Multimed Tools Appl
https://doi.org/10.1007/s11042-018-6410-x

Reversible data hiding using the dynamic block-partition strategy and pixel-value-ordering

Wengui Su [1,2,3] · Xiang Wang [1] · Fu Li [2] · Yulong Shen [3] · Qingqi Pei [1]

Received: 2 April 2018 / Revised: 29 June 2018 / Accepted: 11 July 2018

Abstract The reversible watermarking algorithm based on pixel value ordering (PVO) embeds secret data by modifying the maximum and minimum values in a pixel block. The performance of this algorithm heavily depends on the inherent correlation among adjacent pixels within an image block. To improve the inherent correlation among adjacent pixels within an image block, a new method to incorporate the dynamic block-partition strategy into the PVO algorithm is proposed in this paper. This new method includes the following procedure: First, the host image is divided into non-overlapping regions according to the image pixel values; then, each region is partitioned into different blocks, which are subsequently classified according to the local complexity and predefined threshold values for embedding; and finally, watermark embedding is performed using the PVO-based algorithm. In the proposed method, the pixel values of each embedded block are located in a relatively small region to improve the inherent correlation and thereby enhance the embedding performance of PVO. The experimental results show that the proposed algorithm has a better embedding performance compared with that of the conventional PVO based algorithm.

✉ Xiang Wang
wangxiang@xidian.edu.cn

Wengui Su
wgsu@gxu.edu.cn

Fu Li
645290794@qq.com

Yulong Shen
ylshen@mail.xidian.edu.cn

Qingqi Pei
qqpei@xidian.edu.cn

[1] State Key Laboratory of Integrated Service Networks, Xidian University, Xi'an 710071, China

[2] School of Mechanical Engineering, Guangxi University, Nanning 530004, China

[3] School of Computer Science and Technology, Xidian University, Xi'an 710071, China

Published online: 14 August 2018

Keywords Reversible data hiding (RDH) · Pixel value ordering (PVO) · Prediction error expansion · Dynamic block-partition strategy

1 Introduction

A new information hiding technology called digital data hiding [42] embeds useful identification information directly into multimedia carriers. Although the embedded information is added, this new technology can still maintain the premium visual effect without affecting the performance of multimedia carriers. The advantages of digital data hiding mainly include its imperceptibility, robustness and high capacity. These characteristics make digital data hiding a good candidate for many applications, such as copyright protection, copyright authentication, medical image processing, multimedia management, and image video error concealment coding. Digital data hiding has become an important branch in the field of information security.

The first data hiding algorithm was based on the spatial domain algorithm. The spatial domain algorithm modifies some portion of the pixel values of the host image to create a space to hide data. The algorithm acquires the secret information by analyzing the inherent correlation among pixels during extraction. The disadvantages of these algorithms are their low resistance to attack, poor robustness, and weak immunity to filtering operations or distortion. In a field with high requirements for robustness, the spatial domain data hiding algorithm has been gradually replaced by the transform domain algorithm.

Reversible data hiding (RDH) was first presented in the 1999 patent of the Honsinger et al. [17]. RDH can fully extract embedded secret information and also completely restore the host signal [3, 20, 29]. RDH is mainly applied in fields with high quality requirements, such as law-enforcement, medical images processing, and military. These fields require that the original media be precisely recovered without any distortion after extracting the watermark information. In addition, RDH also has great application prospects in some circumstances that require accurate data or other scenarios with uncertain resolutions, such as the pre-press technology, image archive management and precious art.

So far, many RDH algorithms have been proposed. Early RDH algorithms compress a portion of the host image to create space for data embedding. The performances of these algorithms depend on compression algorithms [2, 4–6, 10, 43]. The drawbacks of these algorithms are the extreme limit of the embedding capacity. Two more efficient RHD algorithms, difference expansion (DE) and histogram shifting (HS), have been extensively investigated in the past few decades.

In 2003, Tian proposed the DE algorithm [35] with the idea of expanding the difference between two adjacent pixels and embedding one bit into the difference of the pixels of each pair if no overflow or underflow occurred. The DE algorithm can achieve a relatively high embedding capacity compared to compression-based works. However, Tian's difference expansion is limited to two adjacent pixels. In [1, 36], the hiding ability and computation efficiency of the DE algorithm were improved using difference expansion of vectors or pixel blocks of arbitrary lengths to replace the pair of adjacent pixels. A previous report [38] proposed an integer transform based on the invariability of the sum of pixel pairs and a concept of pairwise difference adjustment (PDA) to achieve smaller modifications of pairs and higher location map compression rates. Later, the authors of [7, 15, 36] extended the DE algorithm by adjusting the embedding block size according to the complexity of the image's texture. Thodi

et al. and Ou et al. [28, 33] developed the DE algorithm by expanding the prediction errors instead of the difference values for embedding, which can embed a large payload without causing a large distortion. This method is also called the prediction-error extension (PEE) RDH algorithm. Because the accuracy of the prediction dominates the performance of RDH algorithms, many prediction algorithms have been proposed to improve the prediction accuracy, such as median edge detection (MED) [34], gradient adjusted prediction (GAP) [41], simplified GAP (SGAP) [8], rhombus prediction [32] and so on.

The Histogram Shifting (HS) algorithm proposed by Ni et al. [27] is another remarkable RDH algorithm. Li et al. [23] described the general framework of this algorithm. The HS algorithm modifies pixels according to the values between the peak bin and zero bin of the histogram of an image to create an embedding space. HS provides high image quality and a high peak signal to noise ratio (PSNR). However, the embedding rate of HS is low. Lee and his team [21] improved the HS algorithm by utilizing a difference histogram to replace the normal pixel value histogram and obtained an improvement in both the embedding capacity (EC) and image quality compared to those of Ni et al.'s algorithm. Afterwards, other algorithms that achieved further improvements over the conventional HS were shown in many papers, such as methods that used the prediction error histogram [16], down sampling [18], and multilevel histogram modification [11, 26].

Because sorting helps to utilize the correlation among pixels to create extra embedding space, many researchers have carried out studies in the field of the combination of sorting and PEE to dramatically enhance the embedding performance. The variance sorting method introduced in [19] selected pixels located in the smooth region to embed data and efficiently compressed the location map, which offered good performance. Sachnev et al. [32] further developed the rhombus prediction model and the local complexity sorting method, and the experimental results showed that embedding performance outperformed that of contemporaneous RDH algorithms.

