Generalized PVO-K Embedding Technique for Reversible Data Hiding

Jian-Jun Li[1], Yun-He Wu[1], Chin-Feng Lee[2] and Chin-Chen Chang[3]
(Corresponding author: Chin-Chen Chang)

Department of Computer Science and Technology & Hangzhou Dianzi University[1]
1158, No.2 Rd., Jianggan, Hangzhou 10336, China
Department of Information Management & Chaoyang University of Technology[2]
168, Jifeng E. Rd., Wufeng, Taichung 41349, Taiwan
Department of Information Engineering and Computer Science & Feng Chia University[3]
100, Wenhwa Rd., Seatwen, Taichung 40724, Taiwan
(Email: alan3c@gmail.com)
(Received Sept. 29, 2016; revised and accepted Dec. 11, 2016)

Abstract

Recently, several reversible information hiding methods based PVO (pixel value ordering) techniques have been proposed, in these methods, secret data always are embedded in pixels with largest or smallest value in the block. In order to make use of the block with multiple largest-valued (or smallest-valued) pixels, a PVO-K method was proposed, which treats K largest-valued (or smallest-valued) pixels as a unit to embed secret data, and all K pixels are modified together to embed one bit of information. In this paper, we propose a generalized PVO-K method (GePVO-K) that takes full advantage of these pixels with largest or smallest values by embedding K bits of secret data into the K pixels. As a result, the GePVO-K method has greater embedding capacity than the PVO-K method. The superiority of the GePVO-K scheme was verified by the experimental results.

Keywords: Pixel Value Ordering; Prediction Error Expansion; Reversible Data Hiding

1 Introduction

Image data hiding is the technology in which secret data, such as authentication or, copyright information, are embedded in a digital image [7, 25]. In data hiding techniques, it is critically important that the recipient be able to extract the secret data completely from the camouflage image, but, at the same time, any decreases in the quality of the image should not particularly evident, and it is especially important that, the difference cannot be detectable by the human eye.

Information hiding processes can be divided into two categories based on the technology they use. One technology is data hiding with distortion, and the other technology is non-distortion data hiding, which also is called RDH (reversible data hiding). Least Significant Bit (LSB) [4, 23], Revisited Matching [13], and Exploiting Modification Direction (EMD) [26] are well-known, non-reversible data hiding techniques that are simple and have high embedding capacity. Compared with an ordinary data hiding algorithm, RDH must take more requirements into consideration. It also requires that the original cover image be recoverable after the secret data have been extracted from the camouflaged image. That is to say, RDH is a special data hiding method that is always applied for scenarios that are sensitive to image distortion, such as processing military, medical, or remote sensing images.

To date, many reversible data hiding techniques have been proposed. The first kind of RDH method was based on lossless compression [2, 3, 5], and this method acquires embedding space through lossless compression of a specific part of the digital cover image. As a result, they usually have low embedding capacity and produce significant distortion of the image. In 2006, Tian et al. proposed an important spatial data hiding algorithm called "difference expansion" (DE) [20]. They overcame the limitations of embedding secret data through lossless compression, and they focused on diffusion of the difference between pixel pairs to embed secret data reversibly. Later, several improved methods involving DE were proposed. One method tried to decrease the size of the location map [8, 12, 24], the second method was based on integer transform [10, 16, 22], and the third method involved prediction error expansion (PEE) [1, 6, 19]. Image data have spatial redundancy that is caused by the correlation between adjacent pixels in the image. The PEE method has great embedding capacity because it can take advantage of the spatial redundancy of a digital image. The PEE

method was first proposed by Thodi et al. [19], after which Hu et al. [16] made some improvements by constructing a location map that depends on the payload, thereby decreasing the size of the compressed location map.

In addition to the DE technique, Ni et al. proposed another important RDH method, i.e., the histogram shift (HS)-based technique [14]. Later, Lee et al. [9] improved the HS method by using a histogram of the difference between adjacent pixels, and this method improved the embedding capacity and reduced image distortion.

Recently, Li et al. [11] proposed a new RDH method based on pixel value ordering (PVO). This method combines DE and HS with PEE and uses the PVO technique to embed secret data in a block-by-block manner. For each block, the pixel values are sorted in ascending order, then, the largest-valued pixel is predicted by the second-largest pixel, and the smallest-valued pixel is predicted by the second-smallest pixel. Thus, the second-largest and the second-smallest pixel values in the block remain unchanged in the embedding phase, and the largest pixel value may become larger since it always is increased, and the smallest pixel value becomes smaller after being decreased. Therefore, the order of pixel values in the block remains unchanged. Typically, the minimum prediction error is non-positive, and the maximum prediction error is non-negative, so the RDH algorithm based on PVO regards the non-negative and non-positive values, which occur most frequently, as the carriers of secret data. The prediction error in the range of "-1" to "1" is defined as the peak value, and the secret data are embedded in the peaks using HS technology. Then, the smallest and largest pixel values are modified according to the prediction error values.

However, since the local pixel values are correlative, there will be many prediction errors 0, which are ignored by the PVO method. In view of this phenomenon, Peng et al. [17] proposed an improved PVO method, i.e., IPVO, and they used the pixel location number in the blocks before ordering the pixel values to optimize the process of generating the prediction error. In addition to IPVO, another method, known as PVO-K, was proposed in [15], and its aim also was to improve the PVO. The PVO-K method treats K identical largest-valued or smallest-valued pixels as a unit to embed secret data. Compared with PVO, when the largest pixel value and the second-largest value (or the smallest pixel value and second-smallest pixel value) are equal, they are treated as a unit, so this block may be still used to embed secret data. Obviously, the PVO method is a special case of the PVO-K method, i.e., when K=1. In [15], the embedding capacity was improved by using a combination of the PVO-1 and PVO-2 methods. But in the smooth block, there often are more identical largest or smallest pixel values, and the PVO-K method may change all K pixel values to embed just one bit of secret data, so there is still room for improving the PVO-K method. Besides, a new path of methods has been proposed in 2015, they break the block restrictions of other PVO-based methods, therefore make more use of the pixels that can be utilized to embed secrete data, and significantly enhance the performance of the PVO-based method, like the PPVO [18] and the method of Wang et al. [21]. In this paper, a strategy is presented concerning ways to improve the PVO-K method so that K bits of secret data can be embedded into K largest-valued or smallest-valued pixels. We also proposed a new way to produce a special block and compared the performance with traditional treatments.

The remaining sections of this paper are organized as follows. Several PVO-based methods are introduced in Section 2. In Section 3, a new generalized PVO-K scheme is proposed. Section 4 presents relevant experiments and the analysis of the results. Our conclusions are presented in Section 5.

2 Related Works

In this section, two PVO-based reversible data hiding methods are introduced briefly, i.e., PVO [11], PVO-K [15].

2.1 RDH Method Based On PVO

The PVO method proposed by Li et al. provided a new predictor for the prediction error expansion, with both largest and smallest pixel values being used in a block for embedding data. The embedding process is firstly divide the cover image into blocks of pixels, and number the pixels in each block, i.e.,$(x_1, x_2, ..., x_{n1\times n2})$. Then, sort the pixels in ascending order to get an ordered sequence $(x_{\pi(1)}, x_{\pi(2)}, ..., x_{\pi(n1\times n2)})$.

