JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 36, 621-641 (2020)
DOI: 10.6688/JISE.202005_36(3).0009

Selection of Embedding Area: A Better Way to Use Prediction-Error Expansion Method for Reversible Hiding

CHE-YI CHAO AND JA-CHEN LIN
Department of Computer Science
National Chiao Tung University
Hsinchu, 300 Taiwan
E-mail: dpm@ms22.hinet.net; jclin@cs.nctu.edu.tw

Reversible data hiding is widely used because the host image can be recovered without errors after the extraction of hidden data. One of the popular schemes for reversibility involves the use of prediction-error expansion (PEE). Scholars often modify the basic PEE scheme to hide more secret bits or to improve the quality in stego images. Examples include the pairwise PEE, the difference expansion approach, and others. In our PEE-based method here, by identifying which parts of a prediction error histogram indicate inefficient hiding, we propose increasing the ratio of pixels hiding data to the pixels shifted without hiding data, called the efficiency ratio (ER). We used four tests to improve ER and hence improve the quality of stego images. The basic concept of our method is to check whether an image block or pixel is suitable for embedding data. After deleting the blocks or pixels that are likely to yield erroneous predictions, we can reduce the chance of a high PE. A high PE not only deteriorates the quality of stego images, but also contributes nothing to the embedding capacity. As shown in experiments, our image quality is better than that of many other PEE-based algorithms when similar amounts of data are hidden.

Keywords: reversible hiding, prediction-error expansion (PEE) methods, prediction-error (PE) histogram, efficiency ratio (ER), selection of embedding area

1. INTRODUCTION

The protection of secrets is always an interesting issue. While classical cryptography is the concealing of a message's content, steganography is the concealing of its existence [1]. As stated in [2], steganography is an important branch of information hiding. Nowadays, steganography is utilized to hide secrets in media such as audio [3], video [4], and images [5-7].

A very simple approach to image-based hiding is the least-significant-bits (LSB) substitution method or its modulus-based extension [5], and this approach typically achieves good quality-capacity performance. However, after lossless extraction of hidden data, the original host image can never be recovered from the stego image using this method. By contrast, reversible hiding (RH) allows the recovery of the original image from a stego image after the extraction of the hidden content. This reversible individuality is attractive in many fields, such as for military, legal, and medical applications [8].

Honsinger *et al.* [9], Thodi and Rodriguez [10], Tian [11], Sachnev [12], and many other scholars have proposed reversible methods. One of the popular reversible approaches is histogram-based. For instance, the reversible method of Ni *et al.* [6] uses histogram modification to determine the peak point to hide messages; whereas the reversible method

Received October 14, 2018; accepted January 26, 2019.
Communicated by Chung-Lin Huang.

of Thodi and Rodriguez [10] expands the difference between the real pixel value and its prediction value in order to embed more data. In histogram-related approaches, a histogram with peaks is usually better than a flat histogram, for the former can yield both larger embedding capacity (EC) and higher PSNR values for stego images. As stated in [8], Prediction-Error Expansion (PEE) methods (for example, [10]) often give more useful histograms because the Prediction-Error histograms created by PEE often have peak frequency at center, which in turn gives good stego image quality.

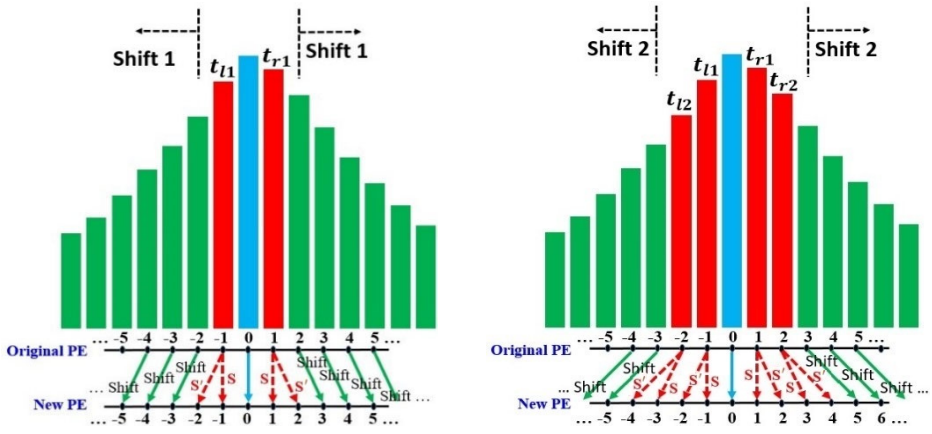
Hereinafter, the term prediction error (*PE*) is defined as the original pixel value minus prediction value of that pixel. In this formula, the prediction value is evaluated using the information of some neighboring pixels. In the design of PEE-based reversible hiding, to improve quality of stego images, researches usually focus on the design of good predictors, because better prediction yields smaller impact to the stego images [12-15]. In this paper, we use a new approach: we will focus on the improvement of stego image quality through determining the suitability of hiding for different areas of a given host image. More precisely, we apply four tests to filter out the portions which are likely to yield bad prediction or hiding. Then, we only hide content in the areas that pass the tests.

The paper is organized as follows: Section 2 briefly introduces PEE. Section 3 gives the proposed algorithm, which includes four phases to increase the performance. The experimental results, as well as the comparison with previous works, are shown in Section 4. The technique dealing with large size secrets and the protection of sensitive secrets is in Section 5. Alternative versions of the design are discussed in Section 6. Conclusions are in Section 7.

2. BACKGROUND

This section briefly reviews the idea of reversible hiding using PEE. Fig. 1 (a) shows a *PE* histogram that counts the number of pixels whose prediction error is the value shown on the upper horizontal line prior to shifting. The shifting of *PE* values is required in order to hide data, which is represented by the shift in values in the upper horizontal line to values in the lower horizontal line of Fig. 1 (a). In Fig. 1 (a), when the *PE* of a pixel is 0, no hiding is performed. However, if the *PE* of a pixel is 1 or −1, then one bit can be hidden. In other words, if the bit to be hidden has a value of 0, then the PE value of the pixel being discussed is not modified. Hence, the *PE* of this pixel is still 1 or −1. However, if the secret bit to be hidden is 1, then modify the pixel value of the pixel being discussed so that the new *PE* of the new pixel value is 2 or −2 (see the s′ in Fig. 1 (a)). By doing this modification, the decoder can understand that the hidden secret is 1 rather than 0, when the decoder sees that the *PE* is ±2 (rather than ±1).

Regarding decoding after using an encoding system, for Fig. 1 (a), a *PE* value in the range {−2, −1, 0, 1, 2} indicates that the hidden secret is identified as {1, 0, no-secret, 0, 1}, respectively. All other *PE* values not in {−2, −1, 0, 1, 2} indicate that no secret is hidden in the current stego pixel. Also note that, for Fig. 1 (a), to avoid errors in the decoding caused by non-uniqueness, and also for the purpose of reversible recovery of the original values, a shift is needed when the original *PE* value differs by 2 or more. Hence, these *PE* values are shifted to get new *PE* values, as indicated by the green arrows in Fig. 1. These green arrows indicate that those pixels contain no secret, and the original *PE* must be shifted by a constant unit (*T* unit) to avoid errors caused by non-uniqueness.



(a) (b)

Fig. 1. Adjustment of *PE* values in conventional PEE. The shift is $T = 1$ in (a); and $T = 2$ in (b).

In Fig. 1, ***s*** indicates that the hidden secret is 0; ***s′*** indicates that the hidden secret is 1. The blue arrow indicates that there is no shift, and therefore no hiding of a secret, whereas a green arrow indicates that there is a shift of *PE* value, though still without the hiding of a secret. To investigate if it is too wasteful that only very few *PE* values are utilized in Fig. 1 (a) to hide secret, [8] also tried using larger values such as those shown in Fig. 1 (b). While Fig. 1 (b) is similar to Fig. 1 (a), as mentioned in [8], the "shift amount" *T*, which is an integer parameter to control embedding capacity, is $T = 2$ rather than $T = 1$. In general, as the *T* value increases, the hiding capacity increases while the visual quality of the stego image decreases. $T = 1$ in Fig. 1 (a), so one bit can be hidden when the original *PE* of a pixel is ±1. $T = 2$ in Fig. 1 (b); thus, one bit can be hidden when the original *PE* of a pixel is ±1 or ±2. Again, a shift of *PE* value might be needed. In Fig. 1 (b), after shifting, when the new *PE* value is ±1 or ±3, then the hidden secret bit is 0; when the new *PE* value is ±2 or ±4, then the hidden secret bit is 1.