To achieve a higher prediction accuracy, a novel algorithm evolving PEE named the pixel-value-ordering (PVO) based RDH [24] was proposed. In the PVO-based RDH, the host image is divided into blocks of equal size according to a certain method, and the pixels in each block are sorted according to the pixel values. The maximum pixel (minimum, respectively) is predicted using the second largest pixel (second smallest, respectively). Data embedding is implemented by applying PEE to the prediction-error histogram. The PVO-based RDH algorithm achieves a good embedding performance due to the strong inherent correlation among pixels within a block. At the encoder, the marked image is first divided into non-overlapping equal-size blocks for embedding and the block size has a strong effect on the embedding performance. A larger block size produces a relatively sharper histogram [9, 13, 14, 22, 25, 31], but significantly reduces the EC. A smaller block size is recommended achieve a higher EC, which often leads to significant image quality degradation. In addition, the performance of the PSNR will fluctuate along with the changes in the block size. For PVO-based algorithms, it is an important issue to determine the appropriate block size to achieve a good imperceptibility versus capacity trade off. The aforementioned PVO algorithms simply use an exhaustive search to find the best block size. Wang et al. [37] presented a block partitioning strategy by dividing a relatively smooth 4×4 sized block into four 2×2 sized sub-blocks to improve the embedding performance, especially at a higher EC. In recent papers [39, 40], smoother regions are adaptively partitioned on the basis of local complexity for data embedding. Two-layer embedding or a different pixel modification method was further employed to reduce the embedding distortion to attain a better performance. A novel block partition strategy is introduced in [12].

To better exploit image redundancy and achieve a better embedding performance, the combination of n thresholds and n-stage blocking are presented to partition the image in the form of a binary tree.

Considering the impact of the block size on the embedding performance, we propose a PVO-based algorithm that includes a dynamic block partition strategy in this paper. Instead of simply dividing the image into non-overlapping equal-sized blocks, our algorithm first partitions the host image into blocks based on the pixel values and a threshold and then classifies each block into different types through two thresholds and block complexity to hide data. Each smooth block is further divided into four 2×2 sized sub-blocks to embed data. Our algorithm better exploits the inherent correlation among the adjacent pixels. As a result, the EC and embedding performance are improved compared to those of three state-of-the-art schemes.

The rest of this paper is organized as follows. Section II provides an introduction of two conventional PVO-based RDH algorithms. Section III introduces the proposed PVO-based embedding and extraction procedures in detail. Section IV compares the experiment results and performance with those of Peng et al.'s PVO-based RDH algorithms. Section V summarizes and concludes this paper.

2 Related works

In this section, we briefly present the basic principles of Li et al.'s and Peng et al.'s PVO-based algorithms in the case of prediction-error expansion (PEE).

2.1 Li et al.'s work

In Li et al.'s [24] PVO-based algorithm, both the maximum and minimum pixel values in an image block are exploited to create embedding spaces in which sufficient payloads can be embedded with low distortion. As the modifications for the maximum and minimum pixels are similar, we take the maximum-modification-based embedding phases as examples. The main idea of this method can be summarized as follows.

This algorithm first divides the host image into non-overlapping and equal-sized blocks. For a given block $X = \{x_1, x_2, \ldots, x_n\}$ containing n pixels, its value $(x_1, x_2, \ldots, x_n)$ is sorted in ascending order to obtain$(x_{\sigma(1)}, \ldots, x_{\sigma(n)})$, where $\sigma : (1, 2, \ldots, n) \rightarrow (1, 2, \ldots, n)$ denotes unique one-to-one mapping such that $x_{\sigma(1)} \leq x_{\sigma(2)} \leq \ldots \leq x_{\sigma(\mathrm{n})}$. We assign $\sigma(i) < \sigma(j)$ if the conditions $x_{\sigma(i)} = x_{\sigma(j)}$ and $i < j$ are satisfied. Then, the second largest value $x_{\sigma(n-1)}$ is used to predict the maximum $x_{\sigma(n)}$. The corresponding prediction-error of the maximum pixel is computed as follows:

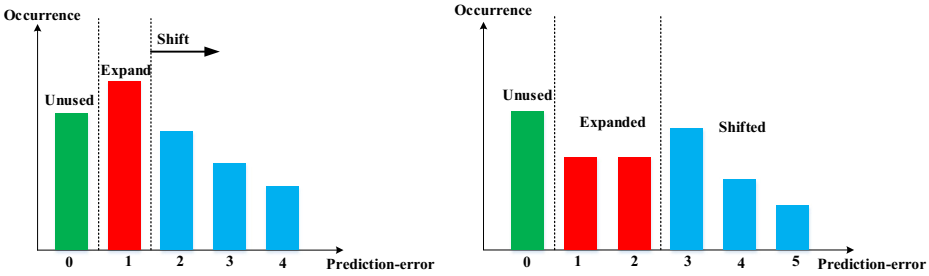
$$PE_{\max} = x_{\sigma(n)} - x_{\sigma(n-1)} \tag{1}$$

After calculating the prediction errors for all blocks, a histogram consisting of PE_{max} is generated (see Fig. 1(a) for an illustration).

In this case, the maximum $x_{\sigma(n)}$ is modified to obtain the encoded pixel value $\hat{x}_{\sigma(n)}$ by taking

$$\hat{x}_{\sigma(n)} = \begin{cases} x_{\sigma(n)}, & \text{if } PE_{\max} = 0 \\ x_{\sigma(n)} + b, & \text{if } PE_{\max} = 1 \\ x_{\sigma(n)} + 1, & \text{if } PE_{\max} > 1 \end{cases} \tag{2}$$

where $b \in \{0, 1\}$ is the data bit to be embedded.



(a) Before data embedding (b) After data embedding

Fig. 1 Histogram of PE_{max} before and after data embedding. Bin1 (marked in red) is expanded for embedding, bin0 (marked with green) remains unchanged, and the blue bins (bins larger than 1) are shifted

At the decoder side, the marked value $(y_1, \ldots, y_n)$ is sorted in ascending order to obtain $(y_{\sigma(1)}, \ldots, y_{\sigma(n)})$, and then, the prediction error $\hat{\mathrm{P}}\mathrm{E}_{\max}$ is calculated as follows:

$$\hat{P}E_{\max} = y_{\sigma(n)} - y_{\sigma(n-1)} \tag{3}$$

As shown in Fig. 1(b), $\hat{P}E_{\max}$ is embedded only when $\hat{P}E_{max} \in^{\max} \{1, 2\}$.

The data extraction procedure is described as follows.

For each block, the original $x_{\sigma(n)}$ can be recovered as follows:

- If $\hat{P}E_{max} \in^{\max} \{1, 2\}$,the embedded data bit is $\mathrm{b} = \hat{\mathrm{P}}\mathrm{E}_{\max} - 1$ and the original maximum is $x_{\sigma(n)} = y_{\sigma(n)} - \mathrm{b}$;
- If $\hat{\mathrm{P}}\mathrm{E}_{\max} > 2$, there is no hidden data and the original maximum is $x_{\sigma(n)} = y_{\sigma(n)} - 1$; and
- If $\hat{\mathrm{P}}\mathrm{E}_{\max} = 0$, there is no hidden data and the original maximum is $x_{\sigma(n)} = y_{\sigma(n)}$.

