Journal of Computers Vol. 34 No. 6, December 2023, pp. 15-29
doi: 10.53106/199115992023123406002

Dynamic Hybrid Reversible Data Hiding Based on Pixel-value-ordering

Fang Ren[1,2], Yi-Ping Yang[1,2*], Zhe-Lin Zhang[1]

[1] School of Cyberspace Security, Xi'an University of Posts and Telecommunications, Xi'an, China
renfang_81@163.com, 1214801226@qq.com, zzl080805@163.com

[2] National Engineering Laboratory for Wireless Network Security Technology,
Xi'an University of Posts and Telecommunications, Xi'an, China

Received 12 December 2022; Revised 10 April 2023; Accepted 3 June 2023

Abstract. Reversible data hiding (RDH) using pixel-value-ordering (PVO) is a well-established technique for embedding data in a cover image by modifying the maximum and minimum in each block. This paper proposes a dynamic hybrid RDH method based on PVO. Specifically, a 3×3 block according to its complexity and two thresholds T1 and T2 is classified as three levels: extremely smooth, smooth, and rough. Different processing algorithms are used for different levels. For rough blocks, they are ignored to avoid reducing the peak signal-to-noise ratio. For smooth blocks, the proposed method employs a block subdivision algorithm that can embed up to 6 bits of data. For extremely smooth blocks, no subdivision is done and a median pixel prediction algorithm is used to predict the remaining eight pixels, which can embed up to 8 bits of data. Moreover, this paper presents a new method that computes complexity by dynamically selecting relevant pixels to enhance block classification accuracy. Extensive experiments demonstrate that the proposed method outperforms existing PVO-based methods, offering larger embedding capacity while maintaining low distortion.

Keywords: reversible data hiding, pixel-value-ordering, complexity, embedding capacity

1 Introduction

Reversible Data Hiding (RDH) techniques have emerged as a popular means of securely embedding sensitive data into cover images, while maintaining the ability to completely restore the original image after the extraction of the hidden information [1]. The versatility of this technique has led to its broad application across a variety of domains including medical image processing [2-4], military image processing, legal forensics, and other fields that require reliable transmission of data with minimal distortion [5-7].

Early RDH methods [8-10] relied on lossless compression of a suitable bit-plane of the cover image for embedding, where the embedding capacity (EC) was highly dependent on the compression algorithm. Later, Tian et al. [11] proposed the difference expansion (DE) method, which used the integer wavelet transform to calculate the pixel difference and expand the difference for embedding. The prediction error expansion (PEE) method, which replaced pixel difference with prediction error in embedding, was introduced in [12] and greatly improved DE's EC. PEE has since been extensively researched and used in several valuable method [13-15].

Unlike the predictor used in previous PEE-based methods, the pixel value ordering (PVO) method was proposed by Li et al. [16], which breaks through the limitation of linear space by selecting the nearest adjacent pixel of a predicted pixel for prediction. PVO first divides the image into non-overlapping blocks, then sorts the pixels in each block and calculates the prediction error as the difference between the maximum (minimum) pixel and the second maximum (minimum) pixel. Then, the maximum and minimum in each block are modified to embed data when the prediction error was 1. PVO produced superior image quality performance, but it was only suitable for low EC and could not adequately utilize a smooth block with the maximum (minimum) pixel equal to the second maximum (minimum) pixel.

To address this limitation, Peng et al. [17] proposed the improved pixel value ordering (IPVO) method. IPVO calculated the prediction error by considering the spatial order of the top two maximum (minimum) pixels and used both prediction errors of 0 and 1 to embed data. This method enhanced block utilization and improved PVO's EC. PVO-K was later proposed by Ou et al. [18], which embedded 1 bit of data by simultaneously modifying *k* maximum (minimum) pixels. Multi-pass PVO was introduced by He et al. [19] to improve PVO-K's EC, where *k* maximum (*k* minimum) pixels were taken independently to carry data. Qu et al. [20] then proposed

* Corresponding Author

the pixel-based pixel value ordering (PPVO) predictor, which treated each pixel as the embedding unit, sorted its neighboring pixels, and achieved a higher EC. Ou et al. [21] proposed the pairwise prediction error expansion method and obtained significant embedding results. Weng et al. [22] proposed a dynamic hybrid reversible data hiding based on block classification and achieved high EC. Recently, He et al. [23] proposed a flexible spatial location (FSL) based on PVO, which defines eight modes of the spatial location of pixels within a block and selects the optimal mode for each block by counting the number of inversions.

Although the above methods have achieved relatively high embedding performance, they still have certain drawbacks. Table 1 shows the advantages and limitations of the above PVO-based RDH methods.

Table 1. Advantages and limitations of PVO-based RDH methods

Method	Technique	Advantages	Limitations
Li et al. [16]	Nearest value pixel prediction.	Provides superior performance.	Limited EC.
Peng et al. [17]	The calculation takes into account the spatial position order of pixels.	Enhances block utilization and provides higher EC than [16].	Not effective for highly textured images.
Ou et al. [18]	Simultaneously modifying *k* max/min pixels.	Efficiently enhances block utilization.	Leads to higher distortion as *k* pixels are modified for embedding 1 bit of data.
He et al. [19]	Multi-pass embedding.	Achieves high EC.	Requires complex computations as multiple passes of embedding.
Qu et al. [20]	Each pixel as an embedded unit.	Results in higher EC than other methods.	Lacks of flexibility because context pixels are only taken from the right and below the current pixel.
Ou et al. [21]	Pairwise prediction errors.	Achieves better performance.	Less effective for low-quality images.
Weng et al. [22]	Block classification.	Achieves more accurate smoothness classification and higher EC.	Exhaustive search for the optimal threshold increases the computational complexity.
He et al. [23]	Spatial location modes and inverse number calculation.	Achieves higher EC and lower distortion rate.	The preprocessing step increases the complexity and time of the calculation.

The purpose of this paper is to improve the embedding performance of RDH methods based on PVO by proposing a hybrid PVO method utilizing a block subdivision technique. The key achievements of this paper can be summarized as follows.

1) The texture complexity of image blocks is classified into three groups (rough, smooth, and extremely smooth) to optimize the processing efficiency of different types of blocks.

2) A novel hybrid algorithm is proposed to efficiently handle different block types. For smooth blocks, a block subdivision technique is applied. For extremely smooth blocks, a median pixel prediction algorithm is used. For rough blocks, which is skipped. This hybrid method has resulted in a significant improvement in embedding performance.

3) A new dynamic algorithm for computing the complexity of the 3×3 block is introduced, which is more flexible than previous techniques based on the adjacent pixels of each predicted pixel in the block.

The rest of the paper is organized as follows. Peng et al.'s IPVO method is briefly reviewed in Section 2. The proposed method is presented in Section 3. An evaluation of the proposed method's performance in comparison to state-of-the-art works in Section 4. Finally, concluding remarks are provided in Section 5.

2 Related Work

In this section, we will briefly describe the IPVO method [17]. IPVO first divides the cover image into disjointed n-sized blocks, then n pixels in each block are sorted in ascending order to obtain a sorted pixel list: $\{p_{\sigma(1)}, p_{\sigma(2)}, \ldots, p_{\sigma(n)}\}$ with $p_{\sigma(1)} \leq p_{\sigma(2)} \leq \ldots \leq p_{\sigma(n)}$, where σ is a sorting related function, $\sigma(w) < \sigma(r)$ if $p_{\sigma(w)} = p_{\sigma(r)}$ and $w<r$. Then, the minimum pixel $p_{\sigma(1)}$ and the maximum pixel $p_{\sigma(n)}$ are selected as the predicted pixels, the second minimum pixel $p_{\sigma(2)}$ and the second maximum pixel $p_{\sigma(n-1)}$ are selected as the prediction pixels. This means, for each block, two

prediction errors e_1 and e_2 are computed respectively as follows:

$$e_i = p_u - p_v \ , \tag{1}$$

when we compute e_1, $u = \min\{\sigma(1), \sigma(2)\}$, $v = \max\{\sigma(1), \sigma(2)\}$; when we compute e_2, $u = \min\{\sigma(n-1), \sigma(n)\}$, $v = \max\{\sigma(n-1), \sigma(n)\}$. Notice that, e_1 means the case where the predicted pixel $p_{\sigma(1)}$ is smaller than or equal to the prediction pixel $p_{\sigma(2)}$; and e_2 means the case where the predicted pixel $p_{\sigma(n)}$ is larger than or equal to the prediction pixel $p_{\sigma(n-1)}$.

