Journal of Information Hiding and Multimedia Signal Processing ©2024 ISSN 2073-4212
Ubiquitous International Volume **15**, Number **3**, September **2024**

A dual-image reversible data hiding based on PVO

Thai Pham Minh

Faculty of Information Technology
University of Economics - Technology for Industries
456 Minh Khai Street, Hanoi, Vietnam.
pmthai@uneti.edu.vn

Hoa Le Quang

School of Information and Communications Technology
Hanoi University of Science and Technology
1 Dai Co Viet Street, Hanoi, Vietnam.
hoa.lequang1@hust.edu.vn

At Pham Van

Faculty of Information Technology
University of Economics - Technology for Industries
456 Minh Khai Street, Hanoi, Vietnam.
phamvanat83@gmail.com

Luyen Cao Thi

Faculty of Information Technology
University of Transport and Communications
3 Cau Giay, Hanoi, Vietnam.
luyenct@utc.edu.vn

Trung Mai Manh

Faculty of Information Technology
University of Economics - Technology for Industries
456 Minh Khai Street, Hanoi, Vietnam.
mmtrung@uneti.edu.vn

Thu Thuy Trieu

Department of Informatics, Institute of Information Technology
Vietnam Academy of Science and Technology
Vietnam.
thuytrieu@ioit.ac.vn

*Corresponding author: Hoa Le Quang

Received May 2024; revised June 2024; accepted July 2024

Abstract. *As known, the reversible data hiding methods based on pixel value ordering (PVO) have a not large embedding capacity. To overcome this weakness, Niu et al. proposed a method that uses PVO to embed data in two images. This method can always embed three or four bits in each block. However, their method only embeds data in the largest and smallest pixels, so its embedding capacity is limited. In addition, each pixel is modified at most by two units, so stego image quality is not high. In this paper, we propose a novel PVO-based reversible data hiding method in two images, where embedding is performed simultaneously at the second-largest and largest pixels as well as at the second-smallest and smallest pixels. Additionally, each pixel is only modified at most by one unit, so the proposed method outperforms Niu et al.'s method in terms of both embedding capability and stego image quality. Furthermore, experimental results and theoretical analysis show that the $PSNR$ value of the proposed method is higher than that of some existing techniques embedding in two images.*

Keywords: Reversible data hiding, PVO, Dual-image, Largest pixel, Smallest pixel

1. Introduction.

In contemporary times, the proliferation of communication technology has facilitated the widespread dissemination of digital documents over the Internet and several other networks. However, the network is an unsafe environment where hackers can attack to modify the content of documents or use them illegally. That's why researching ways to protect documents becomes so urgent. Data hiding and encryption are the two leading solutions to tackle this problem. Encryption methods involve transforming original documents into cryptographic documents with symbols that lack meaning, making them suspicious for hackers. Data hiding is a method of incorporating data bits into a digital image in a way that makes it impossible for the unaided eye to differentiate between the original image and the stego image (which contains the data), thereby making it inconspicuous to hackers.

Data hiding methods are divided into irreversible and reversible (or lossless). The irreversible methods [1, 2, 3, 4, 5] allow the extraction of the embedded data bits correctly but cannot recover the original image. This constraint hinders its capacity to be applied in situations where the original image is required, such as in the processing of medical and military images. Hence, there has been a recent proposal and development of reversible data hiding (RDH) techniques. During the initial stages, certain techniques employed lossless compression of the lower bits of the image in order to acquire the necessary space for inserting the data bits. However, these methods exhibited a limited capacity for embedding.

The difference expansion (DE) method proposed by Tian [6] is the first notable RDH method. This method embeds a bit into a pair of adjacent pixels by expanding their difference to the left. To extract data and restore images, need to use a binary location map to distinguish embeddable and non-embeddable pairs. This method quickly received the attention of many researchers to expand the capability for embedding and the quality of stego images such as methods [7, 8] using pixel blocks instead of pixel pairs, methods [9, 10] using the difference between two prediction errors instead of the difference between two pixels, the methods [11, 12] suggesting ways in order to decrease the dimensions of the location map.