In summary, in Fig. 1 (a), where *T* is set to 1, a secret is embedded when the original *PE* value is $t_{r1}(= 1)$ or $t_{l1}(= -1)$. Moreover, note that when the original *PE* is larger than $t_{r1}(= 1)$ or smaller than $t_{l1}(= -1)$, the *PE* value is always forced to shift one unit, because $T = 1$, even though no secret is hidden. Analogously, in Fig. 1 (b), where *T* is set to 2, a secret is embedded when the original *PE* is equal to $t_{r1}(= 1)$, $t_{r2}(= 2)$, $t_{l1}(= -1)$, or $t_{l2}(= -2)$. Note that when the original *PE* is larger than $t_{r2}(= 2)$ or smaller than $t_{l2}(= -2)$, the *PE* value is always forced to shift two units, because $T = 2$, even though no secret is hidden. Because more host pixels are utilized to hide one bit each, Fig. 1 (b) can hide more secret than Fig. 1 (a); however, the PSNR using Fig. 1 (b) is worse. If the image quality of a stego image is a major concern, then people often use the scheme of Fig. 1 (a).

3. PROPOSED METHOD

Our embedding process has four phases (see Fig. 2). The aim of Phase 1, indicated by the dashed line rectangle in the left half of Fig. 2, is to find and delete the blocks which are

unsuitable for hiding data. Then, for each block which is not deleted in Phase 1, we use Phases 2-4 to check the suitability of the pixels in this block. Finally, secrets are embedded only in the pixels that pass the tests in these phases. The embedding process is run for two iterations of Phases 2-4, as shown in the right half of Fig. 2. The first round deals with (predicts) the pixels (i, j) whose $i+j$ is even, and the second round deals with (predicts) the pixels (i, j) whose $i+j$ is odd, which are referred to as the "black" pixels and "white" pixels, respectively (see Fig. 4). This is because, in Fig. 4, the prediction of white pixels uses the values of black pixels, and vice versa.

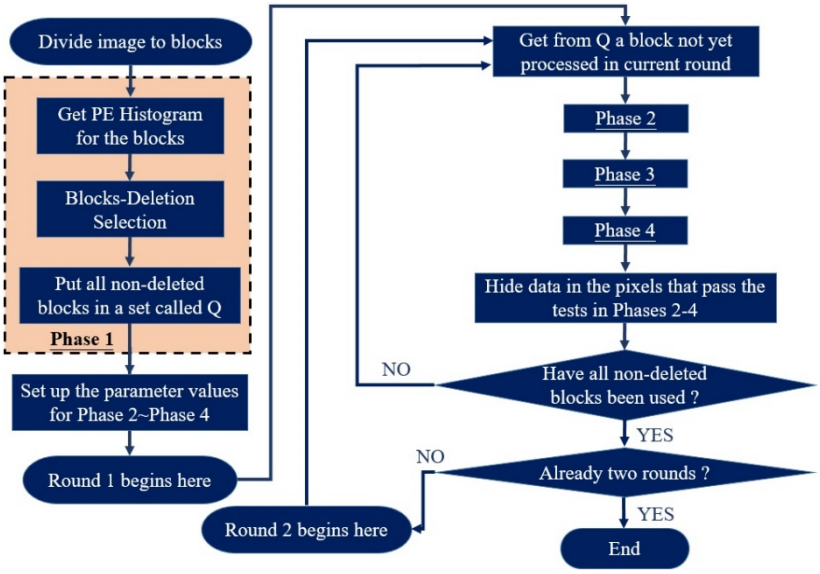
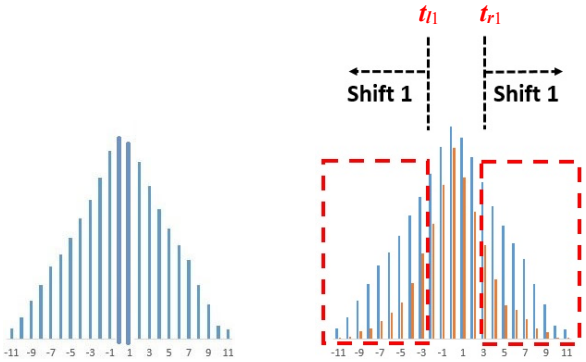


Fig. 2. The embedding process of our algorithm.

As mentioned in [16], the *PE* histograms of natural images are often similar to Gaussian distribution to a certain extent (see the blue bars in Fig. 3 (a)), with center being the point where the *PE* value is zero. Usually, if a *PE* histogram is more centralized around zero, then the average $|PE|$ is smaller, and the embedding capacity is also larger. Without the loss of generality, assume the "shift amount" T is 1, *i.e.* we use Fig. 1 (a) to illustrate the idea. Given that *PE* values are integer values $\{\ldots, -4, -3, -2, -1, 0, 1, 2, 3, 4, \ldots\}$, we may partition *PE* values as a 5-interval union: $\{PE < t_{l1}\} \cup \{PE = t_{l1}\} \cup \{t_{l1} < PE < t_{r1}\} \cup \{PE = t_{r1}\} \cup \{t_{r1} < PE\}$. Here we call the central interval $\{t_{l1} < PE < t_{r1}\}$ as "central-bin". The leftmost interval $\{PE < t_{l1}\}$ and rightmost interval $\{t_{r1} < PE\}$ are called the two "side lobes". $PE = t_{l1}$ and $PE = t_{r1}$ are called "expansion-bins" in some published papers (*e.g.*, [17]).

Referring to the earlier explanation of Fig. 1, we know that the side lobes not only hide nothing, but also deteriorate the quality of stego images due to shifting. Our method tries to reduce the chance using image area which has many pixels in the two side lobes. In Fig. 3 (a) and Fig. 3 (b), the blue bars show the original distribution of prediction error. In Fig. 3 (b), we use several techniques to reduce the chance of using bad pixels such as

those pixels in the two side lobes. After our examination, the *PE* distribution of the qualified pixels is shown by the distribution of red bars in Fig. 3 (b). The number of poor prediction pixels is reduced (see the two side-lobes area enclosed by the two dash rectangles in Fig. 3 (b)). As for pixels in central-bin, they are harmless to the quality of stego-images. In general, the red bars in Fig. 3 (b) are in good embedding situation if their distribution is narrowed down to a few red bars located near the place $PE = 0$. Section 3.2 below lists four techniques to form the four phases of Fig. 2, and these four phases provide good environment for embedding. In these phases, we need a prediction formula.



(a) (b)

Fig. 3. Histogram of *PE*; (a) Blue bars are original distribution of prediction error; (b) Red bars indicate pixels in suitable embedding areas after filtering out and deleting pixels or blocks with large *PE*; The two red rectangles indicate the two side lobes.