After that, count two prediction errors according Equation (1), wherein, the non-negative integer $d_{\max}$ represents the difference between the largest pixel value and the second-largest pixel value; and a non-positive integer $d_{\min}$ represents the difference between the smallest pixel value and the second-smallest pixel value.The secret message $b \in \{0, 1\}$can be embedded when the maximum prediction error is 1 or the minimal prediction error is -1. Prediction errors are modified according to Equation (2)and Equation (3).At last, revise the largest and smallest pixel values using Equation (4) and proceed to the next block until all blocks have been processed or all secret data have been embedded.

$$\begin{cases} d_{\max} = x_{\pi(n1\times n2)} - x_{\pi(n1\times n2-1)} \\ d_{\min} = x_{\pi 1} - x_{\pi 2} \end{cases} \tag{1}$$

$$d'_{\max} = \begin{cases} d_{\max}\ if d_{\max}{=}0 \\ d_{\max}{+}b\ if d_{\max}{=}\ 1 \\ d_{\max}{+}1\ if d_{\max} > 1 \end{cases}, \tag{2}$$

$$d'_{\min} = \begin{cases} d_{\min}\ if d_{\min}{=}0 \\ d_{\min} - b\ if d_{\min}{=}-1 \\ d_{\min} - 1\ if d_{\min} < -1 \end{cases}. \tag{3}$$

$$\begin{cases} {x'}_{\pi(n1\times n2)} = x_{\pi(n1\times n2-1)} + {d'}_{\max} \\ {x'}_{\pi(1)} = x_{\pi(2)} + {d'}_{\min} \end{cases}. \tag{4}$$

Because the order of the pixel values remains unchanged after embedding the secret data, the secret data can be extracted in the extraction phase from the largest-valued and smallest-valued pixels according to reverse process of the embedding procedure. At the same time, the pixel values can be changed back to the original values.

2.2 RDH Method Based On PVO-K

Like IPVO, PVO-K also was proposed for the purpose of using the prediction error "0", which is discarded in the PVO method, but the difference is that PVO-K treats the largest or smallest pixel values as a unit for embedding secret data. Similar to these methods, we take the procedure of embedding secret data into a maximum number of pixels as an example; assume that the sorted pixel values in a block are:$x_{\pi(1)} \leq ... \leq x_{\pi(n1\times n2-K)} < x_{\pi(n1\times n2-K+1)} = ... = x_{\pi(n1\times n2)}$,where K is the number of largest-valued pixels, and the prediction error is calculated using Equation (5).

$$d_{\max} = x_{\pi(n1\times n2-K+1)} - x_{\pi(n1\times n2-K)}. \quad (5)$$

When the prediction error is "1", one bit of secret data can be embedded; otherwise, the K pixel values are shifted. The prediction errors are modified by Equation (6).

$$d'_{\max} = \begin{cases} d_{\max}+b \ if d_{\max}= 1 \\ d_{\max}+1 \ if d_{\max} > 1 \end{cases}. \quad (6)$$

Then the largest pixel values are modified by Equation (7), where, $i \in \{n1 \times n2 - K + 1, n1 \times n2 - K + 2, ..., n1 \times n2\}$.

$$x'_{\pi(i)} = x_{\pi(i)} + d'_{\max}. \quad (7)$$

The common factor of PVO-K and other PVO-based methods is that the original block sorting remains constant after embedding the secret data, which makes the extraction process more convenient.

3 Proposed Scheme

In this section, we propose a generalized scheme for the PVO-K method with respect to embedding capacity, and it is called GePVO-K. First, we introduce how to embed one bit of secret data in each largest-valued pixel by modifying the largest and the second-largest pixel values. Some examples are provided to demonstrate our approach. Then, the process of embedding secret data in each smallest-valued pixel is presented. Finally, we show the detailed steps of the embedding and extraction procedures.

3.1 Embedding Secret Data in Largest-valued Pixels and Data Extraction Procedure

As mentioned in the previous sections, if the PVO-K algorithm embeds one bit of secret data in a block that has K largest-valued or smallest-valued pixels, all of the K pixels must be modified in the same way. Ou et al. [15] indicated that when PVO-1 and PVO-2 are used together to increase the embedding capacity of traditional PVO-based methods; however if $K > 2$, the block should not be used to embed secret data, because a larger K will lead to a greater distortion caused by more changes in the pixels values. In nature images, especially in the blocks of the smooth region, K is often greater than 2, so the smooth region always is ignored, which makes less embedding capacity. For this phenomenon, we propose an improved method that still utilizes the largest-valued and smallest-valued pixels in the block to embed secret data, but one bit of secret data can be embedded in each pixel. Here, we present the details of embedding secret data in the largest-valued pixels as well as the extracting procedure.

3.1.1 Embedding Secret Data in Largest-valued Pixels

First, the cover image should be divided into blocks. Let the size of block B be $n_1 \times n_2$. Then, each block is visited in a zigzag manner to establish a location map, and the rules for establishing the map are as follows. If the block has the pixel values that may overflow/underflow, such as "0," "1," "254," "255," the block's position is recorded as "2;" if all of the pixel values in the block are the same, the block's position is recorded as "1;" the remaining blocks are normal blocks, and their positions are recorded as "0s."

Next, we deal with each block depending on the following cases:

Case 1: If the position number of the block in the location map is $LM(B) = 2$, the block is not used to embed secret data, and it is skipped.

Case 2: If the position number of the block in the location map is $LM(B) = 1$, i.e., all pixel values are equal in the block B, we keep the first pixel value unchanged and then embed the secret data in the remaining pixels and the pixel values are modified by Equation (8) in a zigzag manner. It states that if a to-be-embedded bit $b = 0$, do not change the pixel value; if $b = 1$, increase the pixel value by one.

$$x'_{\pi(i)} = \begin{cases} x_{\pi(i)} \ if i= 1 \\ x_{\pi(i)} + b_{i-1} \ if i= 2,3,...,\mathrm{n1} \times \mathrm{n2} \end{cases}. \quad (8)$$

Case 3: If the position number of the block in location map $LM(B) = 0$, we number the pixels in a zigzag scanning order to get $B\,(x_1, x_2, ..., x_{n1\times n2})$, and then we sort the pixel values in ascending order to obtain

a sorted block $B_\pi\left(x_{\pi(1)}, x_{\pi(2)}, ..., x_{\pi(n1\times n2)}\right)$. Assume that the ordering result is:$x_{\pi(n1\times n2-K-L)} < x_{\pi(n1\times n2-K-L+1)} = ... = x_{\pi(n1\times n2-K)} < x_{\pi(n1\times n2-K+1)} = ... = x_{\pi(n1\times n2)}$. That is, there are K largest pixels $x_{\pi(n1\times n2-K+1)} = x_{\pi(n1\times n2-K+2)} = ... = x_{\pi(n1\times n2)}$, and L second-largest pixels $x_{\pi(n1\times n2-K-L+1)} = x_{\pi(n1\times n2-K-L+2)} = ... = x_{\pi(n1\times n2-K)}$. Then, we calculate the maximum prediction error using Equation (9).

$$d_{\max} = x_{\pi(n1\times n2-K+1)} - x_{\pi(n1\times n2-K)}. \quad (9)$$

Case 3-1: If $d_{\max} > 1$, this block is not fit to be used to embed secret data, and all largest pixel values should be increased by one as Equation (10).Where,$i \in \{n1\times n2-K+1, n1\times n2-K+2, ..., n1\times n2\}$.

$$x'_{\pi(i)} = x_{\pi(i)} + 1. \quad (10)$$

Case 3-2: if $d_{\max} = 1$, K bits secret data can be embedded into the largest-valued pixels. In order to correctly find which pixels were embedded secret data during extracting process, the difference between the second-largest pixels and the third-largest pixels need to be expanded also. Specifically, after embedding secret data, the original largest-valued pixels may be still kept in the position of largest-valued pixels or some of them may change to the second-largest pixels. To distinguish these two situations for the purpose of extracting secret data and recovering the original pixels, the difference between the second-largest pixels and the third-largest pixels can be used as a judgment condition and its detailed usage will be presented in the extracting procedure. So in this embedding case, first we increase the K largest pixel values and the L second-largest pixel values by one; then we embed the secret data, $b_r \in \{0,1\}(r = 1, 2, ..., K)$, into the largest-valued pixels in the numbering order. If $b_r = 0$, keep the largest pixel value unchanged; if $b_r = 1$, increase the largest pixel value by one. In summary, the pixel values are modified by Equation (11). Where, $b_i \in \{0,1\}$, $i \in \{n1\times n2-K+1, n1\times n2-K+2, ..., n1\times n2\}$ and $j \in \{n1\times n2-K-L+1, n1\times n2-K-L+2, \cdots, n1\times n2-K\}$.

$$\begin{cases} x'_{\pi(i)} = x_{\pi(i)} + b_{i-n1\times n2+K} + 1 \\ x'_{\pi(j)} = x_{\pi(j)} + 1 \end{cases}, \quad (11)$$

3.1.2 Extracting Secret Data From Larger-valued Pixels and Restoring the Pixel Values

As can be seen from Equation (11), for a normal block (i.e., its recorded number in the location map is 0), the original ordering may be changed after the secret data have been embedded. Because some largest-valued pixel of the original block may become the second-largest after embedding the secret data, so, in the camouflage block, information can be hidden only in the largest-valued or the second-largest pixels. We can determine whether the secret data are completely hidden in the largest-valued pixels or in both the largest-valued and the second-largest pixels. Therefore, we can completely extract the secret data revise the pixel values according as follows.