The prediction error equal to 0 and 1 are embedded data, while others are shifted for ensuring the reversibility. The prediction error is modified as follows:

$$e_i' = \begin{cases} e_i - b & if \quad e_i = 0 \ , \\ e_i + b & if \quad e_i = 1 \ , \\ e_i - 1 & if \quad e_i < 0 \ , \\ e_i + 1 & if \quad e_i > 1 \ , \end{cases} \tag{2}$$

where $b \in \{0, 1\}$ is a to-be-embedded data. IPVO modifies the minimum pixel to be smaller or unchanged and the maximum pixel to be larger or unchanged in order to ensure reversibility. Therefore, predicted pixels are modified in two cases.

Case 1: when embed b into e_1, $p_{\sigma(1)}$ is modified as follows:

$$p_{\min}' = \begin{cases} p_{\min} - b & if \quad e_i = 1 \quad or \quad e_i = 0 \ , \\ p_{\min} - 1 & if \quad e_i > 1 \quad or \quad e_i < 0 \ , \end{cases} \tag{3}$$

where $p_{\min} = p_{\sigma(1)}$, $p'_{\min} = p'_{\sigma(1)}$.

Case 2: when embed b into e_2, $p_{\sigma(n)}$ is modified as follows:

$$p_{\max}' = \begin{cases} p_{\max} + b & if \quad e_i = 1 \quad or \quad e_i = 0 \ , \\ p_{\max} + 1 & if \quad e_i > 1 \quad or \quad e_i < 0 \ , \end{cases} \tag{4}$$

where $p_{\max} = p_{\sigma(n)}$, $p'_{\max} = p'_{\sigma(n)}$.

Since the minimum is either unchanged or decreased by 1 in value and the maximum is either unchanged or increased by 1 in value after embedding, which guarantees the marked pixel ordering remains the same with the original pixel ordering and the data can be accurately exact and image can be completely restored.

In extraction, the marked error e_1' and e_2' are computed respectively as follows:

$$e_i' = p_u' - p_v' \ , \tag{5}$$

where u and v are the same as eq(1). Then the data b is extracted as follows:

$$b = \begin{cases} e_i' - 1 & if \quad e_i' \in \{1, 2\} \ , \\ -e_i' & if \quad e_i' \in \{0, -1\} \ , \end{cases} \tag{6}$$

when extract b from e_1', $p_{\sigma(1)}$ is recovered as follows:

$$p_{\min} = \begin{cases} p_{\min}' + e_i' - 1 & if \quad e_i' \in \{1, 2\} \ , \\ p_{\min}' - e_i' & if \quad e_i' \in \{0, -1\} \ , \\ p_{\min}' + 1 & if \quad e_i' < -1 \, or \, e_i' > 2 \ , \end{cases} \tag{7}$$

where $p_{\min} = p_{\sigma(1)}$, $p'_{\min} = p'_{\sigma(1)}$.

When extract b from e'_2, $p_{\sigma(n)}$ is recovered as follows:

$$p_{\max} = \begin{cases} p'_{\max} - e'_i + 1 & if \quad e'_i \in \{1,2\}, \\ p'_{\max} + e'_i & if \quad e'_i \in \{0,-1\}, \\ p'_{\max} - 1 & if \quad e'_i < -1 \; or \; e'_i > 2, \end{cases} \tag{8}$$

where $p_{\max} = p_{\sigma(n)}$, $p'_{\max} = p'_{\sigma(n)}$.

In the above procedure, since the minimum $p_{\sigma(1)}$ and the maximum $p_{\sigma(n)}$ are modified to embed data, an image block is embedded at most 2 bits of data. There is still some room for improvement, so this paper proposes a new RDH method based on PVO to increase the embedding rate of a block.

3 Proposed Method

In IPVO, only the maximum pixel and the minimum pixel are used as predicted pixels, resulting in pixel wastage for some block. In this paper, the image block is divided into three levels: rough, smooth, extremely smooth. For smooth blocks, we use block subdivision to divide them into two sub-blocks, respectively using different prediction strategy for prediction and embedding. For extremely smooth blocks, we do not subdivide them, treat them as a whole for prediction and embedding. For rough blocks, we do not use them for prediction and embedding. We will describe in detail the methods used for the two levels of blocks, as well as the complexity method and the specific image block grading criteria.

3.1 Block Subdivision

This paper proposes block subdivision for smooth blocks, which is explained in detail as follows.

First, we subdivide a 3×3 block in the way shown in Fig. 1. For block-A (orange part), pixels are collected line by line. For block-B (blue part), considering the correlation between adjacent pixels, i.e., the closer the distance between pixels, the greater the correlation, so pixels are collected from top to bottom and right to left.

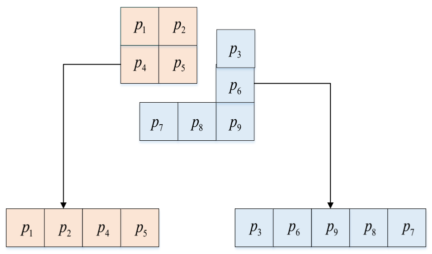


Fig. 1. Partition of a 3×3 block

In embedding, for block-A, we use the IPVO method. Suppose (p_1, p_2, p_4, p_5) has a sorted result of ($p_{\sigma(1)}$, $p_{\sigma(2)}$, $p_{\sigma(3)}$, $p_{\sigma(4)}$) the predicted pixels are $p_{\sigma(1)}$ and $p_{\sigma(4)}$, then we can compute two errors e_1 and e_2 by using eq(1), so we can embed at most 2 bits of data according to eq(2) in block-A.

For block-B, we use the median pixel continuously predict the other pixels. Suppose (p_3, p_6, p_9, p_8, p_7) has a sorted result of ($p_{\tau(1)}$, $p_{\tau(2)}$, $p_{\tau(3)}$, $p_{\tau(4)}$, $p_{\tau(5)}$),where τ has the same meaning as σ. We use the median pixel $p_{\tau(3)}$ to predict the remaining four pixels sequentially. Four predicted pixels can obtain four errors respectively by using eq(1), where $u = \min\{\tau(i), \tau(3)\}$, $v = \max\{\tau(i), \tau(3)\}$.When $i = 1$, e_3 is computed; when $i = 2$, e_4 is computed; when $i = 4$, e_5 is computed; when $i = 5$, e_6 is computed. We can embed at most 4 bits of data according to eq(2) in block-B. As in IPVO, the predicted pixels should be modified in two cases. e_3 and e_4 represent the case where the predicted pixel is smaller than or equal to the prediction pixel, therefore when embed b into e_3 and e_4 respectively, $p_{\tau(1)}$ and $p_{\tau(2)}$ will be modified by using eq(3) respectively; e_5 and e_6 represent the case where the predicted pixel is larger than or equal to the prediction pixel, therefore when embed b into e_5 and e_6 respectively, $p_{\tau(4)}$ and $p_{\tau(5)}$ will be modified by using eq(4) respectively.

