	173	1251	76	0	16
0.839	0	578	-	1015	0	372
0.916	0	4406	-	-	0	2976
0.954	1	-	-	-	4	-

$$D_{i,j} = \begin{cases} 2d_{i,j} + b & \text{if } (i, j) \in EE \cup ES \\ 2d_{i,j} + 0 & \text{if } (i, j) \in E \wedge d_{i,j} < 0 \\ 2d_{i,j} + 1 & \text{if } (i, j) \in E \wedge d_{i,j} \geq 0 \\ d_{i,j} + t_n & \text{if } (i, j) \in SS \cup S \wedge d_{i,j} < 0 \\ d_{i,j} + t_p + 1 & \text{if } (i, j) \in SS \cup S \wedge d_{i,j} \geq 0 \\ d_{i,j} & \text{if } (i, j) \in NE \cup NS. \end{cases} \quad (4)$$

A location map can be created using the equation:

$$L_{i,j} = \begin{cases} 0 & \text{if } (i, j) \in E \cup S \\ 1 & \text{if } (i, j) \in NE \cup NS \\ \text{undefined} & \text{if } (i, j) \in EE \cup ES \cup SS. \end{cases} \quad (5)$$

The original PE can be recovered by:

$$d_{i,j} = \begin{cases} \lfloor D_{i,j}/2 \rfloor & \text{if } (i, j) \in EE \cup ES \cup E \\ D_{i,j} - t_n & \text{if } (i, j) \in SS \cup S \wedge D_{i,j} < 0 \\ D_{i,j} - (t_p + 1) & \text{if } (i, j) \in SS \cup S \wedge D_{i,j} \geq 0 \\ D_{i,j} & \text{if } (i, j) \in NE \cup NS. \end{cases} \quad (6)$$

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 σ_x , σ_y and φ_s 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

$$\text{decimal} = l_b + (u_b - l_b) \left(\sum_{i=1}^m 2^{i-1} b_{i-1} / (2^m - 1) \right) \quad (7)$$

$u_b, l_b \in R$ 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.

$b_{m_{\sigma_x}-1}b_{m_{\sigma_x}-2}\dots b_{0_{\sigma_x}}$	$b_{m_{\sigma_y}-1}b_{m_{\sigma_y}-2}\dots b_{0_{\sigma_y}}$	$b_{m_{\varphi_s}-1}b_{m_{\varphi_s}-2}\dots b_{0_{\varphi_s}}$
gene 1 = σ_x	gene 2 = σ_y	gene 3 = φ_s

Fig. 5 A chromosome in GA representing three parameters [12].

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:

$$MSE = \sum_{i=1}^N \sum_{j=1}^M |u_{i,j} - U_{i,j}|^2 / (N.M) \quad (8)$$

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 σ_x , σ_y and φ_s which give lowest MSE value.

To provide better GA performance, we would have to control the boundary of each parameter. Consider σ_x and σ_y , defined 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_R^x, w_R^y be the Gaussian weight, which located R unit away from the center along x -direction and y -direction respectively, so $w_R^x = e^{-(R^2/2\sigma_x^2)}$ and $w_R^y = e^{-(R^2/2\sigma_y^2)}$. Suppose $w_{R=1}^x = \alpha \cdot w_{R=2}^x$. The Gaussian weight of a point which located 1 unit from the center is α times bigger than the one that is 2 units from the center along x -direction. Suppose further that $w_{R=1}^x = \beta \cdot w_{R=1}^y$, the weight, located 1 unit away from the center along x -direction, is β times weight along the y -direction. Thus, $\sigma_x = \sqrt{3/(2\ln\alpha)}$ and $\sigma_y = \sqrt{3/(2\ln\alpha + 6\ln\beta)}$. If α 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-

Fig. 6 Flowchart represents an encoding algorithm using GA.