3.1 A Simple Prediction Formula and the Definition of the Efficiency Ratio

Every PEE method needs a prediction rule to predict pixel values. Here we use a very simple and common rhombus pattern to predict pixel values (see Fig. 4). A rhombus pattern is a chessboard-like matrix [12, 14]. The black and white pixels in Fig. 4 belong to two different sets. In this kind of prediction scheme, both decoding and encoding will need two rounds which are reflected in the two rounds of Fig. 2. Round 1 manages the black pixels (Set 1), and Round 2 manages the white pixels (Set 2). When people run the first round, all white pixels are locked so that black pixels can be predicted. When people run the second round, all black pixels are kept still so that white pixels can be predicted. To illustrate that our four-phased hiding method can work even if the prediction rule is not of very high precision, we only use a very ordinary prediction equation to predict values, de-

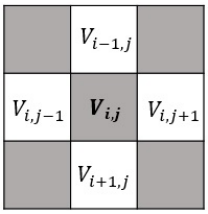


Fig. 4. The very simple prediction pattern used in this study.

fined as Eq. (1). Let $V_{i,j}$ represent the true value of a pixel. Its prediction value $V'_{i,j}$ can be computed by averaging its four surrounding pixels $\{V_{i,j\pm1}, V_{i\pm1,j}\}$ as below:

$$V'_{i,j} = (V_{i,j-1} + V_{i+1,j} + V_{i,j\pm1}, V_{i,j+1} + V_{i-1,j})/4. \quad (1)$$

Our PEE method also needs an evaluation tool called Efficiency Ratio (*ER*) to judge the goodness of an image or a block. For an image (or the union of some pixels of an image), its *ER* is defined as

$$ER = \frac{\text{Number of pixels which really embed some secret data}}{\text{Number of pixels which are shifted without hiding any secret}} \quad (2)$$

A larger *ER* value indicates not only better hiding ratio, but also better PSNR value because the denominator counts the number of "garbage pixels" which are distorted, yet are unsuitable for hiding. The *PE* of the pixels belonging to the denominator must be in two side lobes, and hence satisfy $PE > t_{r1} = 1$ or $PE < t_{l1} = -1$ if $T = 1$ (or, $PE > t_{r2} = 2$ or $PE < t_{l2} = -2$ if the parameter T is $T = 2$, as in Fig. 1 (b)). For example, those pixels in the edge area or texture area often lead to poor prediction, and hence also lead to high distortion in the summation terms of mean square error (MSE) equation defined as Eq. (3). We use Eqs. (1) and (2) to produce Table 1. As expected, smooth images have higher *ER* values. In this paper, we try to produce high *ER* value by erasing bad blocks or bad pixels whose *PEs* are in the two side lobes. Among the four proposed phases, Phase 1 is for the deletion of blocks, whereas Phases 2-4 are for the deletion of pixels.

Table 1. *ER* values for some 512 × 512 images (assuming $T = 1$, as in Fig. 1 (a)).

Images	Lena	Baboon	Airplane	Elaine	Lake	Boat	Barbara
ER	0.463	0.093	0.623	0.164	0.182	0.191	0.237

To evaluate the stego image quality, we may use PSNR. When a host image I of size $M \times N$ pixels is modified and becomes stego image I', the definition of PSNR is

$$PSNR = 10 \times \log\left(\frac{255^2}{MSE}\right); \text{ and } MSE = \left(\frac{\sum_{i=1}^{M}\sum_{j=1}^{N}(I(i,j) - I'(i,j))^2}{M \times N}\right). \quad (3)$$

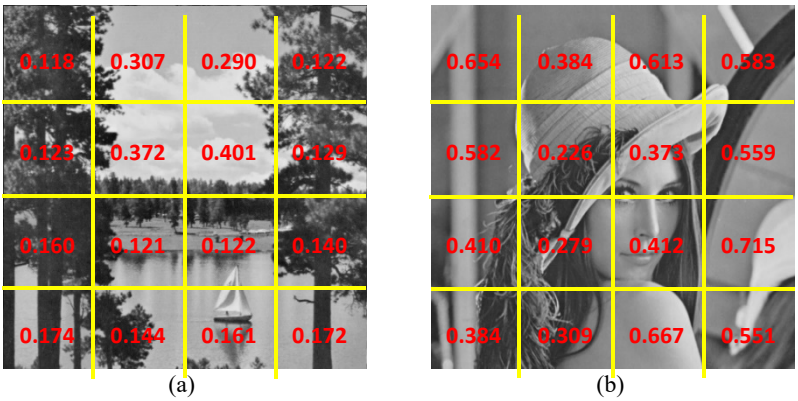
3.2 The Main Process

Firstly, divide the host image into non-overlapping blocks. Then, as shown in Fig. 2, use Phase 1 to delete non-suitable blocks. Then, for each suitable block, run Phases 2-4 to find good pixels for embedding data. The details of the four phases are as follows.

3.2.1 Phase 1: Deletion of bad blocks

The more complex the image, the worse the prediction. Taking Lena and Baboon for example, if we use PEE to hide secrets, we can often embed more secrets in Lena than in

Baboon. The published methods showed that the PSNR of Lena is higher than the PSNR of Baboon when the former hides 20,000 bits and the latter merely hides 10,000 bits [8, 21]. This is because the pixel values are more uniform in Lena than in Baboon, and this feature makes the prediction in Lena more precise. In fact, even in the same image, some blocks are more useful than others. Fig. 5 takes Lake and Lena as examples, and shows that the *ER* varies a lot between blocks and also between images. If the secret file is not large in size, then we should avoid hiding secrets in blocks with high $|PE|$, for these blocks always have low *ER* values. Therefore, we introduce Phase 1, a rough inspection to delete bad blocks. In Phase 1, we delete blocks in which too many pixels have poor prediction ($|PE| > T_{Phase1}$), where T_{Phase1} is a threshold selected by user. Notably, Phase 1 lets us avoid blocks with poor prediction, and hence can improve *ER*, which in turns can improve the stego image quality by improving the PSNR value. Table 2 in Section 4 will show the *ER* improvement caused by deleting the worst blocks. Although the *ER* improvement is not as impressive as other phases, Phase 1 is still needed for the reason explained in the conclusion section.



(a) (b)

Fig. 5. *ER* values for each 128×128 block of the 512×512 image Lake (or Lena). Assuming $T = 1$.

3.2.2 Phase 2: Deleting a pixel (i, j) which is surrounded by some edge pixels

Here we use the neighborhood gray value variation [18] for edge detection. If a pixel (i, j) has too many neighbors which are in edges themselves, then pixel (i, j) cannot get good prediction. Phase 2 checks whether the neighbors are edge pixels, whereas Phase 3 checks whether the pixel (i, j) itself is in an edge.

We will use a simple function to detect edge. Because of our prediction scheme, we need to carefully define the edge detection function. The Canny algorithm [19] used several filters to detect edges. The Canny 3×3 convolution mask is shown in Fig. 6 (a). Here we must modify the mask because our coding sequence is top to bottom and left to right, and also because we have two rounds to predict black pixels and white pixels separately. If we handle the middle pixel in Fig. 6, the two pixels on the left top and right top are already changed to new values. If we include these two values in mask, encoding and decoding will not get the same value. Hence, our mask is Fig. 6 (b) rather than conventional Fig. 6 (a).

−1	0	+1
−2	0	+2
−1	0	+1

Gx

+1	+2	+1
0	0	0
−1	−2	−1

Gy

(a)

0	0	0
−2	0	+2
0	0	0

Gx′

0	+2	0
0	0	0
0	−2	0

Gy′

(b)

Fig. 6. Modifying the conventional mask (a) to get mask (b) for edge detection.

		E_1		
	E_2		E_8	
E_3		$V_{i,j}$		E_7
	E_4		E_6	
		E_5		

Fig. 7. The eight same-polarity (black vs. white) neighbors $\{E_1$ to $E_8\}$ used in Phase 2 for a pixel (i, j) whose gray value is $V_{i,j}$; Notably, pixels $\{E_1$ to $E_8\}$ are at positions $(i\pm1, j\pm1)$, $(i, j\pm2)$, $(i\pm2, j)$.

Steps in Phase 2 (to check whether a pixel (i, j) is surrounded by some edge pixels):

As shown in Fig. 7, let $\{E_1$ to $E_8\}$ be the eight black pixels in these specified positions surrounding pixel (i, j). Let V denote the gray value. Then define:

$e(E_k) = |V$(left white neighbor of $E_k) - V$(right white neighbor of $E_k)|$ +
$|V$(upper white neighbor of $E_k) - V$(lower white neighbor of $E_k)|$.

If $e(E_k) >$ threshold T_e then we consider E_k to be an edge pixel.

Count how many of the 8 points in $\{E_1, E_2, \ldots, E_8\}$ are edge pixels.

If the count is less than the threshold N_t, then let pixel (i, j) pass Phase 2.