In extraction, for block-A, we use the IPVO method to recover the block-A. Suppose that the pixels in marked block-A are sorted as $(p'_{\sigma(1)}, p'_{\sigma(2)}, p'_{\sigma(3)}, p'_{\sigma(4)})$, since the mapping σ keeps unchanged, reversible extraction and recovery are possible. Firstly, two marked errors are computed by using eq(5), where u and v are the same as embedding in block-A. Then, the embedded data b is extracted by using eq(6). Finally, $p_{\tau(1)}$ is recovered by using eq(7), $p_{\tau(4)}$ is recovered by using eq(8).

For block-B, the mapping τ also keeps unchanged. Suppose that the pixels in marked block-B are sorted as $(p'_{\tau(1)}, p'_{\tau(2)}, p'_{\tau(3)}, p'_{\tau(4)}, p'_{\tau(5)})$, four marked errors are computed by using eq(5), where u and v are the same as embedding in block-B. The embedded data b is extracted by using eq(6). When extracting b from e'_3 or e'_4 respectively, $p_{\tau(1)}$ and $p_{\tau(2)}$ will be recovered by using eq(7) respectively, when extracting b from e'_5 or e'_6 respectively, $p_{\tau(4)}$ and $p_{\tau(5)}$ will be recovered by using eq(8) respectively. A specific example will be shown later in Fig. 3.

3.2 Median Pixel Prediction

This paper proposes median pixel prediction for extremely smooth blocks, which is actually the prediction method used by block-B in the Section 3.1. Median pixel prediction method is to select the median pixel as the only prediction pixel and the remaining pixels as the predicted pixel in a sort pixel list. We will now detail the process of using the median pixel prediction method for an extremely smooth block.

In embedding, suppose the original image block is collected as a one-dimensional list of $(p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9)$ which has a sorted result of $(p_{r(1)}, p_{r(2)}, p_{r(3)}, p_{r(4)}, p_{r(5)}, p_{r(6)}, p_{r(7)}, p_{r(8)}, p_{r(9)})$. We use the median $p_{\tau(5)}$ to predict the remaining eight pixels sequentially to compute e_i by using eq(1), where $i=\{1, 2, 3, 4, 5, 6, 7, 8\}$. When $i = 1, 2, 3, 4$, $u = \min\{\sigma(i), \tau(5)\}$, $v = \max\{\sigma(i), \sigma(5)\}$; when $i = 5, 6, 7, 8$, $u = \min\{\sigma(i+1), \tau(5)\}$, $v = \max\{\sigma(i+1), \sigma(5)\}$. We can embed at most 8 bits of data according to eq(2). Accordingly, when i=1,2,3,4, p_i is modified by using eq(3); when i=5,6,7,8, p_{i+1} is modified by using eq(4).

In extraction, suppose that the pixels in marked list are sorted as $(p'_{r(1)}, p'_{r(2)}, p'_{r(3)}, p'_{r(4)}, p'_{r(5)}, p'_{r(6)}, p'_{r(7)}, p'_{r(8)}, p'_{r(9)})$, then eight marked errors are computed by using eq(5), where u and v are the same as embedding. The embedded data b is extracted by using eq(6). Accordingly, when $i = 1, 2, 3, 4$, p_i is recovered by using eq(7); when $i = 5, 6, 7, 8$, p_{i+1} is recovered by using eq(8). A specific example will be shown later in Fig. 4.

3.3 Block Complexity

In PVO-based methods, *NL* is used to denote the complexity of a block, the block whose *NL* is greater than the threshold is a rough block. Rough blocks are less likely to generate embeddable prediction errors, so they would lead to shifting to reduce image visual quality. To prevent distortion caused by unnecessary shifting, rough blocks are skipped.

In [17], the value of difference between the second maximum pixel and the second minimum pixel is defined as the *NL*. However, when size of the block is small, the number of pixels involved in sorting is too small, and it is difficult to reflect the changing trend of pixels. In [18], Ou et al. use $(n_1+2) \times (n_2+2)$ block's right and bottom neighboring pixels to compute *NL* for a $n_1 \times n_2$ block. However, when the block size is large, the correlation between pixels involved in calculating complexity and pixels involved in calculating prediction error is not high.

In the proposed method, we break the block limit and consider each predicted pixel separately. Eight neighboring pixels of the predicted pixel are potentially used to calculate *NL* of the current block. For example, there are six predicted pixels in a 3×3 smooth block. Similarly, these six predicted pixels are in two cases: the predicted pixel is larger than or equal to the prediction pixel and the predicted pixel is less than or equal to the prediction pixel. Therefore, we compose a set C_1 with the union of three types of pixels: 1) eight neighboring pixels of the maximum pixel in block-A; 2) eight neighboring pixels of the maximum pixel in block-B; 3) eight neighboring pixels of the second maximum pixel in block-B. We compose another set C_2 with the union of three types of pixels: 1) eight neighboring pixels of the minimum pixel in block-A; 2) eight neighboring pixels of the minimum pixel in block-B; 3) eight neighboring pixels of the second minimum pixel in block-B.

Meanwhile, all predicted pixels in the current block and its eight neighboring blocks are excluded from the set to ensure reversibility. Fig. 2 shows the sets C_1 and C_2 of a 3×3 smooth block.

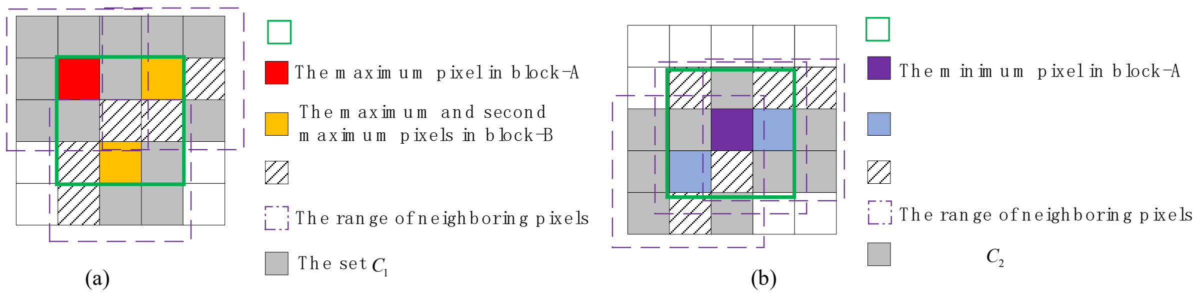

The maximum pixel in block-A
The maximum and second maximum pixels in block-B
The range of neighboring pixels
The set C_1
(a)
The minimum pixel in block-A
The range of neighboring pixels
C_2
(b)

Fig. 2. The sets C_1 and C_2 of a 3×3 smooth block

Based on the adaptive pixel selection strategy, the set of pixels involved in *NL* calculation is adaptively obtained for each predicted pixel. Then, the absolute value of the difference between the average value of the pixels in C_1 and C_2 is defined as complexity *NL*, which is calculated as follows:

$$\mathrm{NL} = \left| AVG(C_1) - AVG(C_2) \right| , \tag{9}$$

where *AVG* represents the average of all pixel values in the set.

3.4 Dynamic Partitioning

The proposed method uses two thresholds, T_1 and T_2, to classify the block into three levels, and we consider the following three cases when embedding data.

Case 1: When $NL>T_1$, the current block level is L_1. This means that the block is probably in a rough region, and it must be skipped from embedding.