2.2 Peng et al.'s work

Peng et al. [30] designed a new prediction difference by introducing bin 0 for expansion embedding to produce a higher EC than conventional PVO-based algorithms. For a given EC, Peng et al.'s algorithm utilizes lager sized blocks and can better exploit image redundancy to achieve a superior embedding performance. The main procedures of the maximum-modification-based embedding are depicted as follows.

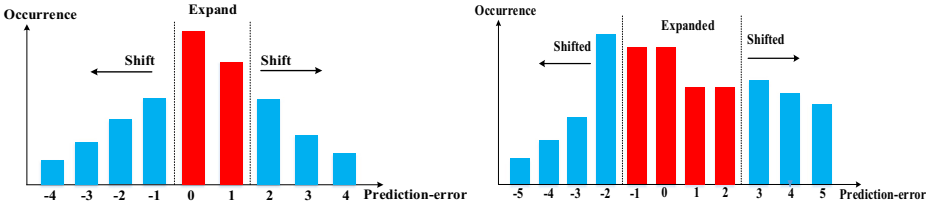
Similar to Li et al.'s work, first the blocks are sorted in ascending order to obtain $\left(x_{\sigma_{(1)}}, \ldots, x_{\sigma_{(n)}}\right)$. However, the method to calculate the prediction error is different than that of Li et al.'s PE_{max} and is defined as follows:

$$d_{\max} = X_u - X_v \tag{4}$$

where

$$\begin{cases} u = \min(\sigma(n), \sigma(n-1)) \\ v = \max(\sigma(n), \sigma(n-1)) \end{cases} \tag{5}$$

The histogram of d_{max} is shown in Fig. 2, in which bin0 and bin1 are utilized for expansion embedding and the other bins are shifted to create the embedding space.



(a) Before data embedding (b) After data embedding

Fig. 2 Histogram of d_{max} before and after data embedding

The marked value of the maximum $\hat{x}_{\sigma(n)}$ is calculated by the following:

$$\hat{x}_{\sigma(n)} = \begin{cases} x_{\sigma(n)} + b, & if\ d_{\max} = 1 \\ x_{\sigma(n)} + 1, & if\ d_{\max} > 1 \\ x_{\sigma(n)} + b, & if\ d_{\max} = 0 \\ x_{\sigma(n)} + 1, & if\ d_{\max} < 0 \end{cases} \tag{6}$$

where b∈{0,1} is the data to be embedded.

In this method, the unchanged blocks with $x_{\sigma(n)} = x_{\sigma(n-1)}$ in Li et al.'s algorithm are exploited for data embedding and bin0 is usually the histogram peak in the histogram of d_{max}. In addition, because Peng et al.'s algorithm utilizes the relative error instead of the absolute error, the histogram of $PE_{\max}$ and d_{max} has the following correspondence:

$$\begin{cases} PE_{\max}(0) = d_{\max}(0) \\ PE_{\max}(k) = d_{\max}(k) + d_{\max}(-k)\ k\ is\ a\ non{-}zero\ integer, k > 0 \end{cases} \tag{7}$$

Li et al.'s algorithm only utilizes $PE_{max}(1)$ for data embedding, whereas Peng et al. uses both $d_{max}(0)$ and $d_{max}(1)$. Because $PE_{\max}(1) < d_{\max}(0) + d_{\max}(1)$, Peng et al.'s algorithm provides a higher EC than Li et al.'s. The EC increment can be determined as follows:

$$\mathrm{d}_{\max}(0) + \mathrm{d}_{\max}(1) - PE_{\max}(1) \approx \mathrm{d}_{\max}(0) - PE_{\max}(1)/2 \tag{8}$$

At the decoder, the marked blocks are first sorted to obtain $\left(y_{\sigma_{(1)}}, \ldots, y_{\sigma_{(n)}}\right)$. Then, the prediction error $\hat{d}_{\max}$ can be obtained as follows:

$$\hat{\mathrm{d}}_{\max} = y_u - y_v \tag{9}$$

The data extraction procedure is described as follows.

1. If $\hat{\mathrm{d}}_{\max} > 0$, the decoder identifies $y_u > y_v$. For $\sigma(n) < \sigma(n-1)$, $u = \sigma(n)$ and $v = \sigma(n-1)$, the following two cases are applied.

- If $\hat{\mathrm{d}}_{\max} \in \{1, 2\}$, the embedded data are $\mathrm{b} = \hat{d}_{\max} - 1$, and then, the original value of $x_{\sigma(n)}$ can be recovered as $x_{\sigma(n)} = y_u - b$.
- If $\hat{d}_{\max} > 2$, there is no hidden data, and then, the original value of $x_{\sigma(n)}$can be recovered as$x_{\sigma(n)} = y_u - 1$.

2. If $\hat{d}_{\max} \leq 0$, the decoder identifies $y_u \leq y_v$. For $\sigma(n) > \sigma(n-1)$, $u = \sigma(n-1)$ and $v = \sigma(n)$, the following two cases are applied.

- If $\hat{d}_{max} \in \{0, -1\}$, the embedded data are $b = -\hat{d}_{max}$, and then, the original value of $x_{\sigma(n)}$ can be recovered as $x_{\sigma(n)} = y_v - b$.
- If $\hat{d}_{max} < -1$, there is no hidden data, and then, the original value of $x_{\sigma(n)}$ can be recovered as $x_{\sigma(n)} = y_v - 1$.

3 Proposed method

3.1 Dynamic block-partition strategy

In existing PVO-based algorithms, data embedding is implemented on pixel blocks that are partitioned according to the pixel coordinates of the image. Although these algorithms can make better use of the inter-pixel correlation, they cannot achieve better results in regions with complex textures, especially in regions with more noise. Aiming for a better utilization of the inherent correlation among pixels and to achieve a good embedding performance, we propose a PVO-based algorithm with the strategy of block-partitioning by pixel values. In this new method, the host image is first partitioned into blocks according to pixel values, and then, PVO embedding is performed within each block. In this way, the distribution of the pixel values in each block is narrowed down to a smaller range and the inherent correlation among pixels is further enhanced. Therefore, the prediction accuracy is improved and the embedding effect is enhanced.

The partition is finished by two steps. First, we perform partitioning on the host image to construct multiple regions. Second, we perform block partitioning over a single region to construct smaller blocks for each region. The detailed descriptions of these two steps are provided in sections 3.1.1 and 3.1.2, respectively.

3.1.1 Region partition

Suppose the host image I is a gray-scale image with n pixels. We first partition the host image into several regions. The step-by-step block partitioning scheme is detailed as follows.