Else, consider pixel (i, j) to be in busy area and hence judge pixel (i, j) as "disqualified" for hiding, and go to check the next pixel.

–END of Phase 2–

The *ER* value improvements caused by Phase 2 will be shown later in Table 3 of Section 4. In Phase 2, T_e is a threshold value used to define edge pixels, and N_t is a positive integer threshold less than 8.

3.2.3 Phase 3: Deleting a pixel which is an edge pixel itself

This Phase tests if pixel (i, j) itself is an edge pixel. The *ER* value improvements caused by Phase 3 will be shown in Table 4. For the steps below, recall that V denotes gray value.

Steps in Phase 3 (to check whether a pixel (i, j) itself is an edge pixel):

If $|V_{i,j-1} - V_{i,j+1}| + |V_{i-1,j} - V_{i+1,j}| >$ threshold (T_{se}), then pixel (i, j) is an edge pixel, so judge pixel (i, j) as "disqualified" for hiding, and check the next pixel.

Else, let pixel (i, j) pass Phase 3.

–END of Phase 3–

3.2.4 Phase 4: Deleting a pixel which has many neighbors with bad prediction

Like Phase 2, this phase also evaluates the characteristics of neighbor pixels. Because we always choose the pixels with small $|PE|$ to hide secrets, here we use the PE of surrounding pixels to forecast whether the pixel (i, j) currently being examined can get good prediction. In Fig. 8, pixel (i, j) has 4 surrounding pixels {A, B, C, D}. Each is a black pixel and hence each can be roughly predicted by the 4 white pixels nearest to this black pixel. When handling pixel (i, j), the four pixels {A, B, C, D} are still the original values in the encoding procedure, or were already recovered back to original values in the decoding procedure. However, the four pixels {A′, B′, C′, D′} are modified values regardless of whether the procedure is encoding or decoding. For better judgement, we do not use the four unreliable pixels {A′, B′, C′, D′} as our reference pixels when dealing with pixel (i, j) in Phase 4. There are two thresholds in Phase 4. GN_t is a threshold of whether the number of good neighbors (GN, defined as the neighbor pixels with $|PE| \le T_{pe}$) is enough, and T_{pe} is a threshold to limit $|PE|$. The ER value improvements caused by Phase 4 are in Table 5.

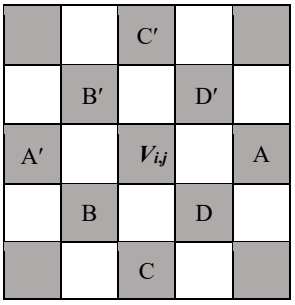


Fig. 8. The 8 – 4 = 4 neighbors {A, B, C, D} for Phase 4. (Do not use {A′, B′, C′, D′}, for the reason explained in text.)

Steps of Phase 4 (to check whether a pixel (i, j) is surrounded by some good-prediction pixels):

Define the prediction quality bit Q_A of the neighbor pixel A as $Q_A = 1$ iff the Prediction Error (PE) of pixel A satisfies $(|PE|)_A \le T_{pe}$. (Hence, $Q_A = 0$ if $(|PE|)_A > T_{pe.}$) Define Q_B, Q_C, and Q_D likewise.

For pixel (i, j), if $(Q_A + Q_B + Q_C + Q_D) \ge GN_t$, then let pixel (i, j) pass the test, for pixel (i, j) is in a good prediction region.

Else, judge pixel (i, j) as "disqualified" for hiding, then go to check the next pixel.

3.2.5 Hiding data using prediction-error expansion

According to Fig. 2, the pixels that pass all of the phases will enter the hiding step. Because Phases 1-4 already filter out the pixels that might not be suitable to hide secret by using PEE, our algorithm can just use the original PEE method without further modification. Eq. (4) describes the encoding rules. Here, m is a to-be-embedded bit (0 or 1)

described in Fig. 1. To get stego images with low distortion, we use $T = 1$ and $e_{i,j}$ is the original *PE* ($e_{i,j} = V_{i,j} - V'_{i,j}$). Each *PE* can be expanded (*i.e.* shifted) as $e'_{i,j}$. Let $t_{l1} = -1$ and $t_{r1} = 1$ for the two expansion bins.

$$e'_{i,j} = \begin{cases} e_{i,j} - 1 & \text{if } e_{i,j} < t_{l1} \\ e_{i,j} - m & \text{if } e_{i,j} = t_{l1} \\ e_{i,j} & \text{if } t_{l1} < e_{i,j} < t_{r1} \\ e_{i,j} + m & \text{if } e_{i,j} = t_{r1} \\ e_{i,j} + 1 & \text{if } e_{i,j} > t_{r1} \end{cases} \tag{4}$$

We then get stego pixel value $V''_{i,j} = e'_{i,j} + V'_{i,j}$ to replace the original pixel value $V_{i,j}$ of input image. In decoding, any pixel that fails to pass the four phases means that there is no secret embedded in it. Otherwise, we can use Eq. (5) to get the embedded secret bit m and also recover the original pixel value. In detail, the first step is to get the prediction value $V'_{i,j}$ using the same prediction formula that we used in encoding. Then, we can get $e'_{i,j}$ by the equation $e'_{i,j} = V''_{i,j} - V'_{i,j}$. The secret bit value m and original pixel value $V_{i,j} = e_{i,j} + V'_{i,j}$ can then be revealed using Eq. (5) below.

$$e_{i,j} = \begin{cases} e'_{i,j} + 1 & \text{if } e'_{i,j} < (t_{l1} - 1) \quad \textit{No secret} \\ e'_{i,j} + m & \text{if } e'_{i,j} = t_{l1}(m = 0 = \textit{secret}) \text{ or if } e'_{i,j} = t_{l1} - 1(m = 1 = \textit{secret}) \\ e'_{i,j} & \text{if } t_{l1} < e'_{i,j} < t_{r1} \quad \textit{No secret} \\ e'_{i,j} - m & \text{if } e'_{i,j} = t_{r1}(m = 0 = \textit{secret}) \text{ or if } e'_{i,j} = t_{r1} + 1(m = 1 = \textit{secret}) \\ e'_{i,j} - 1 & \text{if } e'_{i,j} > (t_{r1} + 1) \quad \textit{No secret} \end{cases} \tag{5}$$

3.3 Auxiliary Information to Be Carried

For reversible methods, certain auxiliary information should also be embedded in the cover image as a part of the payload for extraction and restoration. For the phases of our method, the following auxiliary information are required:

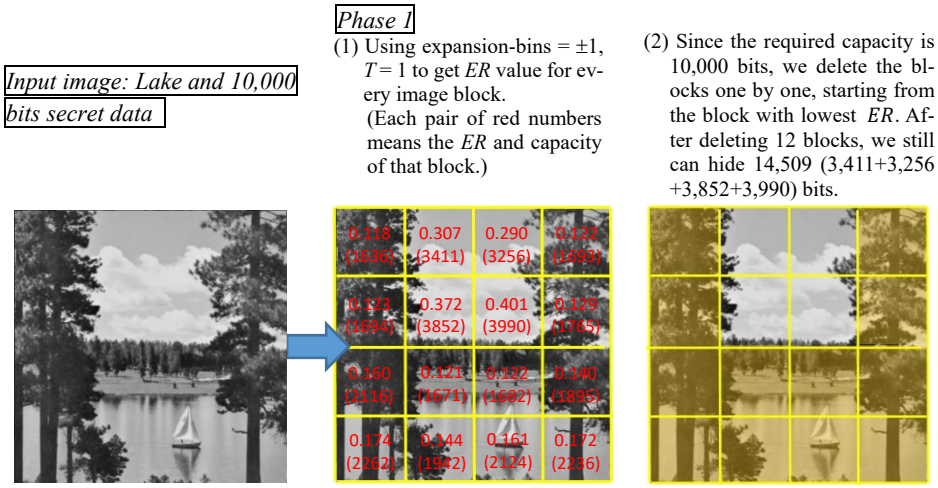
For PEE: Left bin (3 bits) and right bin (3 bits), as their absolute values are less than 8.
For Phase 1: A string to record which blocks are bypassed. Hence, for all 16 blocks of the image combined, we used a 16-bit string. The i^{th} bit is 0 iff i^{th} block is bypassed without hiding.
For Phase 2: T_e (5 bits) and N_t (3 bits)
For Phase 3: T_{se} (5 bits)
For Phase 4: T_{pe} (3 bits) and GN_t (2 bits)

As a summary, at least 3+3+16+5+3+5+3+2 = 40 bits are needed. The extra information can be embedded into the least significant bits (LSBs) of the first 40 pixels and put the real LSBs into the secret so that we can recover original host image after decoding.