First, we divide the camouflage image into blocks as we did in the embedding procedure, then, we handle each block depending on the following cases:

Case 1: If the position number of the camouflage block in location map $LM(B) = 2$, there are no hidden secret data, and the original block is the same as the camouflage block.

Case 2: If the position number of the camouflage block in location map $LM(B) = 1$, we extract the secret data starting from the second pixel. First, we calculate the prediction error d_i according to Equation (12). If $d_i = 0$, extract secret data $b_{i-1} = 0$, keeping the pixel value unchanged; if $d_i = 1$, extract the secret data $b_{i-1} = 1$, decreasing the pixel value by one, as shown in Equation (13). Where, $i \in \{2, 3, ..., n1\times n2\}$.

$$d_i = x'_{\pi(i)} - x'_{\pi(1)}, \quad (12)$$

$$\begin{aligned} x_{\pi(1)} &= x'_{\pi(1)}, \\ x_{\pi(i)} &= \begin{cases} x'_{\pi(i)}, b_{i-1}{=}\,0 \ if d_i{=}0 \\ x'_{\pi(i)} - 1, b_{i-1}{=}\,1 \ if d_i{=}\,1 \end{cases}. \end{aligned} \quad (13)$$

Case 3: If the position number of the camouflage block in location map $LM(B) = 0$, we sort the pixels in ascending order to obtain $B'_\pi\left(x'_{\pi(1)}, x'_{\pi(2)}, ..., x'_{\pi(n1\times n2)}\right)$, assuming that the ordering results are $x'_{\pi(n1\times n2-R-S-T)} < x'_{\pi(n1\times n2-R-S-T+1)} = ... = x'_{\pi(n1\times n2-R-S)} < x'_{\pi(n1\times n2-R-S+1)} = ... = x'_{\pi(n1\times n2-R)} < x'_{\pi(n1\times n2-R+1)} = ... = x'_{\pi(n1\times n2)}$. Which means there are R largest-valued pixels (m1), S second-largest pixels (m2), and T third-largest pixels (m3).

m1: $x'_{\pi(n1\times n2-R+1)} = x'_{\pi(n1\times n2-R+2)} = ... = x'_{\pi(n1\times n2)}$;

m2: $x'_{\pi(n1\times n2-R-S+1)} = x'_{\pi(n1\times n2-R-S+2)} = ... = x'_{\pi(n1\times n2-R)}$;

m3: $x'_{\pi(n1\times n2-R-S-T+1)} = x'_{\pi(n1\times n2-R-S-T+2)} = ... = x'_{\pi(n1\times n2-R-S)}$.

Two prediction errors are calculated according to Equation (14). If T does not equal to 0, it's very plain that the prediction errors $d1 \geq 1$ and $d2 \geq 1$; if T equals to 0which means that there are not the third-largest pixels, only one prediction error $d1 \geq 1$

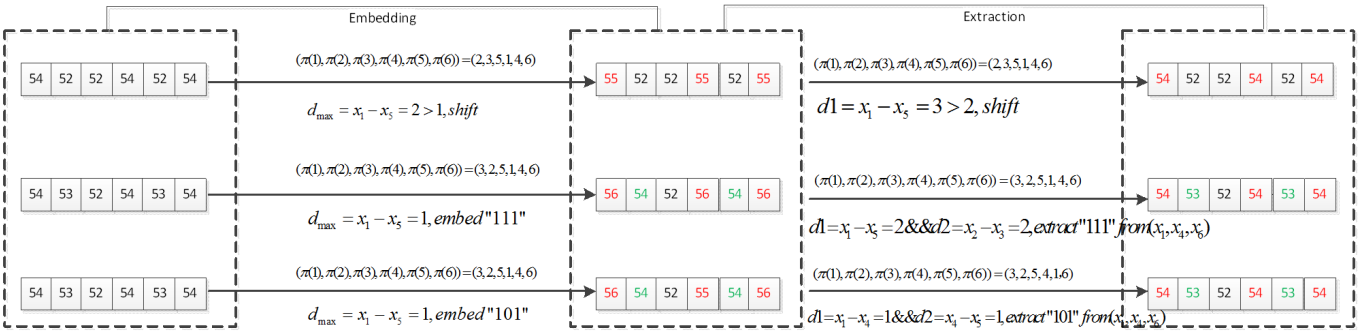


Figure 1: Example of embedding secret data in largest-valued pixels and extracting procedure

is obtained. So we process the block on the basis of the following Cases 3-1–3-3.

$$\begin{cases} d1 = x'_{\pi(n1\times n2-R+1)} - x'_{\pi(n1\times n2-R)} \\ d2 = x'_{\pi(n1\times n2-R-S+1)} - x'_{\pi(n1\times n2-R-S)} \end{cases} \quad (14)$$

Case 3-1: If $d1 > 2$, there are no hidden secret data, and we decrease R largest pixel values by one, the pixel values are revised according to Equation (15).

$$x_{\pi(i)} = x'_{\pi(i)} - 1, \quad i \in \{n1\times n2 - R + 1, n1\times n2 - R + 2, ..., n1\times n2\}. \quad (15)$$

Case 3-2: If $d1 \leq 2$ and $d2 = 1$, the secret data were embedded in the m1 and m2 zones. First, we decrease the value of pixels in the m1, m2, and m3 zones by one. (Notice that d1 and d2 remain unchanged in keeping with Equation (16).

$$x_{\pi(i)} = x'_{\pi(i)} - 1, \quad i \in \{n1\times n2 - R - S - T + 1, n1\times n2 - R - S - T + 2, ..., n1\times n2\}. \quad (16)$$

Then, we extract the secret data $S(B) = \{b_i | b_i \in \{0,1\}, i = 1, 2, ..., R+S\}$ from the pixels in the m1 and m2 zones, depending on the numbering order. We count D_i one by one, where D_i is the difference in the values between the pixels in the m1 or m2 zone and the pixels in the m3 zone, as indicated in Equation (17). If $D_i = 2$, decrease the pixel value by one and extract secret data $b_i = 1$; if $D_i = 1$, let the pixel value remain unchanged and extract secret data $b_i = 0$. The extraction procedure depends on Equation (18). Where, $i \in \{n1\times n2 - R - S + 1, n1\times n2 - R - S + 2, ..., n1\times n2\}$.

$$D_i = x'_{\pi(i)} - x'_{\pi(n1\times n2-R-S)}, \quad i \in \{n1\times n2 - R - S + 1, n1\times n2 - R - S + 2, ..., n1\times n2\}. \quad (17)$$

$$x_{\pi(i)} = \begin{cases} x'_{\pi(i)}, b_i = 0 \ if D_i = 1 \\ x'_{\pi(i)} - 1, b_i = 1 \ if D_i = 2 \end{cases} \quad (18)$$

Case 3-3: If $d1 \leq 2$ and $(d2 \geq 2 || T = 0)$, this means secret data were embedded in m1 zones, and the secret data fragment $S(B)$is a binary string only including 1 (in this case, $d1 = 2$), or an all 0 binary string (in this case, $d1 = 1$). So, first, we decrease the value of pixels in the m1 and m2 zones by one, and then extract the secret data from m1 in the numbering order. The secret data are determined by the value D_i (Equation (19)), which is the difference between the pixels in m1 and the pixels in m2. If $D_i = 2$, decrease the pixel value by one and extract secret data $b_i = 1$; if $D_i = 1$, keep the pixel value unchanged and extract secret data $b_i = 0$. In short, we can recover the original pixel value and extract secret data as Equation (20). Where, $j \in \{n1\times n2 - R - S + 1, n1\times n2 - R - S + 2, ..., n1\times n2\}$ and $i \in \{n1\times n2 - R + 1, n1\times n2 - R + 2, ..., n1\times n2\}$.