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 τ , where $\tau = 0, 1, \dots, 254$, be an index represents a set of thresholds. Define a set of corresponding thresholds for each τ as $t_p(\tau) = \lfloor \tau/2 \rfloor$, and $t_n(\tau) = -\lceil \tau/2 \rceil$. When τ 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, $t_p(\tau)$ and $t_n(\tau)$, the lowest possible k is initialized as the possible range of t_n and t_p that will give the best PSNR based on the required payload. In other words, k is the loop number for iteratively adjusting the parameter. k is then increased until the smallest possible τ where the corresponding thresholds, large enough for embedding data are found. The thresholds are then used to embed data. The MSE is then calculated corresponding to the τ value that has been adjusted five times. According to the experiments, five adaptive times works well for finding the optimal thresholds in most cases. The first modification generally gives the best result.

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 σ_x , σ_y and φ_s . 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 $u'_{i,j}$, whereas pixels from the cross set $u_{i,j}$ 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 σ_x and σ_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; $EB(t_p, t_n, k)$, 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

$$EB(t_n, t_p, k) = \{(i, j)_{s_l} \in EE \cup ES, 1 \leq l \leq k\}$$

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

where $(i, j)_{s_l}$ 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 $n(EB(t_p, t_n, k)) < n(LC(t_p, t_n, k)) + n(Payload)$, 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 φ_p .
2. Start GA by defining initial binary values for n chromosomes σ_x , σ_y , and φ_s .
3. Transform all binary from parameters to decimal numbers using (7).
4. Define positions 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).
 - 5.3. Preserve the first h pixels for header and sort the rest of pixels in the Cross set in ascending order using EVM.
 - 5.4. Check a space for embedding algorithm:
 - 5.4.1 Set up k as an index represents order in the sorted sequence and start from 1.
 - 5.4.2 Increase k one by one and count number of set EB and LC .
 - 5.4.3 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.
 6. For each chromosome marked as available, perform embedding and evaluate the result by the following steps 6.1-6.5.
 - 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 τ 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 σ_x , another 10 bits for σ_y , 3 bits for φ_s (for $\varphi_s = 1, \dots, 8$), 8 bits for τ and the last 20 bits for a payload size. Since the parameter φ_s , 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 hosted 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 σ_x , σ_y , φ_s , 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

Fig. 7 Payload (bpp) vs. distortion (PSNR) graphs of the proposed method compare to Sachnev *et al.* [19] and Li *et al.* [23] apply to six test images, Lena [12], Barbara [13], Baboon [12], Boat [13], Peppers and Sailboat, respectively.

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.