Step 1: For all pixels $I = \{x_1, \ldots, x_n\}$ in host image I, $\{x_1, \ldots, x_n\}$ are sorted in descending order to obtain a set $I_\sigma = \{x_{\sigma(1)}, \ldots, x_{\sigma(n)}\}$, where $\sigma:(1, 2, 3, \ldots, n) \rightarrow (1, 2, 3, \ldots, n)$ is unique one to one mapping such that $x_{\sigma(1)} \geq x_{\sigma(2)} \geq \ldots \geq x_{\sigma(n)}$. Here, we assign $\sigma(i) < \sigma(j)$ if $x_{\sigma(i)} = x_{\sigma(j)}$ and $i < j$. Select the maximum pixel $x_{\sigma(1)}$ found in the raster scan sequence as the reference point of the partition and add it to the set $B_i (i = 1, \ldots, n)$. For the consistency of region-partitioning at the decoder, 1 is added to the pixel value of the reference point such that $x_{\sigma(1)} = x_{\sigma(1)} + 1$. Moreover, if there is a pixel with a value of 255, modify its value to 254 to avoid overflow.

Step 2: In the host image I, traverse the eight neighboring pixels ${x^1}_{\sigma(1)}$, ${x^2}_{\sigma(1)}$, ${x^3}_{\sigma(1)}$, ${x^4}_{\sigma(1)}$, ${x^5}_{\sigma(1)}$, ${x^6}_{\sigma(1)}$, ${x^7}_{\sigma(1)}$, ${x^8}_{\sigma(1)}$ surrounding the reference point $x_{\sigma(1)}$ if a pixel $x^k_{\sigma(1)}$ $(k = 1, \ldots, 8)$ satisfies the following:

$$x_{\sigma(1)} - T \leq x^k_{\sigma(1)} \leq x_{\sigma(1)} \tag{10}$$

Add pixel $x^{k}_{\sigma(1)}$ into set B_i and then remove it from I_σ. T is the predefined threshold used to control the pixel value range of a single region. A smaller T results in more partitions, and the pixel values in each partition are closer.

Step 3: Use every pixel value from B_i as a reference point and repeat step 2 to generate a new B_i. The loop stops when eight neighboring pixels of all pixels in B_i no longer satisfy *Eq.* 10.

Step 4: Perform image dilation on B_i in host image I and pick pixels on the dilation boundary to construct the set E_i $(i = 1,...,n)$. Finally, remove all of the pixels in E_i from I_σ. A pixel region surrounding the reference point $x_{\sigma(1)}$ is generated.

For a reference point $x_{\sigma(1)}$, two sets, B_i and E_i, are generated during the process of block-partitioning, where B_i is embedding and E_i is the determination of B_i at extraction. Search the next reference point in image I and perform the above steps in the new set I_σ after removing B_i and E_i until the set I_σ is empty.

Since the maximum modification of the pixel value modification in PVO is 1, the pixel value of the reference point should be incremented by 1 to ensure that it is still the maximum point in the region during extraction. When extracting, the reference point is first found and subtracted by 1 from its original value. In this case, *Eq.* 10 is still satisfied and the original pixel-region is recovered through the known conditions and boundary region.

To make the region partition procedure clearer, we provide an example to illustrate the partition procedure.

Step 1: Assuming that an 8-bit host image of size 10×10 is used as an example and the predefined threshold is $T = 10$, a region partition is started from the first maximum pixel found in the raster scan sequence (as shown in Fig. 3 (a)). The pixel with a value of 240 in the fourth row is the first partition reference point.

Step 2: Modify the pixel with a value of 240, which is the reference point to 241, and then, perform the first eight-neighborhood-pixel search surrounding the reference point. Remove the pixels that satisfy *Eq.* 10 from I_σ and add them to B_1. The region B_1 constructed from the first search is marked in red, as shown in Fig. 3 (b).

Step 3: Perform eight-neighborhood searches over all of the pixels in B_1 until the eight neighboring pixels of all of the pixels in B_1 no longer satisfy *Eq.* 10. The results of the second and final search are marked in red, as shown in Fig.3 (c) and (d), respectively.

Step 4: Perform image dilation over the host image based on set B_1 and pick the pixels on the block boundary to construct the set E_1. The first region partition is completed and marked in green and red in Fig. 3(e). After removing the pixels of B_1 and E_1 from I_σ, the remaining pixels in I_σ are traversed to find the next maximum pixel. That is, the second partition operation is performed based on the pixel with a value of 240 in the eighth row (as shown in Fig. 3 (a)).

As an example, we perform the above region partition on the image Lena, and the results are illustrated in Fig. 4.

223	225	221	228	225	225	227	226	229	228
221	228	225	221	228	223	226	229	229	227
228	223	234	238	239	237	235	227	225	229
227	226	231	235	240	237	226	225	223	222
224	223	234	233	231	231	232	228	227	223
221	225	227	231	229	231	232	232	221	221
220	229	230	233	229	228	229	226	227	229
229	227	226	227	228	225	240	227	228	225
224	224	223	222	222	221	223	224	225	226
224	225	226	224	224	225	223	224	226	227

(a) Maximum pixel points

223	225	221	228	225	225	227	226	229	228
221	228	225	221	228	223	226	229	229	227
228	223	234	238	239	237	235	227	225	229
227	226	231	235	240	237	226	225	223	222
224	223	234	233	231	231	232	228	227	223
221	225	227	231	229	231	232	232	221	221
220	229	230	233	229	228	229	226	227	229
229	227	226	227	228	225	240	227	228	225
224	224	223	222	222	221	223	224	225	226
224	225	226	224	224	225	223	224	226	227

(b) Result of the first search

223	225	221	228	225	225	227	226	229	228
221	228	225	221	228	223	226	229	229	227
228	223	234	238	239	237	235	227	225	229
227	226	231	235	240	237	226	225	223	222
224	223	234	233	231	231	232	228	227	223
221	225	227	231	229	231	232	232	221	221
220	229	230	233	229	228	229	226	227	229
229	227	226	227	228	225	240	227	228	225
224	224	223	222	222	221	223	224	225	226
224	225	226	224	224	225	223	224	226	227

(c) Result of the second search

223	225	221	228	225	225	227	226	229	228
221	228	225	221	228	223	226	229	229	227
228	223	234	238	239	237	235	227	225	229
227	226	231	235	240	237	226	225	223	222
224	223	234	233	229	231	232	228	227	223
221	225	227	231	229	231	232	232	221	221
220	229	230	233	229	228	229	226	227	229
229	227	226	227	228	225	240	227	228	225
224	224	223	222	222	221	223	224	225	226
224	225	226	224	224	225	223	224	226	227

(d) Result of the first block partition

223	225	221	228	225	225	227	226	229	228
221	228	225	221	228	223	226	229	229	227
228	223	234	238	239	237	235	227	225	229
227	226	231	235	240	237	226	225	223	222
224	223	234	233	231	231	232	228	227	223
221	225	227	231	229	231	232	232	221	221
220	229	230	233	229	228	229	226	227	229
229	227	226	227	228	225	240	227	228	225
224	224	223	222	222	221	223	224	225	226
224	225	226	224	224	225	223	224	226	227

(e)Block partition with expansion

Fig. 3 Illustration of the region partition

3.1.2 Block partition

The host image is divided into several regions B_i ($i = 1,...,n$) via the region partition method detailed in section 3.1.1. For a single region B_1, find the first 16 pixels with the shortest distance from the upper left corner of the region to form a block d_1, and then, remove the pixels of d_1 from B_1. Blocks d_j ($j = 2,...,n$) are obtained and removed from B_1 using the

(a) (b) (c) (d)

Fig. 4 Region partitioning in Lena. (a) Original image Lena. (b)The reference point is marked in red. (c) The green region represents the pixels from the eight-neighboring-pixel searches. (d) The blue boundary represents the partitioned region boundary after image dilation

above method. Block partitioning will be end when the number of remaining pixels in B_1 is less than 16, and all of the remaining pixels will be discarded. Block partitioning occurs as shown in Fig. 5. The red pixel represents the reference point with from the partition starts. The green part is the 16-pixel sub block d_j ($j = 1,...,n$), and the blue part are discarded blocks whose pixel numbers are less than 16.