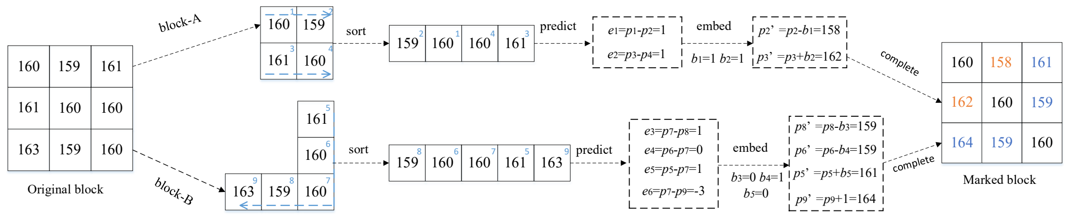
Case 2：When $T_2<NL<=T_1$, the current block level is L_2. This means that the block is probably in a smooth region. For this case, we use the block subdivision algorithm in Section 3.1 to subdivide the current block into two subblocks to embed data. An L_2-block can embed up to 6 bits of data.

Case 3: When $NL<=T_2$, the current block level is L_3. This means that the block is probably in an extremely smooth region. For this case, we use the median pixel prediction algorithm in Section 3.2. An L_3-block can embed up to 8 bits of data.

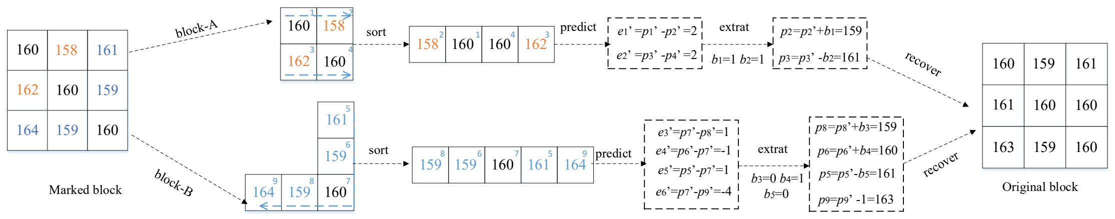
In a natural image, extremely smooth blocks are very rare, so in the above three cases, $T_1 > T_2$, T_1 is required to be as large as possible, while T_2 needs to be as small as possible. The proposed method is a more flexible method compared with previous PVO-based methods, which can make full use of image texture feature and pixels in the block, so that the efficiency of this method is better and the embedding capacity is larger.

In Fig. 3, an example of embedding and extraction in an L_2-block is presented. In Embedding, firstly, the block is divided into two subblocks, and pixels are sorted and predicted for each subblock. For block-A, there are two predicted pixels, so we can get two errors from sorted list (159, 160, 160, 161) according to eq(1), and embed 2 bits of data to block-A according to eq(2), then get marked list (158, 160, 160, 162) according to eq(3) and eq(4). For block-B, there are four predicted pixels, so we can get four errors from sorted list (159, 160, 160, 161, 163) according to eq(1), we can embed 3 bits of data into block-B according to eq(2), and get marked list (159, 159, 160, 161, 164) according to eq(3) and eq(4). Finally, all marked lists of block-A and block-B are repositioned to the corresponding positions to generate the marked block.

Extraction is the same as embedding. Firstly, marked block is divided into two subblocks in the same way. The marked lists (158, 160, 160, 162) and (159, 160, 160, 161, 163) are obtained in the same scanning order. Then, the errors are computed according to eq(5), b_1, b_2, b_3, b_4 and b_5 are extracted according to eq(6), the original pixels are recovered according to eq(7) and eq(8). Finally, all recovered pixels are updated to the marked block to obtain the original block.



(a) Embedding example for a L2- block

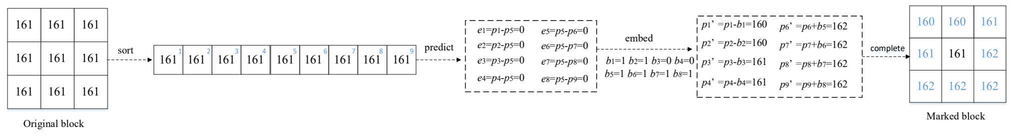


(b) Extraction example for a L2- block

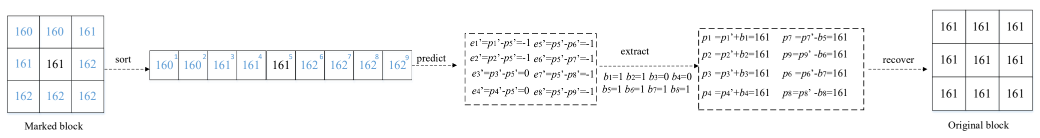
Fig. 3. An example of embedding and extraction in an L_2-block

In Fig. 4 , an example of embedding and extraction in an L_3-block is presented. In Embedding, eight pixels are sorted in ascending order to obtain a sorted list (161, 161, 161, 161, 161, 161, 161, 161, 161), the prediction pixel is the median pixel $p_{\sigma(5)}$, the predicted pixels are the other eight pixels. We can get eight errors by using eq(1). Eight bits of data can be embedded according to eq(2). Accordingly, $p_{\sigma(1)}$, $p_{\sigma(2)}$, $p_{\sigma(3)}$ and $p_{\sigma(4)}$ are modified to by using eq(3) respectively; $p_{\sigma(5)}$, $p_{\sigma(6)}$, $p_{\sigma(7)}$ and $p_{\sigma(8)}$ are modified by using eq(4) respectively. Finally, all marked pixels are repositioned to the corresponding positions to generate the marked block.

In extraction, the marked pixels are sorted to obtain a marked sorted list (160, 160, 161, 161, 161, 162, 162, 162, 162). Then, eight marked errors are computed according to eq(5), eight bits of data are extracted by using eq(6), $p_{\sigma(1)}$, $p_{\sigma(2)}$, $p_{\sigma(3)}$ and $p_{\sigma(4)}$ are recovered by using eq(7) respectively, $p_{\sigma(6)}$, $p_{\sigma(7)}$, $p_{\sigma(8)}$ and $p_{\sigma(9)}$ are recovered by using eq(8) respectively. Finally, all recovered pixels are updated to the marked block to obtain the original block.



(a) Embedding example for a L3- block



(b) Extraction example for a L3- block

Fig. 4. An example of embedding and extraction in an L_3-block

3.5 Description of the Algorithm

The embedding process is described in the following.

Step 1 (Image division and location map construction): First, all pixels except the first two rows in the cover image are divided into 3×3 non-overlapped blocks $\{X_1, X_2, ..., X_N\}$. In order to solve the overflow/underflow problem, if X_i , $i \in \{1, ..., n\}$ contains 255 or 0, that means it is unavailable for embedding and we mark it as LM(i)=1, otherwise we mark it as LM(i)=0. Then, we compress all LM into a shorter bit stream (location map) whose length is denoted as L_{clm}.

Step 2 (Secret data embedding): The complexity NL_i of each X_i is computed by using (9), we only embed data in the following two cases.

Case 1: $T_2 < NL_i <= T_1$ and LM(i)=0. In this case, X_i is an L_2-block, which is subdivided into two sub-blocks by using block subdivision technique. Then, we compute six errors and embed data by using the method in Section 3.1.

Case 2: $NL_i <= T_2$ and LM(i)=0. In this case, X_i is an L_3-block, which is not subdivided. we compute eight errors and embed data by using the method in Section 3.2.

In other cases, X_i remains unchanged. When all data is embedded, this step will stop, and we record X_{end} as the last data-carrying block.

Step 3 (Auxiliary information and location map embedding): In order to guarantee the reversibility, we must embed the auxiliary information and location map into the image. The auxiliary information includes L_{clm}, thresholds T_1 and T_2, block size and end position X_{end}.The auxiliary information and the compressed location map are embedded in the selected pixels from the first two rows by LSB replacement. The replaced LSB is compressed into a bitstream string S_{lsb}, which is embedded into the remaining blocks $\{X_{end}, X_{end+1}, ..., X_N\}$ using the same method in Step 2.

The extraction process is described in the following.

Step 1 (Auxiliary information and location map extraction): First, the LSB of the selected pixels from the first two rows is extracted to retrieve L_{clm}, T_1, T_2, block size, X_{end} and the compressed location map. Then decompress the compressed location map to get the location map LM.