Pay Load (bpp)	Lena	Peppers	Baboon	Barbara	Boat	Sailboat
	$t_p, t_n, \sigma_x, \sigma_y, \varphi_s$	$t_p, t_n, \sigma_x, \sigma_y, \varphi_s$	$t_p, t_n, \sigma_x, \sigma_y, \varphi_s$	$t_p, t_n, \sigma_x, \sigma_y, \varphi_s$	$t_p, t_n, \sigma_x, \sigma_y, \varphi_s$	$t_p, t_n, \sigma_x, \sigma_y, \varphi_s$
0.038	0,-1,1.861,0.627,8	0,-1,1.884,4.958,8	0,-1,0.458,0.533,6	0,-1,1.110,1.739,7	0,-1,1.448,1.350,3	0,-1,0.632,3.930,7
0.076	0,-1,1.856,0.702,8	0,-1,0.923,4.930,8	0,-1,0.458,0.674,2	0,-1,1.110,1.739,7	0,-1,1.181,0.711,2	0,-1,0.514,4.911,6
0.114	0,-1,1.655,0.758,7	0,-1,0.923,4.930,7	1,-2,0.458,0.510,6	0,-1,1.185,0.711,4	0,-1,1.181,0.519,2	1,-2,0.622,4.897,8
0.153	0,-1,1.575,0.777,7	1,-1,1.617,4.512,8	2,-2,0.486,0.538,4	0,-1,0.707,0.543,3	0,-1,0.313,0.205,1	1,-2,0.622,4.911,8
0.191	0,-1,1.289,0.754,4	1,-1,1.570,2.457,4	3,-3,0.482,0.538,4	1,-1,1.185,0.711,4	1,-1,0.463,0.369,2	2,-2,0.552,4.911,8
0.229	0,-1,1.138,0.726,2	1,-2,1.326,4.869,8	4,-4,0.697,0.927,4	0,-1,0.707,0.543,3	1,-2,0.993,0.519,2	2,-3,0.627,4.892,8
0.267	1,-1,1.495,0.796,5	1,-2,1.575,2.588,4	4,-5,0.697,0.927,4	1,-2,1.167,0.749,3	1,-2,0.463,0.350,2	3,-3,0.566,4.897,8
0.305	1,-1,1.289,0.726,2	2,-2,1.608,3.264,6	5,-6,0.679,0.866,4	1,-2,0.707,0.543,4	2,-2,0.463,0.369,2	3,-4,0.585,4.897,8
0.343	1,-2,1.570,0.796,5	2,-3,1.434,4.376,6	6,-7,0.679,0.871,4	2,-3,1.167,0.740,3	2,-3,0.993,0.519,2	4,-4,0.622,3.855,8
0.381	1,-2,1.495,0.796,5	2,-3,1.880,2.584,4	7,-8,0.660,0.857,4	2,-3,0.707,0.543,4	2,-3,0.425,0.350,2	4,-4,0.796,2.593,4
0.420	1,-2,1.251,0.763,2	3,-4,1.781,3.728,6	8,-9,0.660,0.866,6	3,-3,0.697,0.543,4	3,-3,0.444,0.350,2	4,-5,0.791,2.485,4
0.458	2,-2,1.411,0.857,4	3,-4,1.880,2.588,4	9,-10,0.650,0.857,4	3,-4,0.697,0.543,4	3,-4,0.444,0.360,1	5,-6,1.096,2.335,4
0.496	2,-3,1.547,0.913,5	4,-4,1.875,2.555,5	10,-10,0.472,0.566,4	4,-4,0.702,0.538,4	4,-4,1.068,0.519,2	5,-6,0.932,2.035,4
0.534	2,-3,1.335,0.918,2	4,-5,1.889,2.490,4	4,-5,1.889,2.490,4	4,-5,0.707,0.543,4	4,-5,1.012,0.500,2	6,-7,0.941,2.067,4
0.572	3,-3,1.270,0.885,3	5,-5,1.884,2.490,4	10,-10,0.425,0.547,2	4,-5,0.693,0.538,8	5,-5,0.993,0.519,2	7,-8,1.073,1.955,4
0.610	3,-4,1.340,0.885,3	6,-6,1.875,2.560,6	11,-11,0.425,0.547,2	5,-6,0.693,0.538,8	5,-6,1.012,0.505,2	8,-8,1.077,1.584,2
0.648	3,-4,1.190,0.885,3	6,-7,1.945,2.044,4	12,-13,0.430,0.557,4	6,-7,0.510,0.425,8	6,-6,0.993,0.500,1	8,-9,1.021,1.467,2
0.687	4,-5,1.274,0.913,3	7,-7,1.927,2.044,4	13,-14,0.500,0.993,6	7,-8,0.528,0.425,8	6,-7,0.970,0.500,1	9,-10,1.021,1.448,2
0.725	4,-5,1.190,0.843,2	8,-8,2.035,1.640,4	15,-16,0.200,0.209,4	8,-9,0.533,0.421,8	7,-8,1.021,0.500,1	10,-11,1.016,1.401,2
0.763	5,-6,1.115,0.805,2	9,-9,2.021,1.406,4	18,-18,0.425,0.566,4	9,-10,0.514,0.378,8	8,-9,0.937,0.505,1	11,-12,0.988,1.260,2
0.801	6,-7,1.115,0.801,2	10,-10,1.669,1.448,2	20,-21,0.275,0.303,8	11,-11,0.514,0.369,8	9,-9,0.984,0.500,1	12,-13,0.979,1.106,2
0.839	7,-8,1.110,0.801,2	12,-12,1.673,1.199,2	24,-25,0.275,0.308,8	11,-12,0.524,0.238,8	10,-10,0.913,0.500,1	12,-13,0.815,1.106,1
0.877	7,-8,0.829,0.655,2	15,-15,1.462,1.110,2	-----	13,-13,0.618,0.266,8	11,-12,0.862,0.500,1	15,-16,0.801,0.918,1
0.916	9,-9,0.829,0.655,2	18,-18,1.190,0.951,1	-----	-----	13,-14,0.843,0.500,1	17,-18,0.815,0.946,2
0.954	11,-12,0.655,0.519,2	-----	-----	-----	17,-18,0.810,0.467,1	-----