Following the steps mentioned above, the remaining regions B_i ($i = 2,...,n$) are divided into blocks. Then, all blocks d_j ($j = 1,...,n$) are added into a set, A, $A = \{d_1, \ldots, d_n\}$, where $d_1,...,d_n$ represent a block composed of 16 pixels of each; data embedding is performed based on the PVO algorithm proposed by Peng et al.

3.1.3 Block complexity computation and block classification

The noise level (*NL*) is utilized to evaluate the textural complexity of a pixel block and is an important metric to determine the smoothness of a block. The proposed method uses *NL* as the block complexity measurement and two predefined thresholds to determine that a block is normal or smooth. The smooth block is further divided into four sub blocks to improve the embedding capacity.

In our method, we divide each block d_j ($j = 1,...,n$) in set A into four sub blocks, $S_k = \{x_1^k, \ldots, x_n^k\}$ ($k \in \{1, 2, 3, 4\}$). For each sub block S_k, sort its value in ascending order to obtain the following:

$$\begin{cases} \mathrm{S}_1 : \left(x_1^1, x_2^1 \ldots x_n^1\right) \rightarrow \left(x_{\varphi(1)}^1, x_{\varphi(2)}^1 \ldots x_{\varphi(n)}^1\right). \\ \mathrm{S}_2 : \left(x_1^2, x_2^2 \ldots x_n^2\right) \rightarrow \left(x_{\varphi(1)}^2, x_{\varphi(2)}^2 \ldots x_{\varphi(n)}^2\right). \\ \mathrm{S}_3 : \left(x_1^3, x_2^3 \ldots x_n^3\right) \rightarrow \left(x_{\varphi(1)}^3, x_{\varphi(2)}^3 \ldots x_{\varphi(n)}^3\right). \\ \mathrm{S}_4 : \left(x_1^4, x_2^4 \ldots x_n^4\right) \rightarrow \left(x_{\varphi(1)}^4, x_{\varphi(2)}^4 \ldots x_{\varphi(n)}^4\right). \end{cases} \tag{11}$$

where $\varphi : (1, 2, 3\cdots, n) \rightarrow (1, 2, 3\cdots, n)$ is unique one-to-one mapping. The block complexity *NL* is given by the following:

$$NL = \max\left\{x_{\varphi_1(n-1)}^1, x_{\varphi_2(n-1)}^2, x_{\varphi_3(n-1)}^3, x_{\varphi_4(n-1)}^4\right\} - \min\left\{x_{\varphi_1(2)}^1, x_{\varphi_2(2)}^2, x_{\varphi_3(2)}^3, x_{\varphi_4(2)}^4\right\} \tag{12}$$

NL is invariant after embedding and can be obtained by the decoder, which guarantees reversibility when using the block-partition strategy.

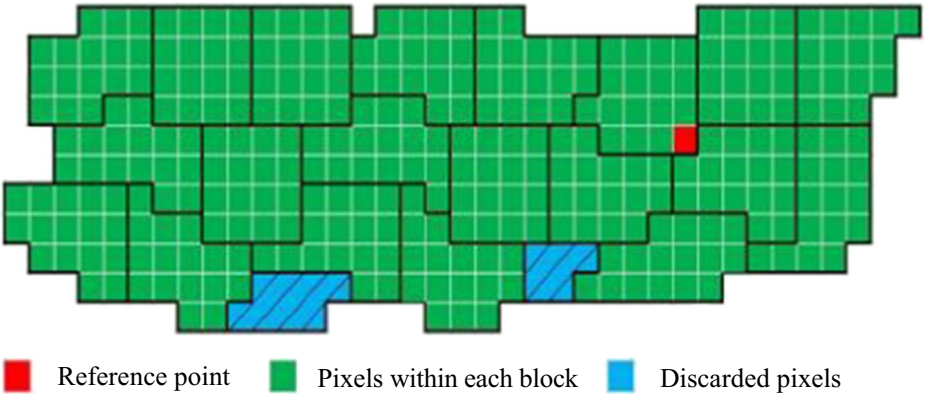


Fig. 5 Illustration of block partitioning

We classify each block d_j ($j = 1,...,n$) into three types based on NL and the two given thresholds T_1 and $T_2(-1 \le T_2 \le T_1 \le 255)$.

- A block d_j with $NL > T_1$ is regarded as a rough block and is not utilized in data embedding.
- A block d_j with $T_2 < NL \le T_1$ is regarded as a normal block for embedding. According to the PVO algorithm [30], a normal block d_j can carry 2 bits of information.
- A block d_j with $NL \le T_2$ is regarded as a smooth block and can be further divided into four 2×2 sized blocks to embed more data. According to the PVO algorithm [30], each 2×2 sized block can carry 2 bits of information, and thus, a smooth block d_j can carry 8 bits of information.

3.2 Data embedding procedure

The embedding steps are described as follows.

Step 1: Image partition.

After scanning the pixel values of the host image, the image is partitioned into non-overlapping regions $B_i(i = 1, \ldots, n)$ using the method described in section 3.1.1.

Following the method described in section 3.1.2, divide all regions $B_i(i = 1, \ldots, n)$ into blocks d_j ($j = 1,...,n$) of 16 pixels each and add d_j ($j = 1,...,n$) to set A.

Step 2: Location map construction.

A location map (LM) is created to avoid the overflow/underflow problem and is then compressed to a bitstream. For pixel with values $(x_1,...,x_{16})$ in each embedding block d_j ($j = 1,...,n$), if $\min\{x_1, x_2, \cdots x_{16}\} = 0$ or $max\{x_1, x_2, \cdots x_{16}\} = 255$, the block is an exceptional block that would cause overflow or underflow and will not be used for embedding. In this case, we set the location map $LM(j) = 1$, and otherwise, $LM(j) = 0$. Consequently, the location map is losslessly compressed using arithmetic coding to reduce its length, and we denote the length of the compressed location map as l_{clm}.

Step 3: Data embedding.

For each block d_j ($j = 1,...,n$), its local complexity NL is calculated using *Eq.*12.