$$D_i = x'_{\pi(i)} - x'_{\pi(n1\times n2-R)}, \quad i \in \{n1\times n2 - R + 1, n1\times n2 - R + 2, ..., n1\times n2\}. \quad (19)$$

$$x_{\pi(j)} = x'_{\pi(j)} - 1, \quad x_{\pi(i)} = \begin{cases} x'_{\pi(i)}, b_i = 0 \ if D_i = 1 \\ x'_{\pi(i)} - 1, b_i = 1 \ if D_i = 2 \end{cases}. \quad (20)$$

3.1.3 Example of Embedding and Extraction Procedures

For a better illustration, there are several examples to demonstrate the above steps in Figure 1. We assume that the block size is $n1 = 2$ and $n2 = 3$. As can be seen from Figure 1, three blocks are selected as examples. All the pixel values in these three blocks are numbered and sorted firstly, for instances, the original pixel sequence in first block is (54, 52, 52, 54, 52, 54), the sorted pixel values are (52, 52, 52, 54, 54, 54) and their numbers in the original block are (2, 3, 5, 1, 4, 6). There are three largest pixels and three second-largest pixels in first block, according embedding rules Case 3 in 3.1.1 part of Section 3.1, we calculate the prediction error $d_{\max} = x_1 - x_5 = 2$, because

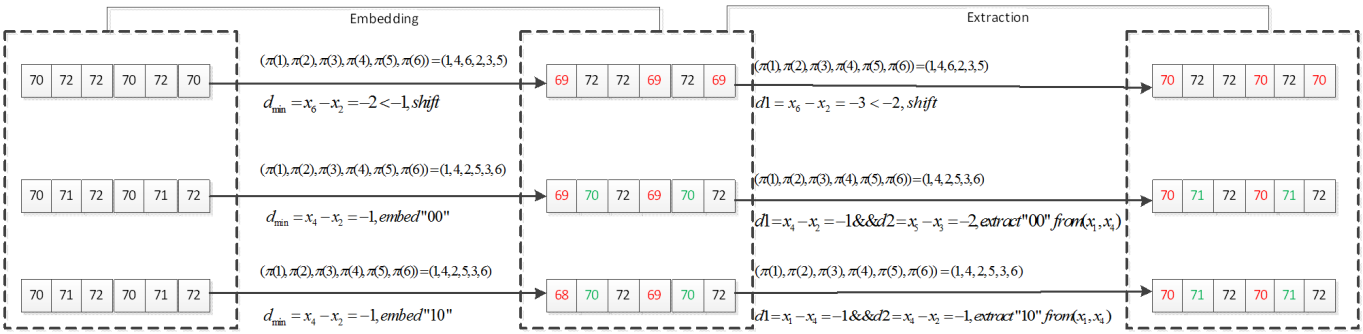


Figure 2: Example of embedding secret data in smallest-valued pixels and extracting procedure

$d_{\max} > 1$, this block is not fit to be used to embed secret data, and all largest pixel values should be increased by one to get the pixel values (52, 52, 52, 55, 55, 55); after that, the pixels need to be moved to their original position according to their number (2, 3, 5, 1, 4, 6). Finally, the pixel values of first block become to (55, 52, 52, 55, 52, 55).

The embedding procedures of the other two blocks are similar. For the second block, the original pixel sequence is (54, 53, 52, 54, 53, 54). After pixel numbering and sorting, the pixels become (52, 53, 53, 54, 54, 54) and their number are (3, 2, 5, 1, 4, 6). There are three largest pixels and two second-largest pixels, we calculate the prediction error $d_{\max} = x_1 - x_5 = 1$. According to the embedding rules in 3.1.1 part of Section 3.1, three bits secret data can be embedded into three largest pixels, the three largest pixels and two second-largest pixels need to be firstly increased by one to get (52, 54, 54, 55, 55, 55), and then three secret bits (we choose111) are embedded into three largest pixels to get (52, 54, 54, 56, 56, 56). Next, move them to their original position according to the number sequence (3, 2, 5, 1, 4, 6), so the final pixel sequence is (56, 54, 52, 56, 54, 56). For the third block, its pixels are the same as the second block. The difference is that we embed the secret data that has both 0 and 1 into three largest pixels for better demonstration of different extraction cases below.

The first step of the extraction procedures is numbering and sorting the pixels in the block too, then the blocks are handled according the prediction errors. As shown in Figure 1, for the first block, the sorted pixel sequence is (52, 52, 52, 55, 55, 55), and the prediction error $d_1 = x_1 - x_5 = 3 > 2$, which means there are no secret data and three largest pixels need to be decreased by one to get (52, 52, 52, 54, 54, 54), then move them to their original position to get the original block.

For the second block, the sorted sequence is (52, 54, 54, 56, 56, 56), prediction error $d_1 = x_1 - x_5 = 2 \leq 2$. In this case, we easily know that there are embedded secret data, but we cannot determine whether it is hidden within all the largest pixels or both the largest pixels and the second-largest pixels. Because if the embedded secret data is an all "0" string or all "1" string, all the largest pixels remains largest after embedding procedure, such as the second block; if the embedded secret data is a string with both "0" and "1", some of the largest pixels becomes to the second largest pixels after embedding procedure, such as the third block. Thus, we need to calculate a prediction error between the second-largest pixels and the third-largest pixels to distinguish these two cases. As presented in Figure 1, another prediction error of the second block $d_2 = x_2 - x_3 = 2 \geq 2$, according Case 3-3 in B part of Section 3.1, all the secret data are embedded in the largest pixels. Therefore, we extract the three bits of secret data 111 and recover the pixels to get (52, 53, 53, 54, 54, 54), then we can obtain the original sequence (54, 53, 52, 54, 53, 54) by moving them to their original positions. For the third block, the sorted sequence is (52, 54, 54, 55, 56, 56), the prediction errors $d_1 = x_1 - x_4 = 1 \leq 2$ and $d_2 = x_4 - x_5 = 1$, which means that the secret data is in both two largest pixels and one second-largest pixels. Therefore we can extract the secret data 101 and recover the original pixels (54, 53, 52, 54, 53, 54), according to Case 3-2 in 3.1.2 part of Section 3.1.

3.2 Embedding Secret Data in Smallest-valued Pixels and Data Extraction Procedure

Except for the special blocks ($LM(B) = 1$ or $LM(B) = 2$), we also used the smallest-valued pixels in the normal block to embed secret data. In order to further demonstrate the proposed methods, Figure 2 presents several examples of embedding data into the smallest-valued pixels and the data extraction procedure. Similar to the example in Figure 1, there are three embedding cases. Case 1: The first block is not fit to embed secret data; Case 2: The second block embeds an all "0" or "1" string; Case 3: The third block embeds a string with both "0" and "1".