Table 4 Comparison of PSNR between Panyindee and Pintavirooj method [13] at $\varphi_p = 4$ and the proposed method at $\varphi_p = 6$, for payload from 0.038 to 0.954 bpp.

Pay Load (bpp)	PSNR (dB)											
	Lena		Baboon		Barbara		Boat		Sailboat		Peppers	
	[13]	Ours	[13]	Ours	[13]	Ours	[13]	Ours	[13]	Ours	[13]	Ours
	($\varphi_p=4$)	($\varphi_p=6$)	($\varphi_p=4$)	($\varphi_p=6$)	($\varphi_p=4$)	($\varphi_p=6$)	($\varphi_p=4$)	($\varphi_p=6$)	($\varphi_p=4$)	($\varphi_p=6$)	($\varphi_p=4$)	($\varphi_p=6$)
0.038	59.190	59.812	54.708	55.001	59.318	59.322	57.336	57.355	57.854	59.663	55.788	57.014
0.076	55.729	56.188	49.473	49.772	56.018	56.011	53.225	53.236	53.195	55.181	52.497	53.397
0.114	53.716	54.202	46.464	46.769	53.630	53.636	50.896	50.883	50.355	51.686	50.451	51.021
0.153	52.204	52.571	43.899	44.251	51.904	51.906	49.010	49.004	47.278	49.292	47.223	48.223
0.191	50.995	51.318	41.974	42.226	49.618	49.613	46.494	46.507	45.545	47.237	46.004	46.708
0.229	49.798	50.098	40.387	40.657	48.467	48.473	45.038	45.043	44.200	45.002	44.590	45.491
0.267	47.782	48.201	38.982	39.354	47.109	47.120	43.996	44.003	42.572	44.009	42.964	44.176
0.305	46.914	47.324	37.724	38.002	45.987	45.988	42.743	42.737	41.402	42.692	42.178	43.185
0.343	46.021	46.543	36.569	36.804	44.956	44.978	41.608	41.678	40.220	41.412	41.182	42.226
0.381	45.332	45.736	35.473	35.724	44.165	44.138	40.916	40.924	39.258	40.356	40.162	41.190
0.420	44.490	44.973	34.452	34.792	43.158	43.182	40.008	40.012	38.278	39.296	39.349	40.498
0.458	43.581	44.105	33.468	33.715	42.379	42.374	39.285	39.297	37.407	38.372	38.568	39.868
0.496	42.922	43.393	32.543	32.864	41.512	41.508	38.485	38.491	36.654	37.507	37.876	39.180
0.534	42.341	42.842	31.646	31.993	40.749	40.752	37.892	37.890	35.963	36.759	37.231	38.434
0.572	41.619	42.106	30.794	31.209	39.991	39.994	37.188	37.192	35.251	35.978	36.636	37.835
0.610	41.070	41.509	29.971	30.468	39.246	39.245	36.582	36.580	34.597	35.338	36.062	37.225
0.648	40.513	40.993	29.193	29.791	38.464	38.465	36.022	36.014	33.994	34.596	35.508	36.512
0.687	39.900	40.394	28.425	28.984	37.638	37.639	35.500	35.494	33.431	33.831	34.978	36.106
0.725	39.360	39.759	27.665	28.465	36.747	36.749	34.993	34.994	32.861	33.253	34.455	35.563
0.763	38.789	39.209	26.910	27.516	35.650	35.652	34.439	34.442	32.293	32.503	33.979	34.974
0.801	38.202	38.498	26.140	27.042	34.512	34.509	34.093	34.095	31.750	32.046	33.509	34.480
0.839	37.600	37.802	25.188	26.288	33.312	33.311	33.656	33.656	31.211	31.311	33.031	33.832
0.877	36.938	37.134	-----	-----	32.257	32.254	33.122	33.119	30.624	30.742	32.527	33.133
0.916	36.192	36.497	-----	-----	-----	-----	32.712	32.711	30.009	30.005	31.974	32.473
0.954	35.303	35.706	-----	-----	-----	-----	32.000	32.004	-----	-----	-----	-----