If $LM(j) = 1$, the block will cause overflow/underflow; skip this block.

If $LM(j) = 0$ and $NL > T_1$, the block is a rough block and cannot be utilized in data embedding. Thus, skip this block.

If $LM(j) = 0$ and $T_2 < NL \le T_1$, the block is an normal block. In this case, this block is located in a moderately flat region that can be utilized for data embedding. The maximum and minimum pixels are embedded using Peng et al.'s PVO algorithm [30].

If $LM(j) = 0$ and $NL \le T_2$, the block is a smooth one and can be further divided into four 2×2 sized blocks. The maximum and minimum pixels of each 2×2 sized block are embedded using Peng et al.'s PVO algorithm [30].

Repeat the above steps until all embeddable blocks are embedded and denote the index of last data-carrying block as ξ_{end}.

Step 4: **Au**xiliary information and location map embedding.

Record the least-significant-bit (*LSB*) of the first $24 + 2\lceil 1og_2(16\xi)\rceil + l_{clm}$ pixels of the image to generate a binary sequence S_{LSB}, where $\lceil \cdot \rceil$ is the ceiling function. Then, these *LSB* are replaced by the following auxiliary information and the compressed location map *LM:*

- Thresholds T (8 bits) for the region partition and T_1 (8 bits) and T_2 (8 bits) for block complexity;
- The end position $\xi_{\text{end}}(\lceil log_2(16\xi)\rceil bits)$;
- The length of the compressed location map $l_{clm}(\lceil \log_2(16\xi)\rceil bits)$; and
- The location map ($l_{clm} bits$).

Finally, the sequence S_{LSB} is embedded into the remaining blocks $\{d_{\xi_{end}+1}, \ldots, d_{\xi}\}$ using the same method in Step 3 to generate the marked image.

3.3 Data extraction procedure

As an inverse process of data embedding, the corresponding data extraction procedure is detailed as follows.

Step 1: Auxiliary information and location map extraction

Extract the auxiliary information, including T, T_1, T_2, ξ_{end} and l_{clm}, by reading the *LSB* of the first $24 + 2\lceil 1og_2(16\xi)\rceil$ pixels of the marked image. Then, the compressed location map is extracted by reading the *LSB* of next l_{clm} pixels. The location map *LM* is recovered through decompression of the compressed map.

Step 2: Image partition and block classification

Using the same method as in Step 1 of the data embedding, the marked image is partitioned into non-overlapping blocks and set A is generated.

Step 3: Sequence S_{LSB} extraction and image extraction.

The binary sequence S_{LSB} generated in Step5 of data embedding is extracted from the block sequence $\{d_{\xi_{end}+1}, \ldots, d_{\xi}\}$.

A block d_k ($k > \xi_{end+1}$) is classified by local complexity *NL*.

If $LM(k) = 0$ and$NL \le T_2$, d_k is a smooth block. Divide d_k into four 2×2 sized sub-blocks, and then, perform PVO-based extraction and image recovery for each

sub-block.

If $LM(k) = 0$ and$T_2 < NL \le T_1$, d_k is a normal block. Perform PVO-based extraction and image recovery directly on d_k .

Otherwise, for rough blocks or blocks with $LM(k) = 1$, no data are hidden and they do not perform any modifications, which means that the pixels in these blocks maintain their original values.

Step 4: Data extraction and image restoration

Replace the LSB of the first $24 + 2\lceil 1og_2(16\xi)\rceil + l_{clm}$ image pixels by the sequence S_{LSB}. Extract the hidden data and recover the host image from blocks $\{d_1, \ldots, d_{\xi_{end}}\}$ using the method depicted in Step 3. Thus, both the host image and secret data are fully recovered.

4 Experiment results

In this section, we conducted experiments using MATLAB to evaluate the embedding performance of our method. The peak signal-to-noise ratio (PSNR) and embedding capacity performance of our method were compared to three state-of-the-art methods proposed by Peng et al. [30] with 2 × 2 and 4 × 4 sized blocks (the 4 × 4 scheme and 2 × 2 scheme for short), Li et al. [24] and Sachnev et al. [32]. Eight 512 × 512 sized gray-scale images were utilized as samples in the testing. These images included Fishingboat, Peppers, Elaine, Goldhill, Lake, Lena, Einstein and Zelda. All of the test images were downloaded from the USC-SIPI image database. Fig. 6 shows the performance comparison using EC within the range of [5000 bits, maximum] and a step-size of 1000 bits.

Sachnev et al's method combines PEE with an embedding-position-selection strategy. It is a representative PEE method and is superior to many typical RDH methods. Referring to Fig. 6, it is obvious that the improvement of our method over Sachnev et al.'s is significant. Our method achieves a higher PSNR for every image, regardless of the EC.

Li et al.'s method is a high-fidelity RDH method. For the images Fishingboat, Peppers, and Elaine, our method shows a better performance for both the PSNR and EC. Especially, for the image Elaine, when EC is 10,000 bits, the PSNR of the proposed method outperforms Li et al.'s method by 3.1 dB. Our method achieves a good performance at a high EC, while there are poor performances at low ECs in Lake, Goldhill and Einstein. For Zelda, our method yields a higher PSNR when EC is smaller than 20,000 bits.

Peng et al.'s method is an extension of [24]. It is one of the best PVO-based algorithms and has an optimum performance. Compared with the improved PVO-based methods of Peng et al., our method consistently achieves a higher PSNR at a given capacity on the images Fishingboat, Peppers, and Elaine. For Lake and Goldhill, both the 4 × 4 scheme and 2 × 2 scheme are better than ours at a low EC. However, when EC increases, our method achieves a better performance. There are also some images that exhibit a good performance in our method at a low EC (e.g., the image Lena when EC is lower than 21,000 bits, and the image Zelda when EC is lower than 23,000 bits). For Einstein, our method is slightly better than the 2 × 2 scheme when EC∈ [13,000, 17,000] bits, while it is slightly worse at other EC values. Especially, the PSNR of our method significantly decreases at both higher and lower ECs.