3.4 An Example Illustrating the Embedding Process

Assume we need to store 10,000 bits in Lake image. Fig. 9 demonstrates an example of our proposed method. Note that Tables 2-5 are for each phase being used "alone", whereas Fig. 9 is for the four phases being stacked together. Typical initial trying parameter values are as follows (assuming capacity is 10,000 bits).

Phase 2: $T_e = 8$, $N_t = 4$
Phase 3: $T_{se} = 30$
Phase 4: $T_{pe} = 4$, $GN_t = 3$



Parameters setting for Phase 2-Phase 4:

Phase 2
(1) $T_e = 8$, $N_t = 4$
Capacity = 13,246 bits, Average $ER = 0.438$
(2) $T_e = 7$, $N_t = 4$
Capacity = 13,012 bits, Average $ER = 0.452$
(3) $T_e = 6$, $N_t = 4$
Capacity = 12,466 bits, Average $ER = 0.470$

Since 12,466 bits are already close to 10,000 bits, stop Phase 2 here and let $T_e = 6$ and $N_t = 4$. This is because Phase 3 and Phase 4 will cut down capacity further.

Phase 3
(Using the final parameter values $T_e = 6$, $N_t = 4$ of Phase 2)
(1) $T_{se} = 30$
Capacity = 12,382 bits, Average $ER = 0.480$
(2) $T_{se} = 25$
Capacity = 12,368 bits, Average $ER = 0.483$

Stop Phase 3 here because no obvious change of ER.

Phase 4
(Using the final parameter values $T_e = 6$, $N_t = 4$ of Phase 2. $T_{se} = 25$ of Phase 3)
(1) $T_{pe} = 4$, $GN_t = 3$
Capacity = 12,057 bits, Average $ER = 0.504$
(2) $T_{pe} = 3$, $GN_t = 3$
Capacity = 11,612 bits, Average $ER = 0.529$
(3) $T_{pe} = 2$, $GN_t = 3$
Capacity = 10,486 bits, Average $ER = 0.578$

Stop Phase 4 here because capacity is already very close to secret data size (10,000 bits). So the final paremeter values are:
$T_e = 6$, $N_t = 4$,
$T_{se} = 25$,
$T_{pe} = 2$, $GN_t = 3$.

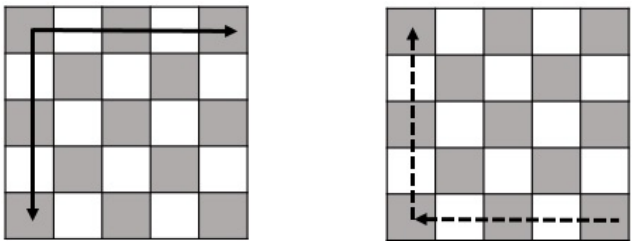
Then use the determined parmeters to run Phase 2, then run Phase 3, then run Phase 4.

Fig. 9. An example illustrating the embedding process. (Capacity: 10,000 bits)

Finally, a systematic way to determine the threshold values is according to the required embedding capacity and the content of the host image. For smooth images (ER > 0.3) or for lower capacity (capacity < 10,000 bits), we use values smaller than typical values as the initial test threshold values. On the contrary, for texture images or larger size secret, we use larger values as initial test values of the thresholds.

3.5 Extraction

The extraction and recovery procedure use the inverse of the steps of the encoding procedure. The first step of extracting is getting the auxiliary information from the LSBs of the first 40 pixels. As shown in Fig. 10, the decoding sequence (to process pixels) is of the opposite direction of the encoding sequence. Given that our encoding procedure encoded the black pixels prior to the white pixels, our decoding procedure should decode all white pixels first.



(a) Encoding sequence. (b) Decoding sequence.

Fig. 10. The encoding sequence and decoding sequence of our algorithm.

4. EXPERIMENTAL RESULTS

In this section, we conduct several experiments to demonstrate the performance of our method. Standard 512 × 512 gray-scale images including Lena, Baboon, Airplane, Elaine, Lake, Boat, and Barbara are utilized in our experiments. Tables 2-5 show the ER value improvements caused by each individual phase, whereas Tables 6-8 list the integrated results of using all four phases. Table 3 shows that after the filtering of Phase 2, the ER values increase more than two times on average. In Phase 2, as shown in Table 3, to get larger ER values we usually use a smaller T_e value and a smaller N_t value. Of course, the T_e value still cannot be too small, or it will become impractical. For example, if $T_e = 0$, then $Count_{i,j} = 8$, so all pixels are in a busy area, and hence no pixels can pass the test. This makes the entire host have no qualified pixels in which to hide secrets. Likewise, if $N_t = 0$, then all pixels are in a busy area, and no pixel can pass the test. In Tables 4 and 5, to get larger ER values, we usually use a smaller T_{se} value, a smaller T_{pe} value, and a larger GN_t value.

Because of our policy of deleting low ER regions, our good performance over that of other methods is more obvious in dealing with secrets of small size. The following table shows our result when compared with [12, 14] when the secret has only 7,000 bits. On average, our 62.3804 dB PSNR is better than the 59.1453 dB PSNR of [12] and the 59.8855 dB PSNR of [14]. Other references did not list experiments of 7,000 bits, and hence are

Table 2. *ER* values after applying Phase 1 alone to the host images (when T=1, as in Fig. 1 (a)).

	Lena	Baboon	Airplane	Elaine	Lake	Boat	Barbara
Original *ER* in Table 1	**0.463**	**0.093**	**0.623**	**0.164**	**0.182**	**0.191**	**0.237**
After deleting two 128× 128 blocks of smallest *ER*	0.503	0.103	0.701	0.175	0.192	0.200	0.259
After deleting Four 128 ×128 blocks of smallest *ER*	**0.530**	**0.115**	**0.801**	**0.181**	**0.204**	**0.209**	**0.283**

Table 3. *ER* values after applying Phase 2 alone to the host images (when T=1, as in Fig. 1 (a)).

	Lena	Baboon	Airplane	Elaine	Lake	Boat	Barbara
Original *ER* in Table 1	**0.463**	**0.093**	**0.623**	**0.164**	**0.182**	**0.191**	**0.237**
ER (if $T_e = 7$, $N_t = 4$)	0.624	0.208	1.091	0.254	0.296	0.270	0.591
ER (if $T_e = 6$, $N_t = 4$)	0.645	0.213	1.166	0.281	0.319	0.285	0.634
ER (if $T_e = 7$, $N_t = 3$)	0.684	0.242	1.300	0.361	0.363	0.323	0.732
***ER* (if $T_e = 6$, $N_t = 3$)**	**0.705**	**0.243**	**1.398**	**0.421**	**0.406**	**0.350**	**0.802**

Table 4. *ER* values after applying Phase 3 alone to the host images (when T=1, as in Fig. 1 (a)).

	Lena	Baboon	Airplane	Elaine	Lake	Boat	Barbara
Original *ER* in Table 1	**0.463**	**0.093**	**0.623**	**0.164**	**0.182**	**0.191**	**0.237**
ER (if $T_{se} = 10$)	0.617	0.191	1.181	0.232	0.286	0.259	0.583
***ER* (if $T_{se} = 5$)**	**0.669**	**0.212**	**1.492**	**0.307**	**0.356**	**0.306**	**0.748**

Table 5. *ER* values after applying Phase 4 alone to the host images (when T=1, as in Fig. 1 (a)).