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 $PSNR = 10\log_{10}((255)^2/MSE)$ 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_n , σ_x , σ_y , and φ_s 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 φ_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

- [1] B. Macq, "Lossless multiresolution transform for image authenticating watermarking," In Proc. EUSIPCO, pp.533–536, Sept. 2000.
- [2] C.W. Honsinger, P. Jones, M. Rabbani, and J.C. Stoffel, "Lossless recovery of an original image containing embedded data," U.S. Patent 6 278 791, 2001.
- [3] C. De Vleeschouwer, J.-F. Delaigle, and B. Macq, "Circular interpretation of bijective transformation in lossless watermarking for media asset management," IEEE Trans. Multimedia, vol.5, no.1, pp.97–105, March 2003.
- [4] J. Tian, "Reversible data embedding using a difference expansion," IEEE Trans. Circuits Syst. Video Technol., vol.13, no.8, pp.890–896, Aug. 2003.
- [5] D.M. Thodi and J.J. Rodriguez, "Prediction-error based reversible watermarking," In Proc. IEEE Int. Conf. Image Process., vol.3, pp.1549–1552, Oct. 2004.
- [6] D.M. Thodi and J.J. Rodriguez, "Reversible watermarking by prediction-error expansion," in Proc. IEEE Southwest Symp. Image Analysis and Interpretation., pp.21–25, March 2004.
- [7] D.M. Thodi and J.J. Rodriguez, "Expansion embedding techniques for reversible watermarking," IEEE Trans. Image Process., vol.16, no.3, pp.721–730, March 2007.
- [8] Y. Hu, H.-K. Lee, and J. Li, "DE-based Reversible data hiding with improved overflow location map," IEEE Trans. Circuits Syst. Video Technol., vol.19, no.2, pp.250–260, Feb. 2009.
- [9] H.J. Hwang, H.J. Kim, V. Sachnev, and S.H. Joo, "Reversible watermarking method using optimal histogram pair shifting based on prediction and sorting," KSII, Trans. Internet Inform. Syst., vol.4, no.4, pp.655–670, Aug. 2010.
- [10] G. Coatrieux, W. Pan, N. Cuppens-Boulahia, F. Cuppens, and C. Roux, "Reversible watermarking based on invariant image classification and dynamic histogram shifting," IEEE Trans. Inf. Forensics Security., vol.8, no.1, pp.111–120, Jan. 2013.
- [11] B. Ou, X. Li, Y. Zhao, and R. Ni, "Reversible data hiding based on PDE predictor," Journal of Systems and Software., vol.86, no.10, pp.2700–2709, Oct. 2013.
- [12] C. Panyindee and C. Pintavirooj, "Reversible Watermarking Using Gaussian Weight Prediction and Genetic Algorithm," in Proc. IMECS, vol.1, pp.457–461, March 2013.
- [13] C. Panyindee and C. Pintavirooj, "Optimizations using the genetic algorithm for reversible watermarking," in Proc. ECTI-CON, pp.1–5, 2013.
- [14] M.U. Celik, G. Sharma, A.M. Tekalp, and E. Saber, "Reversible data hiding," In Proc. IEEE Int. Conf. Image Process., vol.2, pp.157–160, Sept. 2002.
- [15] J. Fridrich, M. Goljan, and R. Du, "Lossless data embedding-New paradigm in digital watermarking," EURASIP J. Appl. Signal Process., vol.2002, no.1, pp.185–196, Jan. 2002.
- [16] Z. Ni, Y.Q. Shi, N. Ansari, W. Su, Q. Sun, and X. Lin, "Robust lossless image data hiding," In Proc. IEEE Int. Conf. Multimedia and Expo, pp.2199–2202, 2004.
- [17] S. Weng, Y. Zhao, R. Ni, and J.-S. Pan, "Lossless data hiding based on prediction-error adjustment," Sci China Ser F-Inf Sci, vol.52, no.2, pp.269–275, Feb. 2009.