Therefore, for most images, the 4 × 4 scheme is better than the 2 × 2 scheme on the PSNR, while it is worse on the EC. Obviously, a smaller block size is helpful to improve the EC, but

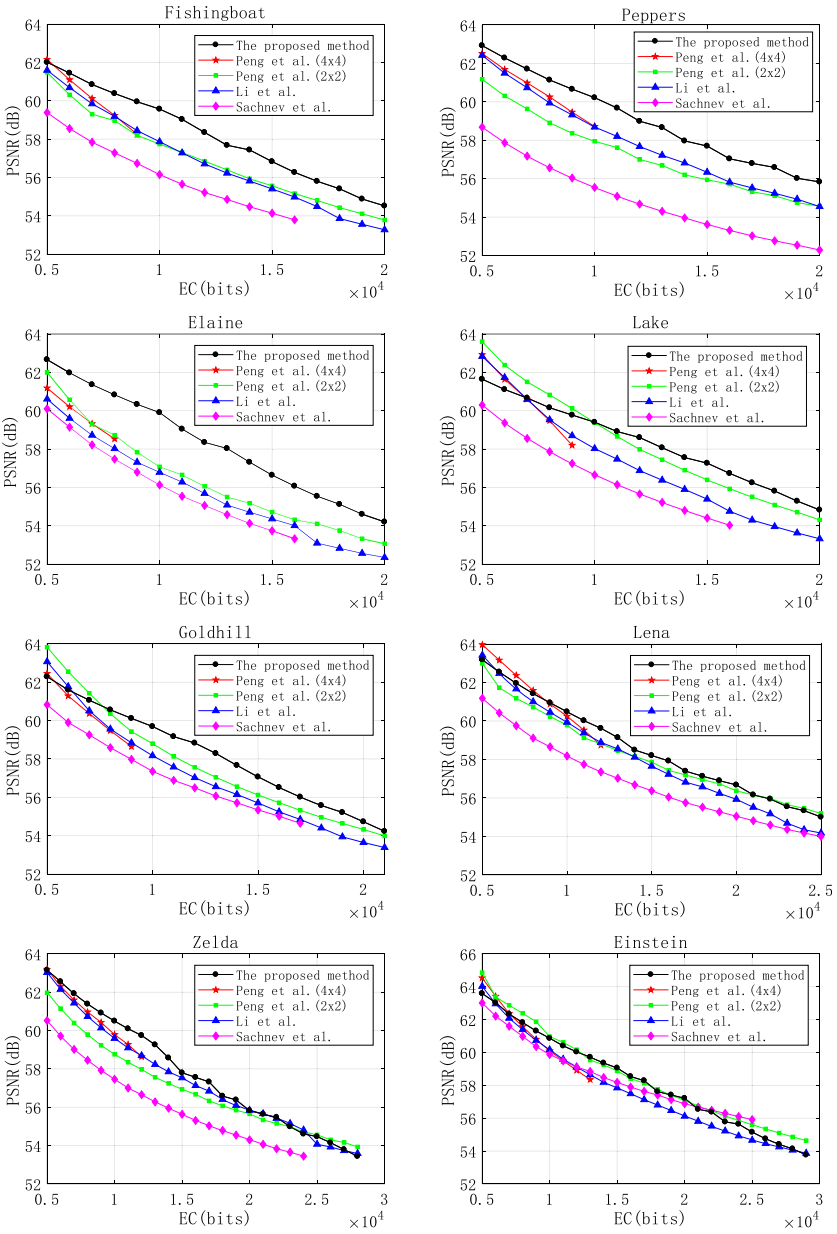


Fig. 6 Performance comparison between the proposed method and three methods of Sachnev et al. [32], Li et al. [24], and Peng et al. [30]

significantly reduces the PSNR. The maximum EC that is provided by the dynamic partitioning strategy based on the pixel value proposed in this paper is smaller than that of the 2×2 scheme, but is much larger than that of the 4×4 scheme.

5 Conclusion

The proposed reversible watermarking algorithm is a combination of an existing PVO algorithm and the dynamic block-partition strategy. The proposed method performs block partitioning according to the pixel value of the image and carries out PVO-based embedding in the blocks. Compared with other PVO algorithms, the proposed method provides a high inherent correlation within each embedding pixel block. Furthermore, by adjusting the pixel value of the reference point within each image region, consistency between the embedding side and extraction side is achieved. The comparison results show that the proposed method yields a better PSNR than conventional PVO algorithms at the same embedding rate.

Acknowledgements This work was supported in part by the National Key Research and Development Program of China (No. 2016YFB0800601) and the Key Basic Research Plan in Shaanxi Province (Grant No. 2017ZDXM-GY-014).

References

1. Alattar AM (2004) Reversible watermark using the difference expansion of a generalized integer transform. IEEE Trans Image Process 13(8):1147–1156
2. Awrangjeb M, Kankanhalli MS (2005) Reversible watermarking using a perceptual model. J Electro Imaging 14(1):013014–013014-8
3. Caldelli R, Filippini F, Becarelli R (2010) Reversible watermarking techniques: an overview and a classification, EURASIPJ . Inf. Secur. articleID134546
4. Celik MU, Sharma G, Tekalp AM, Saber E (2005) Lossless generalized-LSB data embedding. IEEE Trans Image Process 14(2):253–266
5. Celik MU, Sharma G, Tekalp AM (2006) Lossless watermarking for image authentication: a new framework and an implementation. IEEE Trans Image Process 15(4):1042–1049
6. Chang C-C, Kieu TD (2010) A reversible data hiding scheme using complementary embedding strategy. Inf Sci 180(16):3045–3058
7. Chen C-C, Tsai Y-H (2011) Adaptive reversible image watermarking scheme. J Syst Softw 84(3):428–434
8. Chen M, Chen Z, Zeng X, Xiong Z (2009) Reversible data hiding using additive prediction-error expansion. Proc 11th ACM Workshop Multimed Sec :19–24
9. Coatrieux G, Pan W, Cuppens-Boulahia N, Cuppens F, Roux C (2013) Reversible watermarking based on invariant image classification and dynamic histogram shifting. IEEE Trans Inform Forensics Sec 8(1):111–120
10. Fridrich J, Goljan M, Du R (2002) Lossless data embedding - new paradigm in digital watermarking, EURASIPJ. Appl Signal Process (2):185–196
11. He W, Xiong G, Zhou K, Cai J (2016) Reversible data hiding based on multilevel histogram modification and pixel value grouping. J Vis Commun Image R 40:459–469
12. He W, Cai J, Zhou K, Xiong G (2017) Efficient PVO-based reversible data hiding using multistage blocking and prediction accuracy matrix. J Vis Commun Image Represent 46:58–69
13. Hong W (2010) An efficient prediction-and-shifting embedding technique for high quality reversible data hiding. EURASIP J Adv Signal Process. article ID 104835
14. Hong W (2012) Adaptive reversible data hiding method based on error energy control and histogram shifting. Opt Commun 285(2):101–108
15. Hong W, Chen T-S (2010) A local variance-controlled reversible data hiding method using prediction and histogram-shifting. J Syst Softw 83(12):2653–2663
16. Hong W, Chen TS, Shiu CW (2009) Reversible data hiding for high quality images using modification of prediction errors. J Syst Softw 82(11):1833–1842
17. C.W. Honsinger, P.W. Jones, M. Rabbani, J.C. Stoffel (2001) Lossless recovery of an original image containing embedded data, US Patent No 6,278,791