	Lena	Baboon	Airplane	Elaine	Lake	Boat	Barbara
Original *ER* in Table 1	**0.463**	**0.093**	**0.623**	**0.164**	**0.182**	**0.191**	**0.237**
ER (if $T_{pe} = 5$, $GN_t = 1$)	0.498	0.148	0.712	0.187	0.210	0.214	0.375
ER (if $T_{pe} = 3$, $GN_t = 1$)	0.534	0.167	0.793	0.215	0.240	0.235	0.431
ER (if $T_{pe} = 5$, $GN_t = 2$)	0.544	0.191	0.821	0.233	0.258	0.245	0.452
***ER* (if $T_{pe} = 3$, $GN_t = 2$)**	**0.602**	**0.215**	**0.967**	**0.312**	**0.335**	**0.295**	**0.550**

Table 6. Comparison of PSNR (in dB) between the proposed method and other published methods, assuming 10,000 bits are to be hidden.

Images	[12]	[16]	[20]	[21]	[8]	[22]	Our method		
							16 blocks	64 blocks	**256 blocks**
Lena	58.19	57.94	60.28	59.78	59.75	59.64	61.49	61.60	**61.79**
Baboon	54.16	51.44	55.49	53.96	55.21	55.63	55.75	55.81	**55.97**
Airplane	60.38	59.79	63.19	63.18	63.76	61.83	63.96	64.08	**64.19**
Elaine	56.14	54.62	56.72	57.39	58.06	57.13	57.92	58.08	**58.40**
Lake	56.66	54.85	57.57	58.08	58.72	57.48	59.24	59.36	**59.46**
Boat	56.15	55.22	57.49	57.42	57.55	57.23	57.68	57.90	**58.19**
Average	***56.95***	***55.64***	***58.46***	***58.30***	***58.84***	***58.16***	***59.34***	***59.47***	***59.66***

not in Table 7. In Table 8, the secret has 20,000 bits. We combine the experimental results listed in [8, 25] and others, comparing them with our result, and ours are still found to be competitive. Fig. 11 gives the capacity to PSNR curves for our method and the five methods listed in Fig. 7 of [8], which used Lena and Airplane as test images. It can be seen that our method, indicated by black triangles in Fig. 11, has very competitive performance.

Table 7. Comparison of PSNR (in dB) between the proposed method and other published methods, assuming 7,000 bits are to be hidden.

Images	[12]	[14]	Our method		
			16 blocks	64 blocks	**256 blocks**
Lena	60.2253	60.8096	63.1738	63.2206	**63.2853**
Airplane	62.2648	64.2273	65.5621	65.9463	**66.0262**
Baboon	56.5411	56.9626	58.0952	58.3084	**58.3924**
Lake	58.3248	59.2485	61.6466	61.8450	**62.0211**
Boat	57.8449	58.2169	59.7608	59.9579	**60.1508**
Barbara	59.6711	59.8480	63.7182	64.2051	**64.4063**
Average	***59.1453***	***59.8855***	***61.9928***	***62.2472***	***62.3804***

Table 8. Comparison of PSNR (in dB) between the proposed method and other published methods, assuming 20,000 bits are to be hidden.

Images	[12]	[24]	[26]	[27]	[25]	[8]	Our method		
							16 blocks	64 blocks	**256 blocks**
Lena	55.03	54.82	54.92	54.97	56.15	**56.29**	57.95	57.96	**58.04**
Airplane	57.34	56.84	58.58	59.67	59.45	**60.20**	60.23	60.36	**60.41**
Boat	52.65	52.43	52.26	52.37	53.12	**53.34**	53.32	53.39	**53.50**
Barbara	55.04	55.29	54.89	55.18	56.24	**56.27**	57.02	57.12	**57.29**
Average	***55.02***	***54.85***	***55.16***	***55.55***	***56.24***	***56.53***	*57.13*	*57.21*	***57.31***

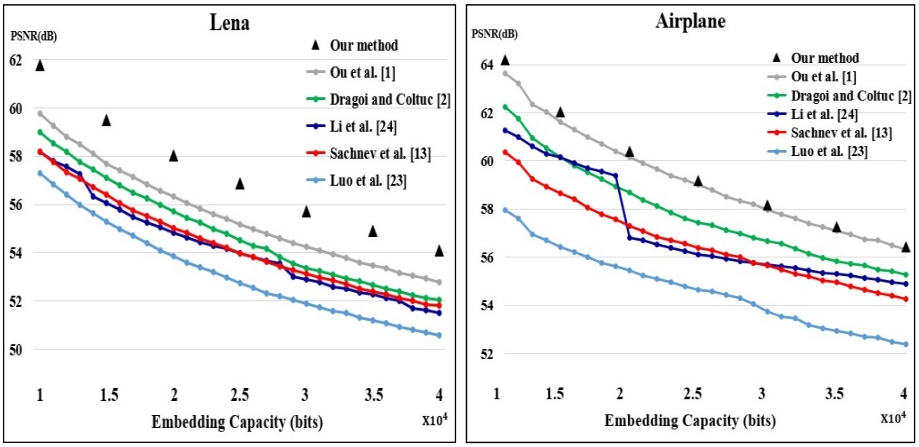


Fig. 11. The capacity to PSNR curves for ours and other methods.

Since the differences between the stego pixel values and the original pixel values are at most 1, human eyes cannot distinguish the differences (see Fig. 12). As for the recovered images, they are identical with the host images.

Host images

Stego images

Recovered images

Fig. 12. Comparison among the host images, the stego images and the recovered images after hiding and extracting 10,000 bits each (Images are Lena, Lake, Elaine and Boat).

5. APPLICATIONS TO THE PROTECTION OF LARGE-SIZED, SENSITIVE SECRETS

In the above, the reversible methods (including our method) listed in Tables 6-8 and Fig. 11 deal with secret data whose size is not large. Section 5 discusses the protection of large-sized secrets. A large secret might be closely related to the given host image or totally independent of that image.

An application example is for the concealment of medical or legal information. For instance, for a given x-ray image, the corresponding patient's name, age, gender, and medical history are very sensitive and must be protected if this information is to be attached to the image for the convenience use of hospital's treatment team. To avoid being charged by the patient, no doctor of the treatment team should be allowed to see any personally identifiable information of the patient (except the lung image), unless the whole treatment

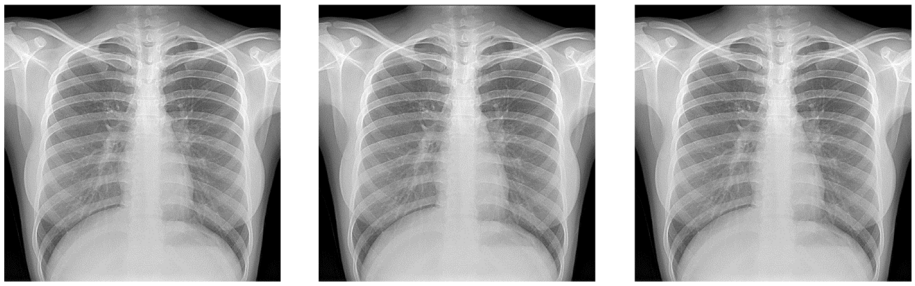
team (or a sufficient number of the members of the team) agree to simultaneously unveil the hidden information S in the treatment meeting.

We can use the sharing technique (see Thien and Lin [28]) to create n shares from the secret S such that any t of the n shares can cooperate to recover S, whereas less than t shares cannot. Here, the value of the positive integer n is arbitrary, whereas the threshold t must satisfy $t \leq n$. According to [28], the size of each share is t times smaller than that of S. Hence, it is easy to hide a share in the host image. Therefore, we can duplicate n copies of the host image, then in each copy we use reversible hiding to hide a share. The created n host images have very high PSNR because each share is t times smaller than original secret S. The protection of the secret S is that:

1) Information security against betrayer: less than t of the n stego images reveal nothing about the secret S.
2) Missing tolerance: up to $n - t$ stego images can be lost or absent in the disclosure meeting.