- [18] L.H.J. Kamstra and A.M. Heijmans, "Reversible data embedding into images using wavelet techniques and sorting," *IEEE Trans. Image Process.*, vol.14, no.12, pp.2082–2090, Dec. 2005.
- [19] V. Sachnev, H.J. Kim, J. Nam, S. Suresh, and Y.Q. Shi, "Reversible watermarking algorithm using sorting and prediction," *IEEE Trans. Circuits Syst. Video Technol.*, vol.19, no.7, pp.989–999, July 2009.
- [20] L. Yang and P. Hao, "Infinity Norm Rotation transforms," *IEEE Trans. Signal Process.*, vol.57, no.7, pp.2594–2603, July 2009.
- [21] D. Coltuc and J.-M. Chassery, "Very fast watermarking by reversible contrast mapping," *IEEE Signal Process. Lett.*, vol.14, no.4, pp.255–258, April 2007.
- [22] C. Dragoi and D. Coltuc, "Improved rhombus interpolation for reversible watermarking by difference expansion," in *Proc. EUSIPCO*, pp.1688–1692, 2012.
- [23] X. Li, B. Yang, and T. Zeng, "Efficient reversible watermarking based on adaptive prediction-error expansion and pixel selection," *IEEE Trans. Image Process.*, vol.20, no.12, pp.3524–3533, Dec. 2011.
- [24] X. Li, B. Li, B. Yang, and T. Zeng, "General Framework to Histogram-Shifting-Based Reversible Data Hiding," *IEEE Trans. Image Process.*, vol.22, no.6, pp.2181–2191, June 2013.
- [25] C. Kotvicha, P. Sanguansat, and M.L.K. Kasemsa, "Expand variance mean sorting for reversible watermarking," in *Proc. IEEE Int. Conf. Signal Inform. Process.*, pp.668–673, 2012.
- [26] A. Chipperfield, P. Fleming, H. Pohlheim, and C. Fonseca, "Genetic Algorithm Toolbox for Use with MATLAB User's Guide Version 1.2," Department of Automatic Control and System Engineering, University of Sheffield, England.
- [27] The USC-SIPI Image Database, Miscellaneous [Online]. Available: <http://sipi.usc.edu/database/database.php?volume=misc>. [Accessed: Oct. 11, 2013].
- [28] Miscellaneous Gray Level Images (512x512) [Online]. Available: <http://decsai.ugr.es/cvg/dbimagenes/g512.php>. [Accessed: Oct. 11, 2013].

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 herewith 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)
 $(D = 1) = t_p; U = u' + D = 252 + 2(1) + 1 = 255$
 (pixel after second expanded) (no map).
- Sample *ES* set $t_p = 1; u = 251; u' = 250; d = 1$
 $d = t_p; U = u' + D = 250 + 2(1) + 1 = 253$
 (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).