18. Kamran A, Khan SAM (2014) A high capacity reversible watermarking approach for authenticating images: exploiting down-sampling, histogram processing, and block selection. Inf Sci 256:162–183
19. Kamstra L, Heijmans HJAM (2005) Reversible data embedding into images using wavelet techniques and sorting. IEEE Trans Image Process 14(12):2082–2090
20. Khan A, Siddiqa A, Munib S, Malik SA (2014) A recent survey of reversible watermarking techniques. Inf Sci 279(20):251–272
21. Lee SK, Suh YH, Ho YS (2006) Reversible image authentication based on watermarking. Proc IEEE ICME :1321–1324
22. Li X, Yang B, Zeng T (2011) Efficient reversible watermarking based on adaptive prediction-error expansion and pixel selection. IEEE Trans Image Process 20(12):3524–3533
23. Li X, Li B, Yang B, Zeng T (2013) General framework to histogram-shifting-based reversible data hiding. IEEE Trans Image Process 22(6):2181–2191
24. Li X, Li J, Li B, Yang B (2013) High-fidelity reversible data hiding scheme based on pixel-value-ordering and prediction-error expansion. Signal Process 93:198–205
25. Li X, Zhang W, Gui X, Yang B (2013) A novel reversible data hiding scheme based on two-dimensional difference-histogram modification. IEEE Trans Inform Forensics Sec 8(7):1091–1100
26. Li X, Zhang W, Gui X, Yang B (2015) Efficient reversible data hiding based on multiple histograms modification. IEEE Trans Info Forensics Sec 10(9):2016–2027
27. Ni Z, Shi Y, Ansari N, Su W (2006) Reversible data hiding. IEEE Trans Circ Syst Video Technol 16(3): 354–362
28. Ou B, Li X, Zhao Y, Ni R, Shi Y-Q (2013) Pairwise prediction-error expansion for efficient reversible data hiding. IEEE Trans Image Process 22(12):5010–5021
29. Pei Q, Wang X, Li Y, Li H (2013) Adaptive reversible watermarking with improved embedding capacity. J Syst Softw 86:2841–2848
30. Peng F, Li X, Yang B (2014) Improved PVO-based reversible data hiding. Digit Signal Process 25:255–265
31. Qin C, Chang CC, Huang YH, Liao L-T (2013) An inpainting-assisted reversible steganogrephic scheme using a histogram shifting mechanism. IEEE Trans Circ Syst Video Technol 23(7):1109–1118
32. Sachnev V, Kim HJ, Nam J, Suresh S, Shi YQ (2009) Reversible watermarking algorithm using sorting and prediction. IEEE Trans Circ Syst Video Technol 19(7):989–999
33. Thodi DM, Rodriguez JJ (2007) Expansion embedding techniques for reversible watermarking. IEEE Trans Image Process 16(3):721–730
34. Thodi DM, Rodriguez JJ (2007) Expansion embedding techniques for reversible watermarking. IEEE Trans Image Process 16(3):721–729
35. Tian J (2003) Reversible data embedding using a difference expansion. IEEE Trans Circ Syst Video Technol 13(8):890–896
36. Wang X, Li X, Yang B, Guo Z (2010) Efficient generalized integer transform for reversible watermarking. IEEE Signal Process Lett 17(6):567–570
37. Wang X, Ding J, Pei Q (2015) A novel reversible image data hiding scheme based on pixel value ordering and dynamic pixel block partition. Information Sci 310:16–35
38. Weng S, Zhao Y, Pan J-S, Ni R (2008) Reversible watermarking based on invariability and adjustment on pixel pairs. IEEE Signal Process Lett 15:721–724
39. Weng S, Pan J-s, Li L (2016) Reversible data hiding based on an adaptive pixel-embedding strategy and two-layer embedding. Inf Sci 369:144–159
40. Weng S, Liu Y, Pan J-S, Cai N (2016) Reversible data hiding based on flexible block-partition and adaptive block-modification strategy. J Vis Commun Image Represent 41:185–199
41. Wu X, Memon N (1997) Context-based, adaptive, lossless image coding. IEEE Trans Commun 45(4):437–444
42. Wu M, Yu H, Liu B (2003) Data hiding in image and video: part II-designs and applications. IEEE Trans Image Process 12(6):696–705
43. Zhang W, Hu X, Li X, Yu N (2013) Recursive histogram modification: establishing equivalency between reversible data hiding and lossless data compression. IEEE Trans Image Process 22(7):2775–2785

Wengui Su received the B.S. degree in mechanical and electronic engineering and the M.S. degree in mechanical manufacturing and automation from Guangxi University, Nanning, Guangxi Province, China, in 1997 and 2000, respectively. She is currently an associate professor at Guangxi University, Nanning, Guangxi Province, China. Her research interests focus on digital contents protection.

Xiang Wang received his B.S., M.E. degrees from Shandong Univ. in 2004 and 2007, and Ph.D. degrees from Peking Univ. in 2011. He is associate professor in the School of Telecommunications Engineering, Xidian University. His research interests focus on digital contents protection and visual cryptography.

Fu Li received his B.S. degree in Industrial Engineering from the Henan University of Engineering, China in 2016, and he is currently a Master Degree Candidate at Guangxi University. His research interests include Industrial engineering, logistics, and Informatization.

Yulong Shen received the B.E., M.E., and Ph.D. degree in Computer Science from Xidian University, China, in 2002, 2005, and 2008, respectively. Since 2005, he has been employed by the Key Laboratory of Computer Network and Information Security, Ministry of Education, China. Dr. shen is currently a Professor at the School of Computer and Science, Xidian University. Dr. Shen has visited the Wayne State University, USA, Future University-Hakodate University, Japan, Feng Chia University, Asia University, Taiwan, City University of Hong Kong. Dr. Shen mainly focuses on wireless sensor networks, network and information security, Internet of Things, Cloud computing. Dr. Shen has over 50 scientific publications in reputable journals, academic books and international conferences, over 30 technical invention patents. Dr. Shen is a member of the IEEE, professional membership of the ACM, senior member of CCF, council member of Shaanxi Province CCF, secretary-general of Network and Information Security Technique Committee of Shaanxi Province CCF, senior member of CIE, member of CACR, committee of CCF TCSN, and committee of CCF TCSC. Dr. Shen is editorial Board Member of AISS, TPC member of IEEE ICEBE, PC member of INCoS, Chair of Special Session on Internet Of Things, IEEE CIS, PC member of SOWN, PC member of CWSN.

Qingqi Pei received B.Sc., M.Sc. and Ph.D. degrees from Xidian University, Xi'an, China, in 1998, 2004 and 2008, respectively. In 2011, he was supported by the program for New Century Excellent Talents in University of China by MOE. Since 1998, he has worked in Xidian University. He is now a Professor and Ph.D. student supervisor in Xidian University. He is a member of IEEE and ACM, senior member of Chinese Institute of Electronics and senior member of China Computer Federation. His research interests mainly focus on wireless communication networks & security, and information security. He has published over 90 papers in the significant international journals or conferences, and granted 47 patents. He is currently serving on the editorial board of China Communications, International Journal of Advancements in Computing Technology and International Journal of Distributed Sensor Networks.