The histogram shift (HS) method of Ni et al. [13] can be considered the second most important research direction of the RDH method after Tian's DE method. In this method, first, a pixel histogram is established, which is a function of the pixel values. The bits of data are embedded in the pixels corresponding to the largest values of the histogram. The stego image quality of this method is good; however, its embedding capacity is limited due to the histograms of natural images often being flat. Many authors, such as [14, 15, 16] have improved the HS method to extend the embedding capacity.

Works [17, 18, 19] exploit smooth regions in images because the embedding performance in these regions is often higher than in the other areas.

Li et al. [20] introduced a novel prediction predictor using pixel value ordering (PVO) and RDH method based on this predictor. The proposed methodology entails dividing the image into separate blocks that are non-overlapping. Each block's pixels are arranged in increasing order. The forecast for the largest pixel is obtained from the pixel that is the second largest. A bit is then embedded when the prediction error reaches a value of 1. The process of embedding smallest values is carried out in a similar manner. Nevertheless, the approach's embedding ability is limited, prompting numerous studies [21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32] to propose strategies for enhancing the embedding capacity of the PVO method.

In general, RDH methods in one image do not have a high embedding rate, so recently, RDH methods in two or more images, called dual images, have been proposed and developed.

The dual-image RDH methods [33, 34, 35] used a function matrix consisting of 256 rows and 256 columns with elements calculated by a modification direction formula in [5]. These methods have a large embedding capacity but low stego image qualities.

Dual-image RDH methods [36, 37] are developed from irreversible data-hiding methods. Specifically, [36] is built from [5] and [37] from [4].

In their study, Lu et al. [38] introduced an RDH technique in two images that utilizes center folding to embed k bits into a pixel x, resulting in the creation of two distinct sego pixels x' and x''. The method proposed by Yao et al. [39] has been enhanced in two key aspects: the expansion of the embedding domain and the increase of one embedded bit when all first k bits are equal to one. Kim et al. [40] employed the averaging technique to embed a bit group into a pixel. The absence of the center folding approach in this method results in a low-quality stego image.

Chen et al. [41] also use the function matrix as in [33] to embed data in a pair of pixels. They take into account four distinct portions around the target pixel: the left horizontal line segment, the right horizontal line segment, the under vertical line segment, and the upper vertical line segment. By selecting one of the above segments to execute the embedding algorithm, an additional embedding bit can be obtained.

The embedding domain of techniques [38] and [39] has been significantly increased by Luyen et al. [42]. So, the method [42] demonstrates improved embedding capacity and image quality in comparison to these two approaches.

Niu et al. [43] introduced a dual image RDH technique that utilizes PVO to achieve a higher embedding capacity than PVO-based approaches in a single image. However, this method only uses the largest and smallest pixels to embed data, so its embedding capacity is limited. In addition, each pixel is modified by a maximum of two units, so the stego image quality is not high.

This paper introduces a novel dual-image RDH method that utilizes PVO with significant advantages compared to existing methods. The main contributions of the paper include: The main contributions of the paper include:

- Suggest an RDH technique in two images using PVO, which simultaneously embeds data in the second-biggest and biggest pixels, as well as the second-smallest and smallest pixels.
- During the data embedding process, each pixel is changed by a maximum of one unit.

- The suggested method has an embedding capacity of approximately 1.65 times higher and a $PSNR$ value about $0.72dB$ higher compared to the method developed by Niu et al.
- Compared with the recent existing dual-image RDH methods, the stego image quality of the proposed method is also higher.

The subsequent sections of the paper are structured sequentially. Section 2 provides a concise introduction to the PVO method and the method proposed by Niu et al. Section 3 presents the novel PVO-based RDH approach in two images. In section 4, experiments and analyses are conducted to compare the suggested method with the one proposed by Niu et al. and other relevant methods. Finally, the conclusions are presented in section 5.