The specific steps of embedding secret data in the smallest pixels and extraction procedures are significant similar to Figure 1, in order to avoid duplication, we just take the complex cases (the third block) as an example to illustrate the procedure. For the third block, the original sequence (70, 71, 72, 70, 71, 72) is numbered and

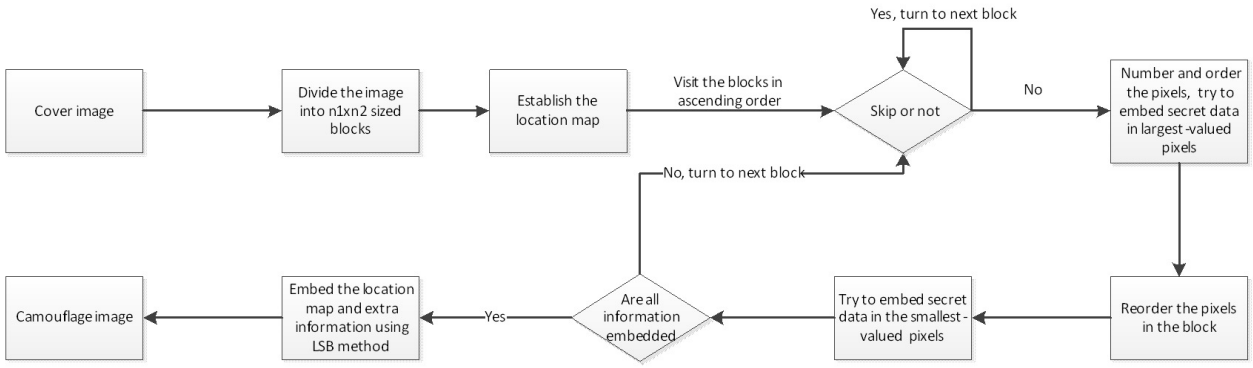


Figure 3: The flowchart of proposed data embedding procedure

sorted to get (70, 70, 71, 71, 72, 72), where their number are (1, 4, 2, 5, 3, 6). There are two smallest pixels and two second-smallest pixels, and the prediction error $d_{\min} = x_4 - x_2 = -1$. Thus two bits of secret data can be embedded in two smallest pixels. The two smallest pixels and two second-smallest pixels need to be decreased by one to get (69, 69, 70, 70, 72, 72), and then we embed "10" in two smallest pixels in two smallest pixels to get (68, 69, 70, 70, 72, 72). The last step of the embedding procedure is to move them to their original positions to get (68, 70, 72, 69, 70, 72). During extraction procedures, the first step is the same as embedding procedure. After numbering and sorting, we can get the sequence (68, 69, 70, 70, 72, 72). Next, we compute the prediction errors $d_1 = x_1 - x_4 = -1$, and $d_2 = x_4 - x_2 = -1$. It is evident that two secret data bits are embedded: one in the smallest pixel and one in the second-smallest pixel. Then, the extract procedure gets "10" from the smallest pixel and the second-smallest pixel and then the recovery process increases the smallest pixel by two, the second-smallest pixel by one, and two third-smallest pixels are also increased by one to get the sequence (70, 70, 71, 71, 72, 72). The last step is also to move the sequence (70, 70, 71, 71, 72, 72) according to their original positions; finally, we can obtain the original sequence (70, 71, 72, 70, 71, 72).

3.3 Proposed Data Embedding Procedure

In this section, we describe the detailed steps of the embedding phase in the proposed scheme. For the normal block, first, we embed the information into the largest-valued pixels, and, then, we reorder the pixels in the block; next, the smallest-valued pixels also are utilized to embed secret data. Like PVO-K, there also is the possibility of an overflow/underflow situation. So, we build a location map to record the position of the pixels that may overflow/underflow. In addition, we record some auxiliary information, as shown in Table 1. We assume that the size of the cover image is $H \times W$ and that the minimal and maximal block sizes are 2×2 and 4×4, respectively.

The detailed steps of the embedding phase are shown as follows, and Figure 3 is the embedding flowchart.

Step 1. Divide the cover image into $n1 \times n2$ -sized blocks and then visit each block in a zigzag manner to establish the location map according to the rules in Section 3.1. After that, two binary bits are used to represent a value in the location map and then the location map is compressed using arithmetic coding, a lossless data compression, to reduce its length.

Step 2. For each block B, if $LM(B) = 2$, skip; if $LM(B) = 1$, embed secret data in the pixels except for the first one; if $LM(B) = 0$, number and order the pixels in the block, try to embed secret data in the largest-valued pixels, and then reorder the pixel values in the block and try to embed the secret data in the smallest-valued pixels.

Step 3. When embedding the secret data is completed, embed first $2 \times \log_2((H \times W)/(n1 \times n2)) + \log_2(H \times W) + L_1 + 4$ least significant bits (LSB) of the pixels in the cover image into the remaining blocks and then record the last embedding position.

Step 4. Use the LSB method [4] to embed extra information and the compressed location map into the cover image from the first pixel.

3.4 Proposed Data Extraction Procedure

The corresponding proposed data extraction procedure is presented in this section; its flowchart is shown in Figure 4.

Step 1. First, extract the extra information and the compressed location map by using LSB method, and then decompress the location map to obtain LM.

Step 2. Next, divide the camouflaged image into blocks depending on the size that was extracted from Step1. Then, visit the blocks in reverse order, which means the extraction procedure must start from the last embedding position. For each block, if $LM(B) = 2$,

Table 1: The extra information

Extra information	Purposes	Memory required
The compressed location map	Record the special blocks	L_1 bits
The length of compressed location map	Correctly extract the location map	$2\times\log_2((H\times W)/(n1\times n2))$ bits
Block size $n1\times n2$	Divide the camouflage image	4 bits
The last position of embedding secret data i and j	Reversely extracting the secret data.	$\log_2(H\times W)$ bits

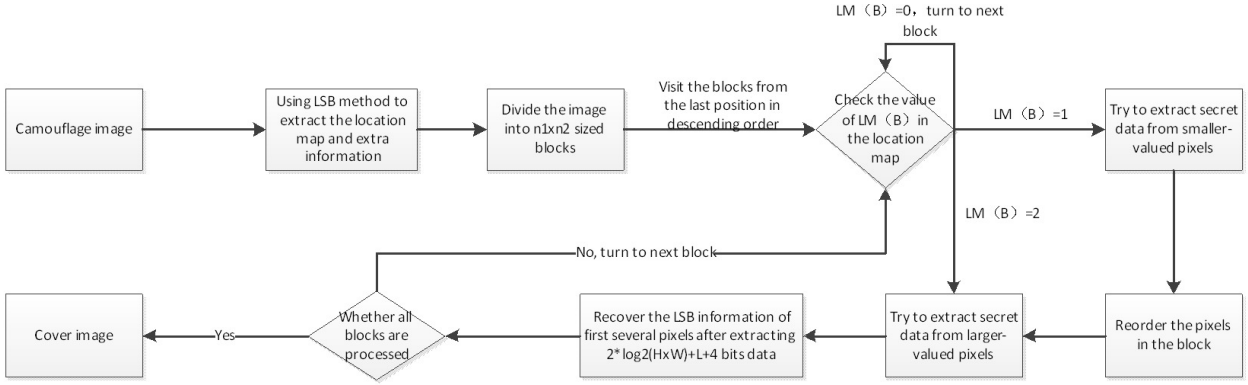


Figure 4: The flowchart of proposed data extraction procedure

skip; if $LM(B)=1$, extract secret data from the pixels in zigzag order except the first one; if $LM(B)=0$, number and order the pixels and try to extract the secret data from the smaller-valued pixels and then reorder the pixels and try to extract data from the larger-valued pixels.

Step 3. After extracting $2\times\log_2((H\times W)/(n1\times n2))+\log_2(H\times W)+L_1+4$ bits of data, revise the LSB information of the first several pixels that were modified to embed the compressed location map and extra information. Then, continue to extract secret data from the remaining blocks until all blocks have been processed.

To better demonstrate the embedding and data extraction procedures, a detailed example is given in Figure 5.

4 Experimental Results

In this section, first, we describe the experimental environment and the evaluation criteria in Section 4.1. Then, the results are discussed in five parts. Section 4.2 is the self-analysis of the proposed scheme in which we discuss the relationship between embedding capacity and the image quality; Section 4.3 compares the performance of GePVO-K with that of PVO-K by using a different block size; in Section 4.4, we analyze two methods of dealing with special blocks; since the proposed scheme is aimed mainly at enhancing the embedding capacity, several PVO-based methods were compared with the proposed scheme in terms of maximum embedding capacity in Section 4.5; Section 4.6 shows the multi-level embedding performance of several PVO-based methods, including GePVO-K.

4.1 Experimental Environment and the Evaluation Criteria

We tested eight cover images using the MATLAB R2010a Platform; all of the cover images were 8-bit grayscale images, and the size of each image was 512×512. Refer to Figure 6. During the experiment, the secret message was a randomly-generated string composed of 0s and 1s.