Step 2 (Data extraction): In this step, the marked image except the first two rows is divided into 3×3 blocks, we skip all marked blocks marked LM(i) = 1, and calculate complexity NL_i of each block X'_i that marked LM(i) = 0. If $T_2 < NL_i <= T_1$, data is extracted by using the method in Section 3.1. If $NL_i <= T_2$, data is extracted by using the method in Section 3.2. The extracted data includes secret data and S_{LSB}.

Step 3 (Image recovery): We recover the blocks $\{X_1, X_2, ..., X_N\}$ according to the extracted secret data, then decompress S_{lsb} and recover the selected pixels from the first two rows by LSB replacement, finally recover the whole image.

4 Experimental Results

This section details the results of the proposed reversible data hiding (RDH) method based on the PVO. To assess the performance of the proposed method, a set of six grayscale images, with a size of 512×512, were selected as test images, as illustrated in Fig. 5. The USC-SIPI image database was the source of all the images, except for the Barbara image. During the embedding process, a randomly generated sequence of 0 and 1 was employed as the secret data, represented by b. These experiments aimed to evaluate the effectiveness of the proposed method in terms of embedding capacity (EC) and peak signal-to-noise ratio (PSNR) performance.

It should be noted that the proposed method subdivides a 3×3 block into block-A and block-B, where different prediction methods are used for each sub-block. In block-A, the second minimum and second maximum pixel are selected as the prediction pixel, whereas in block-B, the median pixel is selected as the prediction pixel. To measure the effectiveness of the proposed block subdivision method, we compared it with two single methods, in which block-A and block-B methods were used to predict the entire 3×3 block. In Fig. 6, the comparison of the three methods applied to the six test images is shown. As seen, the proposed method outperforms the other two single methods remarkably. Moreover, it should be noted that using the median pixel as the prediction pixel to predict the other eight pixels (the method in block-B) resulted in very low PSNR. This is due to the fact that the median pixel prediction method is only suitable for extremely smooth blocks, which are uncommon in natural images. Therefore, although the median pixel prediction method can theoretically embed up to eight bits of data in an image block, it would increase the shifted pixels of image blocks and reduce the PSNR of the image if applied to all the image blocks in the experiment.

Next, the proposed method was compared with six previously proposed RDH methods, namely, Li et al [16], Peng et al. [17], Weng et al. [22], Kim et al. [24], Ou et al. [25], and Zhang et al. [26]. Li et al.'s method is the classical PVO method, while Peng et al.'s IPVO method improved on the PVO method by introducing spatial position order for calculating prediction errors. Weng et al.'s method further improved the IPVO algorithm by dynamically selecting different pixel numbers for data embedding, depending on different levels of image blocks. Kim et al.'s method obtained a pair of extreme predicted values when calculating prediction errors, and generated two tilted asymmetric histograms with short and long tails. The tilted histograms were used to implement data embedding. Ou et al.'s method used adaptive pixel pairing and adaptive mapping to generate two-dimensional histograms for data embedding. Zhang et al.'s method designed various pairwise modifications to obtain high embedding performance. All the previous RDH methods aimed for higher EC and lower PSNR. The comparison of these six methods and the proposed method are shown in Fig. 7. The proposed method achieves a substantial improvement in PSNR on almost all test images, particularly on smooth images such as Lena, Peppers, Boat, and Elaine, where it has a much higher PSNR at the same EC compared to the other methods. This is attributed to the proposed method making full use of every pixel in the block. However, for rough images such as Baboon and Barbara, the PSNR improvement is less significant when the EC is small (e.g., less than 6000 bits on Baboon and less than 15,000 bits on Barbara), compared to other images. This is due to the fluctuation in pixel values of rough images, making it unsuitable to use too many predicted pixels at a low EC compared to other images.

Additionally, the proposed method is tested on six different images using an EC of 10,000 bits, and the PSNR results for each method on each image are summarized in Table 2. The proposed method outperforms the existing methods on most images, with only a slight difference of 0.27 dB less than Zhang et al. [26] on the Barbara image. However, the proposed method achieves the highest average PSNR gain among all methods, with average gains of 2.33 dB, 2.25 dB, 1.47 dB, 2.24 dB, 1.93 dB, and 0.95 dB for the six test images. These results demonstrate the superiority of the proposed RDH method over existing methods in terms of PSNR performance.