The protection in 2) is particularly useful if people cannot guarantee that the n holders of the n stego images are loyal to the company for their entire lives, or that the n computers storing the n stego images are always on-line and functional. It also avoids the dilemma of using single stego image. Without sharing, the crash of the single media, which stores the single stego image, will make the secret S lost forever with no way to recover it. Besides erasing the secret forever, this event also deletes the host image (such as Fig. 13 (a)) forever.

Fig. 13 gives an example of this kind of application. Fig. 13 (a) is the original 512 × 512 lung x-ray image of a patient. Fig. 13 (b1) and Fig. 13 (b2) show two of the n created



(a) (b1) (b2)

Patient's name: D. M. Chung; Chart number: 7356-435-556. Blood type: A; Birth: 1960. Weight: 85 Kg. Height: 182 cm. Male, Married. Allergic to medicine: no. Surgery history: no. Smoking: (1 bag per day) for last 16 years. Occupation: working in a road construction company (for last ten years). Patient's parents: Father had cancer (lung and brain), mother is healthy. Observation: cancer was seen on chest radiographs and computed tomography (CT) scans. ……Reminder for analysis or for other teammates of the treatment team: more than 80% of lung cancer cases are caused by smoking in a long period. Especially notice his working condition (air pollution) and smoking history; for this doubled his factors of getting cancer. Also needs to know that the patient was not cooperative (did not take medicine according to schedule or quit smoking completely)……..

(c)

Fig. 13. An application example; (a) The original 512 × 512 lung x-ray image of a patient; (b1-b2) Two of the n created 512 × 512 stego images (each looks like (a)); (c) Any t of the n stego images can be used together to extract S (*i.e.*, medical history) in amounts of 40,000 × t bits which equals t × 5,000 bytes; Each stego image can recover (a) without errors.

512 × 512 stego images (as each stego image looks like Fig. 13 (a)). Any t of the n stego images can be used together to extract S (the patient's name and medical history shown in Fig. 13 (c)). Each stego image can recover image Fig. 13 (a) without errors because the hiding is reversible. Notably, the size of each share is t times smaller than that of S, hence if a single stego image can hide, say, 40,000 bits without heavily distorting the image, then the secret S (Fig. 13 (c)) can be as large as 40,000 × t bits. In the example shown in Fig. 13, the PSNRs of the stego images in Fig. 13 (b) are about 53.71 dB if the secret S (Fig. 13 (c)) has 40,000 × t bits, or 55.33 dB if S has 30,000 × t bits. Note that 40,000 × t bits is 5,000 × t Kilobytes; enough to store a very long medical history of a patient.

6. ALTERNATIVE VERSIONS OF DESIGN

This paper is to illustrate how to use *ER* to delete improper areas so that PSNR can be improved. The readers can use this idea to design his own methods, by using the alternatives of each component of PEE-based hiding. The replacement of components includes the use of other prediction formulas, the use of other kinds of adjustment of *PE* values in PEE, etc. For example, the adjustment of *PE* in Fig. 1 (a) can be replaced by Fig. 14 below. In Fig. 14, there is no central-bin (this kind of *PE* adjustment was called as C-PEE method in [16]). Table 9 shows the *ER* and PSNR values using Fig. 1 (a) and Fig. 14. We can see that using Fig. 14 does not mean better *ER* or PSNR can be obtained, as compared with using Fig. 1 (a).

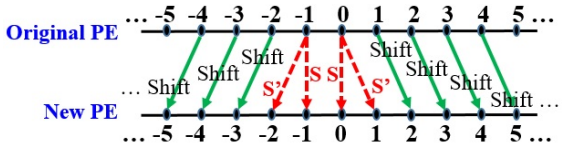


Fig. 14. One of many alternatives for the adjustment of *PE* values in PEE (assuming $T = 1$).

Table 9. Comparison of *ER* and PSNR (in dB) between using two different expansion-bin systems (Fig. 1 (a) vs. Fig. 14).

	ER		PSNR (assuming 10,000 bits are to be hidden.)	
	Using Fig. 1 (a), *i.e.* expansion-bins=–1 &1	Using Fig. 14, *i.e.* expansion-bins=–**1 &0**	Using Fig. 1 (a), *i.e.* expansion-bins=–1 &1	Using Fig. 14, *i.e.* expansion-bins=–**1 &0**
Boat	0.191	**0.176**	57.68	**57.08**
Lake	0.182	**0.167**	59.24	**58.14**

Finally, even the *ER* formula itself can be replaced by other finite-range formulas. One such formula is:

$$ER' = \frac{\text{Number of pixels which really embed some secret data}}{[\text{Total Number of pixels}] + (\text{Number of pixels which are shifted without hiding any secret})}. \quad (6)$$

The value of ER' is in the range 0 to 1; whereas the range of ER is 0 to ∞. This new finite-range definition might let some readers have more intuitive feeling about judging how good an image (or a block) is for embedding data. Table 10 is an example comparing the ER and ER' for different images. We can see that smaller ER almost always implies smaller ER', and vice versa. Finally, since there are so many possible alternatives, it makes the hackers harder to steal the hidden secret.

Table 10. The ER and ER' values after applying Phase 1 alone to the host images (when T = 1, Fig. 1 (a) vs. Fig. 14).

		Lena	Baboon	Airplane	Elaine	Lake	Boat	Barbara
The whole image (*i.e.* before doing any deletion)	ER	0.463	0.093	0.623	0.164	0.182	0.191	0.237
	ER'	**0.170**	**0.0437**	**0.206**	**0.0718**	**0.0799**	**0.0836**	**0.0998**
After deleting two 128 × 128 blocks of smallest ER (ER')	ER	0.503	0.103	0.701	0.175	0.192	0.200	0.259
	ER'	**0.180**	**0.0480**	**0.222**	**0.0759**	**0.0836**	**0.0868**	**0.107**
After deleting Four 128 × 128 blocks of smallest ER (ER')	ER	0.530	0.115	0.801	0.181	0.204	0.209	0.283
	ER'	**0.188**	**0.0523**	**0.240**	**0.0795**	**0.0885**	**0.0898**	**0.116**

7. CONCLUSION AND REMARKS

By identifying which parts of a *PE* histogram are very wasteful in hiding, we propose the idea of increasing the so-called efficiency ratio (ER). We propose four phases as four tests to increase ER values. Phase 1 is run to avoid using low ER blocks. Phases 2-4 are run to avoid using unsuitable pixels. Details of the implementation are provided. Experiments are done using several standard images. Tables 6-8 and Fig. 11 show that our PSNR is superior to that of many other PEE-based methods such as [8, 12-13, 24-27]. When similar amounts of data are hidden using each method, our resulting stego image quality is often better than that of other methods.

In general, just like many other prediction-based methods, our performance for smooth images such as Lena surpasses that for texture images such as Baboon. This can be seen from Tables 6 and 7, in which Baboon is also the most unsuitable image for all methods. To explain this, note that the hair of Baboon has extremely noisy texture rather than smooth area; hence it is hard to predict pixel values using neighboring pixels.

As for the number of blocks used in our methods, although using finer blocks seems result in better PSNR, note that the overhead also increases because hiding a binary string is required to indicate the used and bypassed blocks. This string becomes longer when we increase the number of blocks, and this overhead will eventually reduce the hiding capacity. Therefore, we usually partition each 512 × 512 image into 256 blocks. Its PSNR perf-ormance is slightly better than that of using 64 or 16 blocks, but the extra hiding space provided does not increase overhead to a point that reduces the hiding capacity.