In the experiment, we used EC (embedding capacity), bpp (bits per pixel), and PSNR (peak signal-to-noise ratio) to evaluate the proposed scheme. EC means the number of embedded bits in the camouflaged image; bpp represents the average embedded bits per pixel; PSNR was used to evaluate the quality of the camouflaged image. PSNR is defined as Equation (21), where H and W are the image height and width, respectively, and MSE (mean square error) is defined as Equation (22).

$$PSNR = 10\times\log_{10}\left(\frac{255^2}{MSE}\right), \tag{21}$$

$$MSE = \frac{1}{H\times W}\sum_{i=1}^{H}\sum_{j=1}^{W}(\text{CoverImage}_{(i,j)}-\text{StegoImage}_{(i,j)})^2. \tag{22}$$

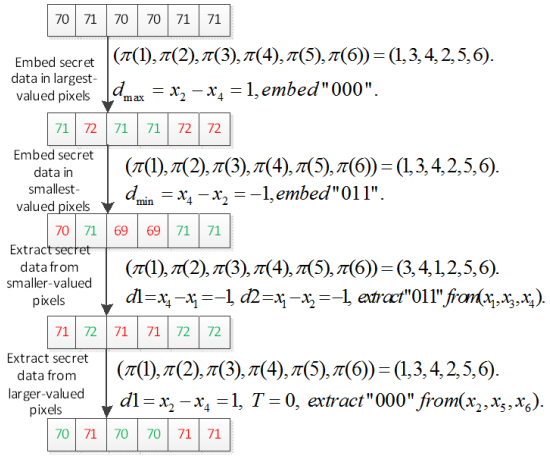


Figure 5: A detailed example of embedding and data extraction procedures

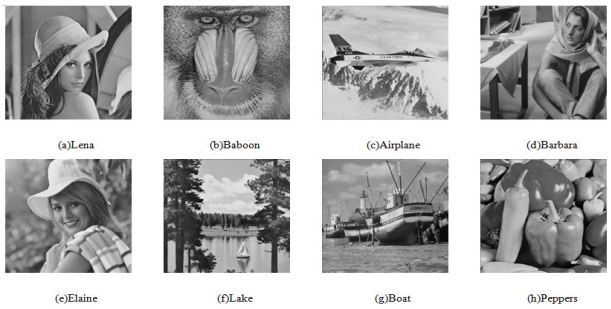


Figure 6: The test cover images

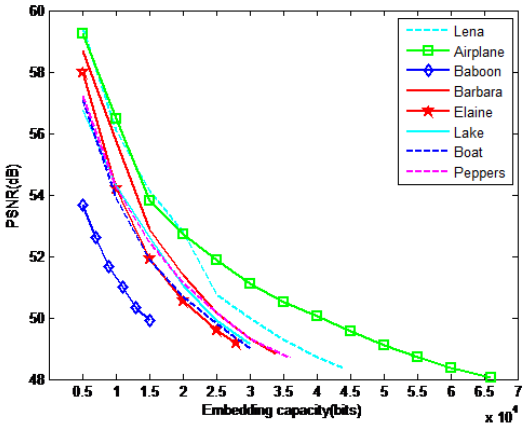


Figure 7: Performance of the proposed scheme

4.2 The Analysis of Proposed Scheme

In this section, we evaluate the proposed scheme by using eight standard grayscale images as the cover image. Figure 7 shows the relationship between embedding capacity and the quality of the camouflaged image. Figure 7 shows that, like all data-hiding algorithms, the quality of the different images gradually declined as the embedding capacity increased. However, comparing the graphs of the different images, it is apparent that the effects caused by increasing the embedding capacity for the different kinds of images are still different. For example, from the test results of images "Baboon" and "Airplane", it is apparent that the degradation of the quality of image "Airplane" is greater than that of image "Baboon" when the embedding capacity was increased from 5000 to 15,000 bits. This occurred because there are more smooth blocks in image "Airplane", and the quality of the image is more sensitive to changes in these blocks. In addition, since we embedded secret data in the same largest and smallest values, the image with more smooth blocks would have the higher embedding capacity. From the experimental results as shown in Figure 7, the maximum embedding capacity of image "Baboon" was the smallest, i.e., just 15,000 bits. Compared to image "Baboon", smoother images, such as images "Lena" and "Elaine" can hide between 20,000 and 30,000 more bits of secret data, and the embedding capacity of smoothest image, i.e., image "Airplane", had 51,000 more bits than that of image "Baboon".

4.3 Comparison of the Performances of PVO-K and GePVO-K

Similar to PVO-K, the size of the block will affect the performance of the proposed scheme. Therefore, we used different block sizes to test the performance of the GePVO-K and PVO-K methods. Taking the local correlation of digital image into account, block sizes that are too large will reduce the maximum embedding capacity drastically. So, five kinds of blocks were tested, the sizes of which were 2×2, 2×3, 3×3, 3×4, and 4×4, respectively. Tables 2-3 show the experimental results of "Lena", "Airplane".

As can be seen from the results in the Tables 2-3, in the process of increasing the block size, the maximum embedding capacity will be reduced because the relevance of the pixel values in the block will be reduced. However, the image quality will be enhanced. In the case of different blocks, GePVO-K greatly improved the maximum embedding capacity. The PVO-K method embeds one bit of secret data to move K bits, so the modification to each pixel is, at most, 1, while the GePVO-K method can embed K bits of data into K largest-valued pixels, and, of course, each pixel value may be expanded by 1 or 2; in addition, the second-largest pixels are modified by one, but when the block is small, the modified pixels may be shifted to the original value in the procedure of embedding secret data in the smallest-valued pixels. As shown in Example 3.4, we embedded six bits of secret data, but

the total modification to all pixels was only 2. Therefore, although GePVO-K inevitably decreases the image quality while improving the embedding capacity, the negative impact is not too high.

Table 2: Performance comparison between PVO-K and proposed GePVO-K (Lena)

Lena	**PVO-K**			**Proposed**			
B size	EC	Payload	PSNR	EC	Payload	PSNR	Gain in EC
2x2	37000	0.14	51.36	44000	0.17	48.37	7000
2x3	28000	0.11	52.27	37300	0.14	49.32	9300
3x3	20000	0.08	54.32	31000	0.12	50.45	11000
3x4	16300	0.06	55.52	25700	0.10	51.41	9400
4x4	12700	0.05	56.87	21200	0.08	52.28	8500

In addition, we can determine from the experiment results, for different types of images, the performance in improving the EC of the proposed scheme was not the same. for the smoother images, i.e., "Lena", the average EC was increased by about 5,800 bits; for the smoothest image, "Airplane," the average EC was increased by about 21,000 bits. These experimental data indicate that the performance of the proposed scheme was better for the smoother images.

Table 3: Performance comparison between PVO-K and proposed GePVO-K (Airplane)

Lena	**PVO-K**			**Proposed**			
B size	EC	Payload	PSNR	EC	Payload	PSNR	Gain in EC
2x2	47000	0.18	51.76	66000	0.25	48.01	19000
2x3	35800	0.14	53.36	58000	0.22	48.49	22200
3x3	24600	0.09	55.41	47900	0.18	49.31	23300
3x4	18900	0.07	56.15	41000	0.16	49.92	22100
4x4	14200	0.05	57.37	33500	0.13	50.71	19300

4.4 Handling Special Blocks

In all PVO-based methods, the overflow/underflow problem must be taken into consideration, because the boundary pixel valued "0" and "255" will be beyond the gray scope after the expansion. The usual practice is to modify the boundary pixel values to a safe range and then construct a map to record the location of the modified pixels. In the proposed scheme, the overflow/underflow may occur in the pixels which are "0," "1," "254," and "255." If these pixels are handled using traditional methods, we must use two bits to record a modified pixel, which leads to a doubling of the size of the location map. In addition, since the block with same pixel values and the normal block cannot be used in the same manner to embed secret data, if data are embedded in the block with the same pixel values, we must establish another location map to record these blocks; however, this treatment often outweigh the benefits. In response to this phenomenon, we proposed an alternative way of handling special blocks; our location map is no longer constructed in a pixel unit, but it records information of each block instead. If the block may overflow/underflow, record it as "2"; if all pixel values in the block are equal, "1" is recorded, the rest blocks are recorded as "0."