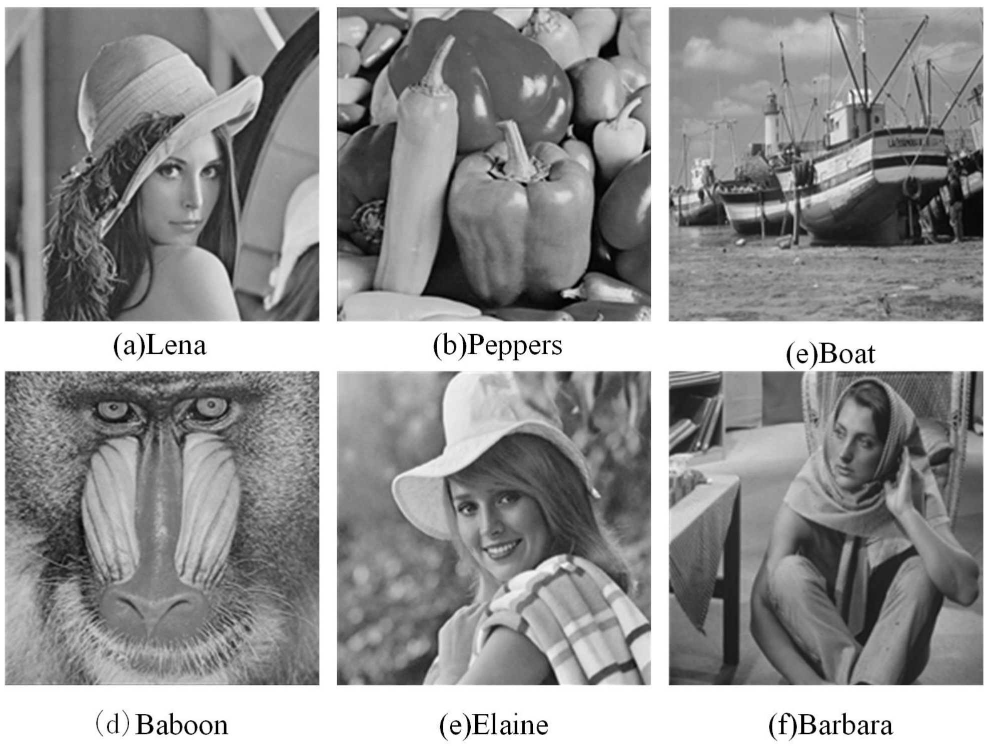


Fig. 5. Six gray test images in the experiment

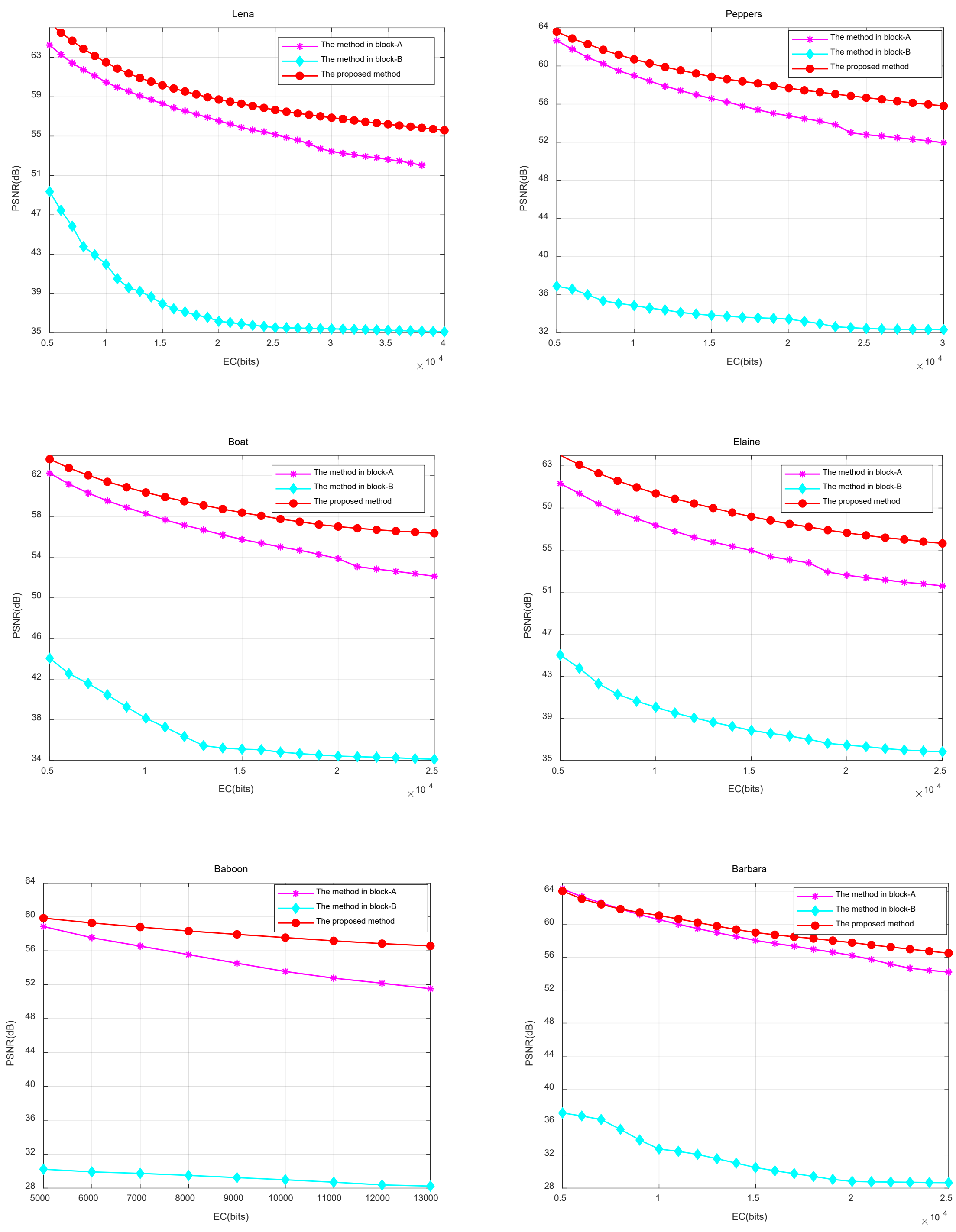


Fig. 6. Performance comparison between the method in block-A, the method in block-B and the proposed method

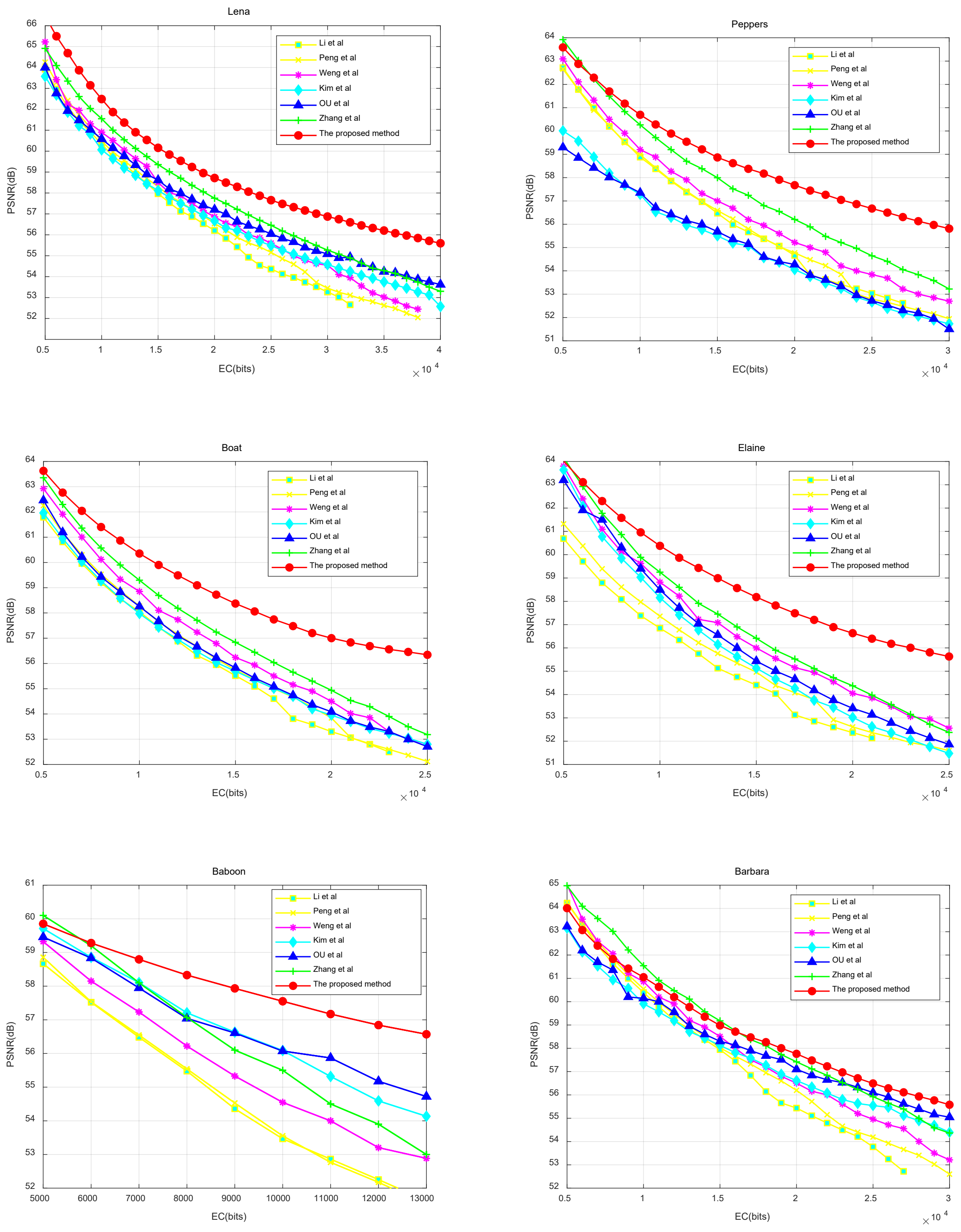


Fig. 7. Performance comparisons between the proposed method and the following six methods: Li et al. [16], Peng et al. [17], Weng et al. [22], OU et al. [24], Kim et al. [25], Zhang et al. [26]

Table 2. Comparison of PSNR between six PVO-based methods and the proposed method, for an EC of 10000 bits

	Li et al.	Peng et al.	Weng et al.	Kim et al.	Ou et al.	Zhang et al.	The proposed method
Lena	60.49	60.52	60.95	59.92	60.59	61.53	62.46
Peppers	58.99	59.01	59.11	56.93	57.05	59.97	60.92
Boat	58.12	58.20	58.92	57.67	58.28	59.32	60.21
Elaine	56.99	57.26	58.89	58.33	58.10	59.18	60.13
Baboon	53.32	53.36	54.59	55.99	56.05	55.19	57.41
Barbara	60.32	60.37	60.92	59.95	60.54	61.35	61.08
Average	58.04	58.12	58.90	58.13	58.44	59.42	60.37

The maximum EC of each method is presented in Fig. 8. It is a well-known fact that smaller block sizes lead to larger EC values. In this paper, our proposed method employs 3×3 blocks, which is larger than the 2×2 blocks used in other methods. Despite the larger block size, our method still achieves the highest maximum EC among all the compared methods. This is attributed to the flexibility of our method, allowing up to 8 bits of data to be embedded in extremely smooth blocks and up to 6 bits in smooth blocks. In contrast, most other methods are limited to embedding only up to 2 bits of data. These results suggest the effectiveness of our proposed in the embedding capacity.