Some users might wonder why Phase 1 is needed. If we do not have Phase 1, then in an experiment using 512 × 512 image Lake as host, after doing Phases 2-4, the sixteen 128 × 128 blocks can embed, respectively, (“32”, “1,695”, “1,695”, “86”); (“37”, “1,924”, “2,394”, “142”); (“96”, “57”, “79”, “53”); and (“275”, “157”, “315”, “378”) bits. Here,

("32", "1,695", "1,695", "86") are for the top four 128 × 128 blocks in Row 1. Likewise, ("275", "157", "315", "378") are for the bottom four blocks in Row 4. We can see that the best four blocks together can embed 1,695 + 1,695 + 1,924 + 2,394 = 7,708 bits. These four blocks are the four blocks in Fig. 5 (a) with $ER \geq 0.28$. Likewise, the best eight blocks together embed 1,695 + 1,695 + 1,924 + 2,394 + 378 + 315 + 275 + 157 = 8,833 bits. By contrast, the worst eight blocks together embed only 32 + 86 + 37 + 142 + 96 + 57 + 79 + 53 = 582 bits. The hiding amount in these eight worst blocks together is less than that of a single good block, and the best single block can embed 2,394 bits. Also note that these blocks with poor hiding capacity, if they are used, will have many pixels being value-shifted, thus causing greater distortion. Therefore, it is better to skip these poor-capacity and yet larger-distortion blocks. Given that Phase 1 is faster than each of the remaining three phases, we suggest using Phase 1 to remove the less suitable blocks before running Phases 2-4, as Phases 2-4 are more time-consuming. In summary, using Phase 1 can save time and improve PSNR without significantly reducing hiding capacity. In the above case, even if we delete 8/16 = 50% of the sixteen blocks of Lake, the reduction of data size is only 582/(582 + 8,833) = 582/9,415 = 6%.

Section 5 discusses the protection of large-sized secrets. We use sharing so that large-sized secrets can still be embedded; besides the benefits of missing tolerance and the protection against betrayers. Typical applications include the medical and legal information for hospitals and law offices. Fig. 13 gives an example. Section 6 also gives some alternatives for the design.

ACKNOWLEDGEMENT

Before his retirement, the second author would like to thank National Science Council (NSC) and Ministry of Science and Technology (MOST), Taiwan, for their help in his academic career. The two authors also thank the reviewers, the associate editor, and the editor-in-chief.

REFERENCES

1. R. J. Anderson and F. A. P. Petitcolas, "On the limits of steganography," *IEEE Journal on Selected Areas in Communications*, Vol. 16, 1998, pp. 474-481.
2. F. A. P. Petitcolas, R. J. Anderson, and M. G. Kuhn, "Information hiding-a survey," *Proceedings of the IEEE*, Vol. 87, 1999, pp. 1062-1078.
3. K. Gopalan, "Audio steganography using bit modification," in *Proceedings of International Conference on Multimedia and Expo.*, Vol. 1, 2003, pp. I-629-632.
4. H. L. Yeh, S. T. Gue, P. Tsai, and W. K. Shih, "Reversible video data hiding using neighbouring similarity," *IET Signal Process*, Vol. 8, 2014, pp. 579-587.
5. J. C. Lin, "Synchronizing two frequently-cited high-capacity image hiding methods (Modulus-based vs. LSB-based)," *Journal of Computer Engineering & Information Technology*, Vol. 5, 2016.
6. Z. Ni, Y. Q. Shi, N. Ansari, and W. Su, "Reversible data hiding," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 16, 2006, pp. 354-362.
7. J. Marin and F. Y. Shih, "Reversible data hiding techniques using multiple scanning

difference value histogram modification," *Journal of Information Hiding and Multimedia Signal Processing*, Vol. 5, 2014, pp. 451-460.
8. B. Ou, X. Li, Y. Zhao, R. Ni, and Y.-Q. Shi, "Pairwise prediction error expansion for efficient reversible data hiding," *IEEE Transactions on Image Processing*, Vol. 22, 2013, pp. 5010-5021.
9. C. W. Honsinger, P. Jones, M. Rabbani, and J. C. Stoffel, "Lossless recovery of an original image containing embedded data," U.S. Patent No. 6278791, 2001.
10. D. M. Thodi and J. J. Rodriguez, "Expansion embedding techniques for reversible watermarking," *IEEE Transactions on Image Processing*, Vol. 16, 2007, pp. 721-730.
11. J. Tian, "Reversible data embedding using a difference expansion," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 13, 2003, pp. 890-896.
12. V. Sachnev, H. J. Kim, J. Nam, S. Suresh, and Y. Q. Shi, "Reversible watermarking algorithm using sorting and prediction," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 19, 2009, pp. 989-999.
13. C. Dragoi and D. Coltuc, "Improved rhombus sorting and prediction for reversible watermarking by difference expansion," in *Proceedings of the 20th European Signal Processing Conference*, 2012, pp. 1688-1692.
14. R. Liu, R. Ni, and Y. Zhao, "A reversible data hiding based on adaptive prediction technique and histogram shifting," in *Proceedings of IEEE Asia-Pacific Signal and Information Processing Association Annual Summit and Conference*, 2014, pp. 1-6.
15. I. C. Dragoi and D. Coltuc, "Reversible watermarking based on complementary predictors and context embedding," in *Proceedings of IEEE 24th European Signal Processing Conference*, 2016, pp. 1178-1182.
16. C. Wang, X. Li, and B. Yang, "Efficient reversible image watermarking by using dynamical prediction-error expansion," in *Proceedings of the 17th IEEE International Conference on Image Processing*, 2010, pp. 3673-3676.
17. Y. Q. Shi, X. Li, X. Zhang, H. T. Wu, and B. Ma, "Reversible data hiding: advances in the past two decades," *IEEE Access*, Vol. 4, 2016, pp. 3210-3237.
18. W. Gao, X. Zhang, L. Yang, and H. Liu, "An improved Sobel edge detection," in *Proceedings of the 3rd IEEE International Conference on Computer Science and Information Technology*, 2010, pp. 67-71.
19. J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 8, 1986, pp. 679-698.
20. 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 Transactions on Internet and Information Systems*, Vol. 4, 2010, pp. 655-670.
21. X. Li, W. Zhang, X. Gui, and B. Yang, "A novel reversible data hiding scheme based on two-dimensional difference-histogram modification," *IEEE Transactions on Information Forensics and Security*, Vol. 8, 2013, pp. 1091-1100.
22. G. Coatrieux, W. Pan, N. Cuppens-Boulahia, F. Cuppens, and C. Roux, "Reversible watermarking based on invariant image classification and dynamic histogram shifting," *IEEE Transactions on Information Forensics and Security*, Vol. 8, 2013, pp. 111-120.
23. L. Luo, Z. Chen, M. Chen, X. Zeng, and Z. Xiong, "Reversible image watermarking using interpolation technique," *IEEE Transactions on Information Forensics and Security*, Vol. 5, 2010, pp. 187-193.
24. X. Li, B. Yang, and T. Zeng, "Efficient reversible watermarking based on adaptive

prediction-error expansion and pixel selection," *IEEE Transactions on Image Processing*, Vol. 20, 2011, pp. 3524-3533.
25. E. Gayathri and K. A. Palaniswamy, "Modified difference-histogram based reversible data hiding scheme," *International Journal of Innovative Research in Computer and Communication Engineering*, Vol. 2, 2014, pp. 2287-2293.
26. W. Hong, "Adaptive reversible data hiding method based on error energy control and histogram shifting," *Optics Communications*, Vol. 285, 2012, pp. 101-108.
27. W. Hong, "An efficient prediction-and-shifting embedding technique for high quality reversible data hiding," *EURASIP Journal on Advances in Signal Processing*, Vol. 2010, 2010, Article ID 104835.
28. C. C. Thien and J. C. Lin, "Secret image sharing," *Computers and Graphics*, Vol. 26, 2002, pp. 765-770.

Che-Yi Chao (趙哲誼) has an M.S. degree in the field of Electronic Engineering. He is a Ph.D. candidate in the Department of Computer Science, National Chiao Tung University, Taiwan. His research interests include data hiding and image compression.

Ja-Chen Lin (林志青) received his B.S. and M.S. degrees from National Chiao Tung University, Taiwan. He received his Ph.D. degree in Mathematics from Purdue University, U.S.A. In 1984-1988, he was a graduate instructor at Purdue University. He joined the Department of Computer Science at National Chiao Tung University in 1988 then he became a Professor there. His recent research interests include pattern recognition and image processing. Dr. Lin is a member of the Phi-Tau-Phi Scholastic Honor Society.