Table 4 shows a comparison of the performance of using two overflow/underflow handling methods in the proposed scheme. The first method is to use the traditional way to build a pixel location map, and we used two bits to record the location of a modified pixel and abandoned embedding secret data in the block with the same pixel values; the second is to create a block location map, use two bits to record a special block position, and do nothing with the block that may overflow/underflow. Also, the block with same pixel values is used to embed the secret information.

As can be seen from the data in Table 4, the second method has better performance in both PSNR and maximum embedding capacity. Therefore, it was selected by the proposed scheme. In addition, note that the location map of the selected method is smaller than that of the PVO-K scheme when the block size is greater than 2×2.

Table 4: Performance comparison of two overflow/underflow handling methods

Images	**Use pixel LM**		**Use block LM**	
	EC	**PSNR**	**EC**	**PSNR**
Lena	42600	48.25	44000	48.37
Baboon	14700	49.79	15000	49.90
Airplane	63000	48.14	66000	48.13
Barbara	33000	48.85	34000	48.84
Elaine	26300	49.17	28000	49.17
Lake	29700	49.03	30000	49.14
Boat	29500	48.98	30000	49.00
Peppers	35000	48.58	36000	48.67
Average	34225	48.85	35375	48.90

4.5 Comparison of the Performances of the Proposed Scheme and Several PVO-based Methods

In this section, we evaluate our proposed GePVO-K scheme by comparing it with several PVO-based methods. Because our GePVO-K scheme focuses on increasing the maximum embedding capacity of the cover image, the quality of the camouflaged image may be affected to some degree. Table 5 shows the comparison result of PVO, IPVO, PVO-K, as well as the proposed scheme by test-

ing eight standard gray-scale images; the block size in the experiments was 2×2.

From Table 5, we can conclude that the proposed scheme successfully improved the performance of the three compared methods in embedding capacity. The average embedding capacity of the proposed scheme was 9,625 bits more than PVO, 5,625 bits more than IPVO, and 6,375 bits more than PVO-K. Note that the improvement in the embedding capacity was more obvious for the images with more smooth area. As can be seen from the experimental results of "Airplane," the maximum embedding capacity increased by nearly 20,000 bits compared with PVO-K. With this significant improvement in embedding capacity, the quality of the image will inevitably be affected. But, in the case of substantially increasing the EC, the PSNR remained at a high level in our proposed scheme. In "Airplane," for example, after increasing the embedding capacity by nearly 20,000 bits, the PSNR was still 48.13 dB. Moreover, compared with two most advanced PVO-based RDH methods, PVO-K and PPVO, the proposed scheme also shows excellent performance in terms of embedding capacity under the premise of ensuring the quality of the stego-image. It achieves greatest average EC while maintaining the average PSNR in 48.91dB.

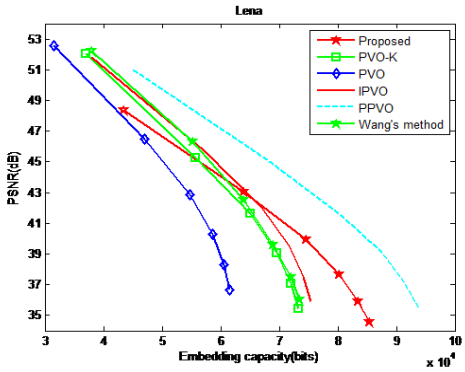
4.6 Multilevel Embedding Analysis

Like majority of the reversible data hiding algorithms, GePVO-K also supports multilevel embedding, that is, treating the camouflaged image as the new cover image to continue embedding secret data. Fig. 8 shows the multilevel embedding performance of PVO, IPVO, PVO-K, and the proposed GePVO-K by testing "Lena" and "Airplane."

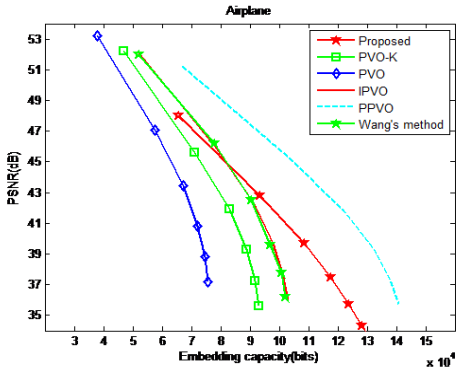
For the proposed scheme, the largest and smallest pixel values in the block were used to embed secret data. Since the secret data were randomly generated, assuming that the numbers of "0" and "1" were basically the same. Then, after embedding the secret data, the previous amount of largest-valued and smallest-valued pixels was halved, which means the embedding capacity of the next level also will be halved, and because the pixels are moved further, the quality of the camouflaged image will decrease step by step. Since the prediction error will be extended by embedding 1, the other three PVO-based methods will have a similar trend.

As can be seen from Fig. 8, the GePVO-K method was better than the embedding capacity of the other three methods (PVO, IPVO, PVO-K) at each embedding level, and the total EC has more obvious differences with the increasing of the embedding level. For image "Lena", the proposed method embedded a total of about 10,000 more bits of secret data than PVO-K and IPVO, and it increased EC by nearly 25,000 bits compared with PVO; for "Airplane," the total EC of our proposed method has 27,000 more bits than IPVO, 37,000 more bits than PVO-K, and nearly 54,000 more bits than PVO. At the same time, we can notice that when using multiple levels to embed the secret data, in the case of same EC (when EC is greater than 65,000 bits for "Lena", 82,000 bits for "Airplane"), the quality of the camouflaged image in the proposed scheme was better and the degree of image distortion did not decline sharply as it did in the other methods.

Since our proposed scheme aims to enhance the performance of the embedding capacity of PVO-K by embedding K bits of secret data into K largest-valued/smallest-valued pixels, the block constrains of PVO-K still not be broken. Due to the reason that the PPVO scheme not only breaks the block constrains but also reuses the pixels to embed secret data, it achieves best multilevel embedding performance. However, compared with another PVO-based scheme, i.e., Wang et al.s method, the proposed scheme has better performance in each embedding level.



(a) Lena



(b) Airplane

Figure 8: Multilevel embedding performance comparison of proposed and other five PVO-base methods

5 Conclusions

In this paper, we presented a new PVO-based RDH method, which is the enhancement on the basis of PVO-K method. It is an extension of PVO-K, and it no longer treats the largest and smallest values in the block as a unit to embed secret data, but it embeds the data in each of

Table 5: Performance comparison between proposed scheme and several PVO-based methods

Images	PVO		IPVO		PVO-K		PPVO		Wang's method		Proposed	
	EC	PSNR	EC	PSNR	EC	PSNR	EC	PSNR	EC	PSNR	EC	PSNR
Lena	32000	52.32	38000	52.35	37000	52.02	44000	51.25	38000	52.15	44000	48.37
Baboon	13000	51.75	13000	51.80	13000	52.00	15000	51.50	13000	51.85	15000	49.90
Airplane	38000	53.12	52000	52.50	47000	52.24	69000	50.95	52000	52.00	66000	48.13
Barbara	27000	52.24	29000	52.05	29000	52.16	33000	51.55	31000	52.45	34000	48.84
Elaine	21000	52.05	24000	52.00	23000	51.87	28000	50.00	25000	51.78	28000	49.17
Lake	23000	52.43	26000	51.79	26000	51.78	29000	52.65	26000	51.85	30000	49.14
Boat	24000	52.00	26000	51.80	26000	51.82	29000	51.20	26000	51.95	30000	49.00
Peppers	28000	52.05	30000	52.00	31000	51.92	33000	51.30	30000	52.00	36000	48.67
Average (payload)	25750 (0.098)	52.25	29750 (0.113)	52.04	29000 (0.111)	51.98	35000 (0.133)	51.30	30125 (0.115)	52.00	35375 (0.135)	48.91

the largest-valued and smallest-valued pixels. Therefore, the maximum embedding capacity has been improved significantly. Also, since it modifies more pixel values, the quality of the image will be subject to a certain decrease, but because of the modified pixel values may be shifted towards the original position in the procedure of embedding secret data in smallest-valued pixels, the quality of the camouflaged image can still be maintained at a high level in the case of maximum embedding capacity. And when embedding secret data with multiple levels, the proposed method achieved better performance in both EC and image quality than the other four PVO-based methods.