To further evaluate the performance of our proposed method, we conducted experiments on a set of twelve color images of size 512×512 and twelve gray images from the USC-SIPI Image Dataset, as shown in Fig. 9 and Fig. 10, respectively. We first converted the color images to gray images for consistency. We varied the EC from 10,000 bits to 20,000 bits and measured the peak signal-to-noise ratio (PSNR) of each image using our proposed method. Table 3 summarizes the results. The results demonstrate that our proposed method achieves superior PSNR values compared to the other methods for all tested images. The average PSNR values of the color images are 64.66 dB, 63.00 dB, and 61.60 dB for EC values of 10,000 bits, 15,000 bits, and 20,000 bits, respectively. Similarly, the average PSNR values of the gray images are 61.93 dB, 60.37 dB, and 59.42 dB for the same EC values. These results indicate that our proposed method is stable and consistently outperforms the other methods, even when varying the EC.

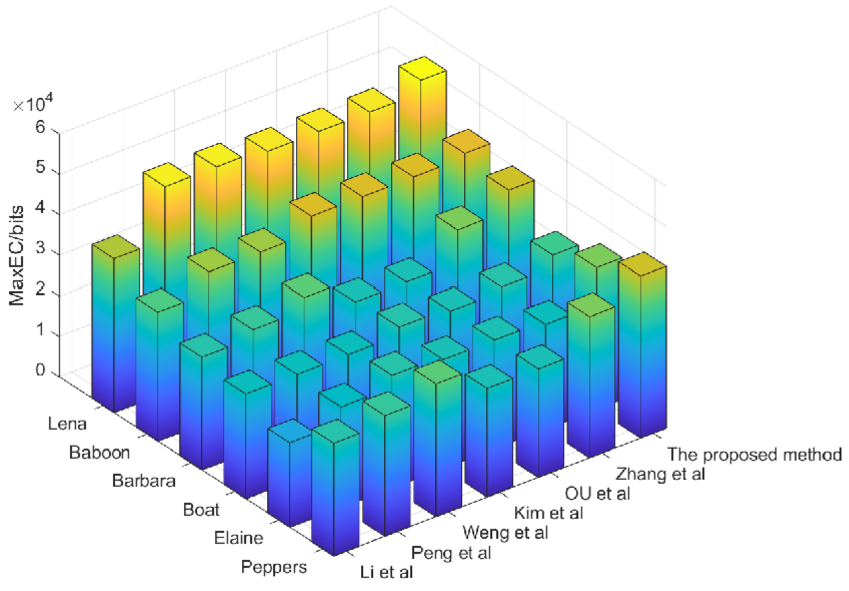


Fig. 8. Comparison of maximum EC between the proposed method and the following six methods: Li et al. [16], Peng et al. [17], Weng et al. [22], OU et al. [24], Kim et al. [25], Zhang et al. [26]

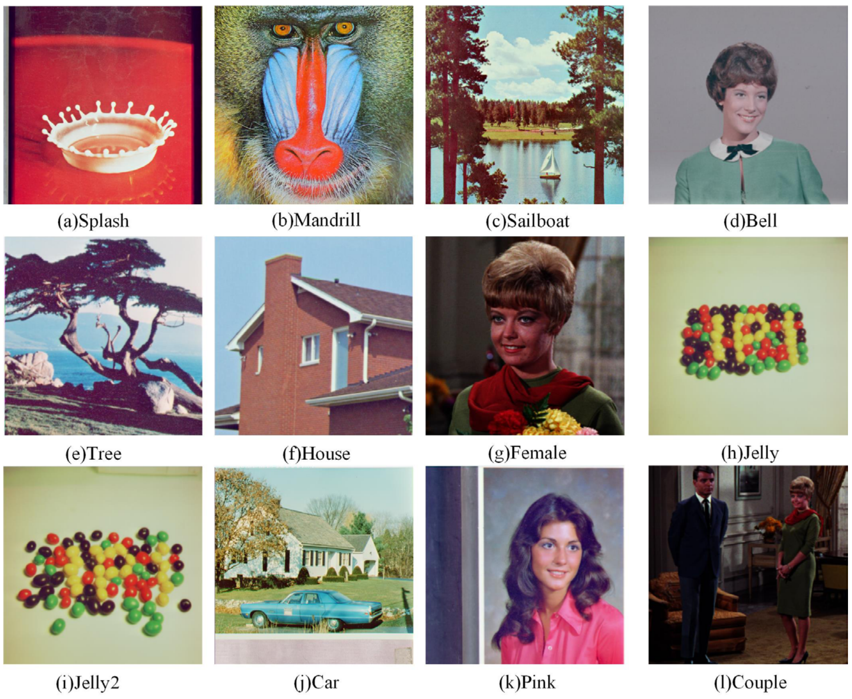


Fig. 9. Twelve color test images in the experiment

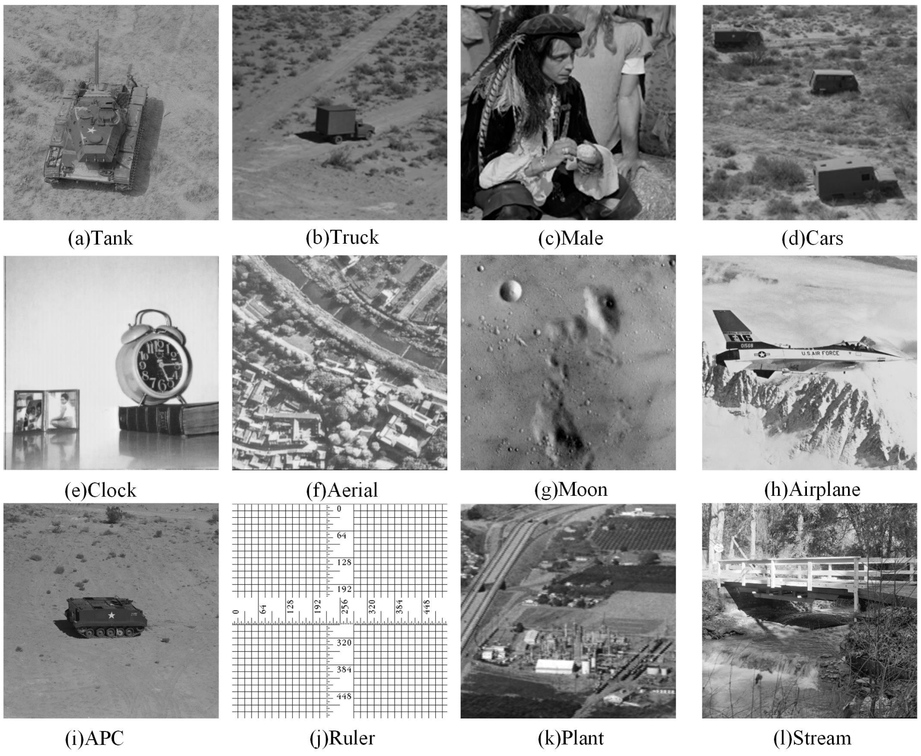


Fig. 10. Twelve gray test images in the experiment

Table 3. PSNR of the proposed method for EC of 10000 bits,15000bits, 20000bits on 12 color images and 12 gray images