- 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 $t_p = 1; u = 252; u' = 251; d = 1$
 $d = t_p; U = u' + D = 251 + 2(1) + 1 = 254$
 (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 $t_p = 1; u = 254; u' = 253; d = 1$
 $d = t_p; U = u' + D = 253 + 2(1) + 1 = 256$
 (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:

- $EE(t_n, t_p) \in \{(i, j) \in J \mid (0 \leq 2d_{i,j} + 1 \leq t_p) \wedge (u'_{i,j} + 4d_{i,j} + 3 \leq 255)\} \cup \{(i, j) \in J \mid (t_n \leq 2d_{i,j} < 0) \wedge (u'_{i,j} + 4d_{i,j} \geq 0)\}$.
- $ES(t_n, t_p) \in \{(i, j) \in J \mid (0 \leq d_{i,j} \leq t_p) \wedge (2d_{i,j} + 1 > t_p) \wedge (u'_{i,j} + 2d_{i,j} + t_p + 2 \leq 255)\} \cup \{(i, j) \in J \mid (t_n \leq d_{i,j} < 0) \wedge (2d_{i,j} < t_n) \wedge (u'_{i,j} + 2d_{i,j} + t_n \geq 0)\}$.
- $SS(t_n, t_p) \in \{(i, j) \in J \mid (d_{i,j} > t_p) \wedge (u'_{i,j} + d_{i,j} + 2t_p + 2 \leq 255)\} \cup \{(i, j) \in J \mid (d_{i,j} < t_n) \wedge (u'_{i,j} + d_{i,j} + 2t_n \geq 0)\}$.
- $E(t_n, t_p) \in \{(i, j) \in J \mid (0 \leq 2d_{i,j} + 1 \leq t_p) \wedge (u'_{i,j} + 2d_{i,j} + 1 \leq 255) \wedge (u'_{i,j} + 4d_{i,j} + 3 > 255)\} \cup \{(i, j) \in J \mid (0 \leq d_{i,j} \leq t_p) \wedge (2d_{i,j} + 1 > t_p) \wedge (u'_{i,j} + 2d_{i,j} + 1 \leq 255) \wedge (u'_{i,j} + 2d_{i,j} + t_p + 2 > 255)\} \cup \{(i, j) \in J \mid (t_n \leq 2d_{i,j} < 0) \wedge (u'_{i,j} + 2d_{i,j} \geq 0) \wedge (u'_{i,j} + 4d_{i,j} < 0)\} \cup \{(i, j) \in J \mid (t_n \leq d_{i,j} < 0) \wedge (2d_{i,j} < t_n) \wedge (u'_{i,j} + 2d_{i,j} \geq 0) \wedge (u'_{i,j} + 2d_{i,j} + t_n < 0)\}$.
- $NE(t_n, t_p) \in \{(i, j) \in J \mid (0 \leq d_{i,j} \leq t_p) \wedge (u'_{i,j} + 2d_{i,j} + 1 > 255)\} \cup \{(i, j) \in J \mid (t_n \leq d_{i,j} < 0) \wedge (u'_{i,j} + 2d_{i,j} < 0)\}$.
- $S(t_n, t_p) \in \{(i, j) \in J \mid (d_{i,j} > t_p) \wedge (u'_{i,j} + d_{i,j} + t_p + 1 \leq 255) \wedge (u'_{i,j} + d_{i,j} + 2t_p + 2 > 255)\} \cup \{(i, j) \in J \mid (d_{i,j} < t_n) \wedge (u'_{i,j} + d_{i,j} + t_n \geq 0) \wedge (u'_{i,j} + d_{i,j} + 2t_n < 0)\}$.
- $NS(t_n, t_p) \in \{(i, j) \in J \mid (d_{i,j} > t_p) \wedge (u'_{i,j} + d_{i,j} + t_p + 1 > 255)\} \cup \{(i, j) \in J \mid (d_{i,j} < t_n) \wedge (u'_{i,j} + d_{i,j} + t_n <$

0)).