The proposed scheme also established a block location map instead of a pixel-based location map, and it processed each block in the different cases. It skipped the blocks that may overflow/underflow, and utilized blocks with the same pixel values, which further increased the embedding capacity; in addition, the quality of the camouflaged image was improved to some extent due to the reduced size of the location map and the smaller modification of the amplitudes of the pixel values.

Acknowledgments

This research work was partially supported by the Ministry of Science and Technology of the Republic of China under the Grant No. MOST105-2221-E-324 -014 and MOST 103-2632-E-324-001-MY3.

References

[1] K. Bharanitharan, C. C. Chang, H. R. Yang, and Z. H. Wang, "Efficient pixel prediction algorithm for reversible data hiding," *International Journal of Network Security*, vol. 18, no. 4, pp. 750–757, 2016.

[2] M. U. Celik, G. Sharma, A. M. Tekalp, *et al.*, "Lossless watermarking for image authentication: a new framework and an implementation," *IEEE Transactions on Image Processing*, vol. 15, no. 4, pp. 1042–1049, 2006.

[3] M. U. Celik, G. Sharma, A. M. Tekalp, and E. Saber, "Lossless generalized-lsb data embedding," *IEEE Transactions on Image Processing*, vol. 14, no. 2, pp. 253–266, 2005.

[4] C. K. Chan and L. M. Cheng, "Hiding data in images by simple lsb substitution," *Pattern Recognition*, vol. 37, no. 3, pp. 469–474, 2004.

[5] J. Fridrich, M. Goljan, and R. Du, "Lossless data embedding: new paradigm in digital watermarking," *EURASIP Journal on Applied Signal Processing*, vol. 2002, no. 1, pp. 185–196, 2002.

[6] Y. Hu, H. K. Lee, and J. Li, "De-based reversible data hiding with improved overflow location map," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 19, no. 2, pp. 250–260, 2009.

[7] B. Jana, D. Giri and S. K. Mondal, "Dual-image based reversible data hiding scheme using pixel value difference expansion," *International Journal of Network Security*, vol. 18, no. 4, pp. 633–643, 2016.

[8] L. Kamstra and H. J. Heijmans, "Reversible data embedding into images using wavelet techniques and sorting," *IEEE Transactions on Image Processing*, vol. 14, no. 12, pp. 2082–2090, 2005.

[9] S. K. Lee, Y. H. Suh, and Y. S. Ho, "Reversiblee image authentication based on watermarking," in *2006 IEEE International Conference on Multimedia and Expo*, pp. 1321–1324, IEEE, 2006.

[10] F. Li, Q. Mao, and C. C. Chang, "A reversible data hiding scheme based on iwt and the sudoku method," *International Journal of Network Security*, vol. 18, no. 3, pp. 410–419, 2016.

[11] X. Li, J. Li, B. Li, and B. Yang, "High-fidelity reversible data hiding scheme based on pixel-value-ordering and prediction-error expansion," *Signal Processing*, vol. 93, no. 1, pp. 198–205, 2013.

[12] M. Liu, H. S. Seah, C. Zhu, W. Lin, and F. Tian, "Reducing location map in prediction-based difference expansion for reversible image data embedding," *Signal Processing*, vol. 92, no. 3, pp. 819–828, 2012.

[13] J. Mielikainen, "Lsb matching revisited," *IEEE Signal Processing Letters*, vol. 13, no. 5, pp. 285–287, 2006.
[14] Z. Ni, Y. Q. Shi, N. Ansari, and W. Su, "Reversible data hiding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 16, no. 3, pp. 354–362, 2006.
[15] B. Ou, X. Li, Y. Zhao, and R. Ni, "Reversible data hiding using invariant pixel-value-ordering and prediction-error expansion," *Signal Processing: Image Communication*, vol. 29, no. 7, pp. 760–772, 2014.
[16] F. Peng, X. Li, and B. Yang, "Adaptive reversible data hiding scheme based on integer transform," *Signal Processing*, vol. 92, no. 1, pp. 54–62, 2012.
[17] F. Peng, X. Li, and B. Yang, "Improved pvo-based reversible data hiding," *Digital Signal Processing*, vol. 25, pp. 255–265, 2014.
[18] X. Qu and H. J. Kim, "Pixel-based pixel value ordering predictor for high-fidelity reversible data hiding," *Signal Processing*, vol. 111, pp. 249–260, 2015.
[19] D. M. Thodi and J. J. Rodríguez, "Expansion embedding techniques for reversible watermarking," *IEEE Transactions on Image Processing*, vol. 16, no. 3, pp. 721–730, 2007.
[20] J. Tian, "Reversible data embedding using a difference expansion," *IEEE Transactions on Circuits Systems for Video Technology*, vol. 13, no. 8, pp. 890–896, 2003.
[21] X. Wang, J. Ding, and Q. Pei, "A novel reversible image data hiding scheme based on pixel value ordering and dynamic pixel block partition," *Information Sciences*, vol. 310, pp. 16–35, 2015.
[22] X. Wang, X. Li, B. Yang, and Z. Guo, "Efficient generalized integer transform for reversible watermarking," *IEEE Signal Processing Letters*, vol. 17, no. 6, pp. 567–570, 2010.
[23] Y. L. Wang, J. J. Shen, M. S. Hwang, "An improved dual image-based reversible hiding technique using LSB matching", *International Journal of Network Security*, vol. 19, no. 5, pp. 858–862, 2017.
[24] Z. H. Wang, X. Zhuang, C. C. Chang, C. Qin, and Y. Zhu, "Reversible data hiding based on geometric structure of pixel groups," *International Journal of Network Security*, vol. 18, no. 1, pp. 52–59, 2016.
[25] S. Zhang, T. Gao, L. Yang, "A reversible data hiding scheme based on histogram modification in integer DWT domain for BTC compressed images," *International Journal of Network Security*, vol. 18, no. 4, pp. 718–727, 2016.
[26] X. Zhang and S. Wang, "Efficient steganographic embedding by exploiting modification direction," *IEEE Communications Letters*, vol. 10, no. 11, pp. 781–783, 2006.

Biography

Jian-Jun Li received the B.Sc. degree in information engineering from Xian University of Electronic Science and Technology, Xian, China, and the M.Sc. and Ph.D. degrees in electrical and computer from The University of Western Ontario and University of Windsor, Canada separately. He is currently working at HangZhou Dianzi University as a chair professor. His research interests include micro-electronics, audio, video and image processing algorithms and implementation.

Yun-He Wu is in his Third year of the master program at Hangzhou Dianzi University, Zhejiang, China, in 2016. His major is Computer Science and Technology. He received his BS degree in Computer Science and Technology in 2014 in Henan University of Science and Technology, Henan, China. His research interests include data hiding, video coding/decoding and image processing.

Chin-Feng Lee received Ph.D. degree in Computer Science and Information Engineering in 1998 from National Chung Cheng University in Taiwan. She is currently a professor in the Department of Information Management at Chaoyang University of Technology, Taiwan. Her research interests include steganography, image processing, and data mining.

Chin-Chen Chang received the B.S. degree in applied mathematics and the M.S. degree in computer and decision sciences from National Tsing Hua University, Hsinchu, Taiwan, R.O.C., in 1977 and 1979, respectively. He received the Ph.D. degree in computer engineering from National Chiao Tung University, Hsinchu, in 1982. From July 1998 to June 2000, he was Director of the Advisory Office, Ministry of Education, R.O.C. From 2002 to 2005, he was a Chair Professor at National Chung Cheng University. From February 2005, he has been a Chair Professor at Feng Chia University. In addition, he was severed as a consultant to several research institutes and government departments. He is currently a Fellow of IEEE and a Fellow of IEE, UK. His current research interests include database design, computer cryptography, image compression, and data structures.