Color images	10000bits	15000bits	20000bits	**Gray images**	10000bits	15000bits	20000bits
Splash	61.95	60.36	59.23	Tank	59.96	58.14	56.98
Mandrill	57.55	56.04	55.26	Truck	60.39	59.11	57.93
Sailboat	60.82	59.31	57.97	Male	60.48	58.97	58.07
Bell	66.93	65.32	64.18	**Cars**	62.82	61.29	59.97
Tree	65.45	6261	60.30	Clock	66.01	64.59	63.54
House	66.05	64.23	62.02	Aerial	60.86	59.47	59.02
Female	64.22	62.60	61.43	Moon	62.27	60.49	59.21
Jelly	67.84	66.20	65.02	Airplane	62.48	61.05	60.11
Jelly2	67.58	66.02	64.97	APC	60.28	58.74	57.82
Car	66.94	66.29	64.14	Ruler	66.53	64.71	64.48
Pink	66.19	63.98	62.58	Plant	61.16	59.24	58.55
Couple	64.46	63.06	62.11	Stream	59.93	58.69	57.35
Average	64.66	63.00	61.60	Average	61.93	60.37	59.42

5 Conclusions

In this paper, we present a novel reversible data hiding (RDH) method based on pixel-value-ordering (PVO) that has high embedding capacity and low distortion. The proposed method employs a classification technique that divides a 3×3 block into three categories: extremely smooth, smooth, and rough. The treatment methods applied to each block type are different. For example, a block subdivision algorithm is proposed for smooth blocks, while a median pixel prediction algorithm is applied to extremely smooth blocks. The rough blocks are left untreated. Our proposed method effectively exploits the correlation between image pixels and reduces the number of invalid shifted pixels. Moreover, we propose a dynamic pixel selection method that accurately measures the complexity of an image block. Our proposed method has demonstrated superior performance compared to existing methods in both the maximum embedding capacity (EC) and peak signal-to-noise ratio (PSNR). However, there are limitations to our work. For instance, the proposed method lacks flexibility in terms of image block size. To address this limitation, we plan to explore the design of embedding and extraction methods for 5×5 image blocks in the future.

6 Acknowledgement

This paper was supported by the National Natural Science Foundation of China (No. 62202377), the Natural Science Basic Research Plan of Shaanxi Province of China (No. 2021JM-463), and Postgraduate Innovation Fund project of Xi'an University of Posts and Telecommunications (CXJJDL2022015).

References

[1] S. Kumar, A. Gupta, G.S. Walia, Reversible data hiding: A contemporary survey of state-of-the-art, opportunities and challenges, Applied Intelligence 52(7)(2022) 7373-7406.

[2] R. Karmakar, A. Basu, Implementation of a Reversible Watermarking Technique for Medical Images, in: Research Anthology on Improving Medical Imaging Techniques for Analysis and Intervention, IGI Global, 2023 (pp. 490-526).

[3] V.M. Manikandan, K.S.R. Murthy, B. Siddineni, N. Victor, P.K.R. Maddikunta, S. Hakak, A High-Capacity Reversible Data-Hiding Scheme for Medical Image Transmission Using Modified Elias Gamma Encoding, Electronics 11(19) (2022) 3101.

[4] G. Gao, S. Tong, Z. Xia, B. Wu, L. Xu, Z. Zhao, Reversible data hiding with automatic contrast enhancement for medical images, Signal Processing 178(2021) 107817.

[5] L. Singh, A.K. Singh, P.K. Singh, Secure data hiding techniques: a survey, Multimedia Tools and Applications 79(23-24)(2020) 15901-15921.
[6] K.K. Manjunath, R.S. Kunte, A Robust Reversible Data Hiding Framework for Video Steganography Applications, International Journal of Advanced Computer Science and Applications 13(3)(2022) 430-440.
[7] K.S. Rekha, M.J. Amali, M. Swathy, M. Raghini, B.P. Darshini, A steganography embedding method based on CDF-DWT technique for data hiding application using Elgamal algorithm, Biomedical Signal Processing and Control 80(2023) 104212.
[8] M.U. Celik, G. Sharma, A.M. Tekalp, E. Saber, Lossless generalized-LSB data embedding, IEEE transactions on image processing 14(2)(2005) 253-266.
[9] Y.Q. Shi, X. Li, X. Zhang, H.T. Qu, M. Ma, Reversible data hiding: advances in the past two decades, IEEE access 4(2016) 3210-3237.
[10] X. Yin, S. Wu, B. Chen, K. Wang, W. Liu, Reversible data hiding in JPEG document images based on zero coefficients embedding, Signal Processing 206(2023) 108917.
[11] J. Tian, Reversible data embedding using a difference expansion, IEEE transactions on circuits and systems for video technology 13(8)(2003) 890-896.
[12] D.M. Thodi, J.J. Rodríguez, Expansion embedding techniques for reversible watermarking, IEEE transactions on image processing 16(3)(2007) 721-730.
[13] W. He, G. Xiong, Y. Wang, Reversible Data Hiding Based on Multiple Pairwise PEE and Two-Layer Embedding, Security and Communication Networks 2022(2022) 2051058.
[14] G. Kaur, S. Singh, R. Rani, R. Kumar, A. Malik, High-quality reversible data hiding scheme using sorting and enhanced pairwise PEE, IET Image Processing 16(4)(2022) 1096-1110.
[15] N. Mao, F. Chen, H. He, Y. Yang, Reversible data hiding based on adaptive IPVO and two-segment pairwise PEE, Signal Processing 198(2022) 108577.
[16] X. Li, J. Li, B. Li, B. Yang, High-fidelity reversible data hiding scheme based on pixel-value-ordering and prediction-error expansion, Signal processing 93(1)(2013) 198-205.
[17] F. Peng, X. Li, B. Yang, Improved PVO-based reversible data hiding, Digital Signal Processing 25(2014) 255-265.
[18] B. Ou, X. Li, Y. Zhao, R. Ni, Reversible data hiding using invariant pixel-value-ordering and prediction-error expansion, Signal processing: image communication 29(7)(2014) 760-772.
[19] W. He, K. Zhou, J. Cai, L. Wang, G. Xiong, Reversible data hiding using multi-pass pixel value ordering and prediction-error expansion, Journal of Visual Communication and Image Representation 49(2017) 351-360.
[20] X. Qu, H.J. Kim, Pixel-based pixel value ordering predictor for high-fidelity reversible data hiding, Signal Processing 111(2015) 249-260.
[21] B. Ou, X. Li, J. Wang, High-fidelity reversible data hiding based on pixel-value-ordering and pairwise prediction-error expansion, Journal of Visual Communication and Image Representation 39(2016) 12-13.
[22] S. Weng, Y.Q. Shi, W. Hong, Y. Yao, Dynamic improved pixel value ordering reversible data hiding, Information Sciences 489(2019) 136-154.
[23] W. He, Z. Cai, Y. Wang, Flexible spatial location-based PVO predictor for high-fidelity reversible data hiding, Information Sciences 520(2020) 431-444.
[24] S. Kim, X. Qu, V. Sachnev, H.J. Kim, Skewed histogram shifting for reversible data hiding using a pair of extreme predictions, IEEE Transactions on Circuits and Systems for Video Technology 29(11)(2019) 3236-3246.
[25] B. Ou, X. Li, W. Zhang, Y. Zhao, Improving pairwise PEE via hybrid-dimensional histogram generation and adaptive mapping selection, IEEE Transactions on Circuits and Systems for Video Technology 29(7)(2019) 2176-2190.
[26] T. Zhang, X. Li, W. Qi, Z. Guo, Location-based PVO and adaptive pairwise modification for efficient reversible data hiding, IEEE Transactions on Information Forensics and Security 15(2020) 2306-2319.