Definition 2 The defining sets for a decoder:

- $EE(t_n, t_p) \in \{(i, j) \in J \mid (0 \leq D_{i,j} \leq t_p) \wedge (u'_{i,j} + 2D_{i,j} + 1 \leq 255)\} \cup \{(i, j) \in J \mid (t_n \leq D_{i,j} < 0) \wedge (u'_{i,j} + 2D_{i,j} \geq 0)\}$.
- $ES(t_n, t_p) \in \{(i, j) \in J \mid (t_p < D_{i,j} \leq 2t_p + 1) \wedge (u'_{i,j} + D_{i,j} + t_p + 1 \leq 255)\} \cup \{(i, j) \in J \mid (2t_n \leq D_{i,j} < t_n) \wedge (u'_{i,j} + D_{i,j} + t_n \geq 0)\}$.
- $SS(t_n, t_p) \in \{(i, j) \in J \mid (D_{i,j} > 2t_p + 1) \wedge (u'_{i,j} + D_{i,j} + t_p + 1 \leq 255)\} \cup \{(D_{i,j} < 2t_n) \wedge (u'_{i,j} + D_{i,j} + t_n \geq 0)\}$.
- $E(t_n, t_p) \in \{(i, j) \in J \mid (0 \leq D_{i,j} \leq t_p) \wedge (u'_{i,j} + 2D_{i,j} + 1 > 255) \wedge (L_{i,j} = 0)\} \cup \{(i, j) \in J \mid (t_p < D_{i,j} \leq 2t_p + 1) \wedge (u'_{i,j} + D_{i,j} + t_p + 1 > 255) \wedge (L_{i,j} = 0)\} \cup \{(i, j) \in J \mid (t_n \leq D_{i,j} < 0) \wedge (u'_{i,j} + 2D_{i,j} < 0) \wedge (L_{i,j} = 0)\} \cup \{(i, j) \in J \mid (2t_n \leq D_{i,j} < t_n) \wedge (u'_{i,j} + D_{i,j} + t_n < 0) \wedge (L_{i,j} = 0)\}$.
- $NE(t_n, t_p) \in \{(i, j) \in J \mid (0 \leq D_{i,j} \leq t_p) \wedge (u'_{i,j} + 2D_{i,j} + 1 > 255) \wedge (L_{i,j} = 1)\} \cup \{(i, j) \in J \mid (t_n \leq D_{i,j} < 0) \wedge (u'_{i,j} + 2D_{i,j} < 0) \wedge (L_{i,j} = 1)\}$.
- $S(t_n, t_p) \in \{(i, j) \in J \mid (D_{i,j} > 2t_p + 1) \wedge (u'_{i,j} + D_{i,j} + t_p + 1 > 255) \wedge (L_{i,j} = 0)\} \cup \{(i, j) \in J \mid (D_{i,j} < 2t_n) \wedge (u'_{i,j} + D_{i,j} + t_n < 0) \wedge (L_{i,j} = 0)\}$.
- $NS(t_n, t_p) \in \{(i, j) \in J \mid (D_{i,j} > t_p) \wedge (u'_{i,j} + D_{i,j} + t_p + 1 > 255) \wedge (L_{i,j} = 1)\} \cup \{(i, j) \in J \mid (D_{i,j} < t_n) \wedge (u'_{i,j} + D_{i,j} + t_n < 0) \wedge (L_{i,j} = 1)\}$.

For better understanding, we have shown the pathway status of $d_{i,j}$ 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 and 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 and SS can be checked by a decoder 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 $b = 0$; $t_p = 1$; $u = 252$; $u' = 251$; $d = 1$; $d = t_p$; $U = u' + D = 251 + 2(1) + 0 = 253$ (pixel after first expanded)
- $(D = 2) > t_p$; $U = u' + D + t_p + 1 = 251 + 2 + 1 + 1 = 255$ (pixel after second shifted) (from E set to ES set).

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

Fig. A-1 Graph for checking status of $d_{i,j}$ from 7 sets for an encoder. Green arrows indicate the status that has no problem, orange arrows indicate the status that seem to be good for embedding data but cannot be used and the red arrows show the status that has problem from the beginning.

Fig. A-2 Graph for checking status of $D_{i,j}$ from 7 sets for a decoder. Green arrows indicate the status that has no problem, orange arrows indicate the status that seem to be good for embedding data but cannot be used and the red arrows show the status that has problem from the beginning.

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.