2. **Related works.**

This section provides a concise overview of two interconnected methodologies, specifically the PVO approach proposed by Li et al. and the method introduced by Niu et al.

2.1. PVO **approach.**

The algorithm proposed by Li et al. [20] for embedding data in a block bX of size $r \times c$ can be expressed as follows. First, convert bX into a sequence of pixels $sX = (x_1, \dots, x_n)$, then sort sX in ascending order to get $sX_\sigma = \left(x_{\sigma(1)}, \dots, x_{\sigma(n)}\right)$ such that $x_{\sigma(1)} \leq \cdots \leq x_{\sigma(n)}$, where $n = r \times c$. Also, if $i < j$ and $x_{\sigma(i)} = x_{\sigma(j)}$ then $\sigma_i < \sigma_j$. To embed a bit in the largest pixel, compute $d_{max} = x_{\sigma(n)} - x_{\sigma(n-1)}$, and consider the following three cases:

Case 1: $d_{max} = 0$, no bit is embedded and nothing to do $\left(x'_{\sigma(n)} = x_{\sigma(n)}\right)$.

Case 2: $d_{max} > 1$, no bit is embedded and $x_{\sigma(n)}$ is increased by 1 $\left(x'_{\sigma(n)} = x_{\sigma(n)} + 1\right)$.

Case 3: $d_{max} = 1$, a bit b is embedded in $x_{\sigma(n)}$ by the formula: $x'_{\sigma(n)} = x_{\sigma(n)} + b$.

Similarly, to embed a bit b in the smallest pixel, compute $d_{min} = x_{\sigma(1)} - x_{\sigma(2)}$, and consider the following cases.

Case 1: $d_{\min} = 0$, no bit is embedded and nothing to do $\left(x'_{\sigma(1)} = x_{\sigma(1)}\right)$.

Case 2: $d_{\min} < -1$, no bit is embedded and $x_{\sigma(1)}$ is decreased by 1 $\left(x'_{\sigma(1)} = x_{\sigma(1)} - 1\right)$.

Case 3: $d_{\min} = -1$, a bit b is embedded in $x_{\sigma(1)}$ by the formula: $x'_{\sigma(1)} = x_{\sigma(1)} - b$.

Note that, after embedding data, the PVO (pixel value ordering) of sX_σ is kept unchanged, and pixels $x_{\sigma(i)}$ with $i = 2, \dots, n-1$ are preserved.

At the decoder side, for extracting b and restoring $x_{\sigma(n)}$, compute $d'_{\max} = x'_{\sigma(n)} - x'_{\sigma(n-1)}$, and consider the following cases:

Case 1: $d'_{\max} = 0$, no bit is extracted, $x_{\sigma(n)} = x'_{\sigma(n)}$.

Case 2: $d'_{\max} => 2$, no bit is extracted, $x_{\sigma(n)} = x'_{\sigma(n)} - 1$.

Case 3: $d'_{\max} = 1$ or $d'_{\max} = 2, b = d'_{\max} - 1, x_{\sigma(n)} = x'_{\sigma(n)} - b$.

Similarly for extracting b and restoring $x_{\sigma(1)}$, calculate $d'_{\min} = x'_{\sigma(1)} - x'_{\sigma(2)}$, and consider the following cases:

Case 1: $d'_{\min} = 0$, no bit is extracted, $x_{\sigma(1)} = x'_{\sigma(1)}$.

Case 2: $d'_{\min} < -2$, no bit is extracted, $x_{\sigma(1)} = x'_{\sigma(1)} + 1$.

Case 3: $d'_{\min} = -1$ or $d'_{\min} = -2, b = -\left(d'_{\min} + 1\right), x_{\sigma(1)} = x'_{\sigma(1)} + b$.

A block-based location map (LM) is generated in order to prevent both overflow and underflow as follows: $LM(u, v) = 1$ if the block with a row index of u, and column index of v has $x_{\sigma(n)} = 255$ and $d_{\max} > 0$
or $x_{\sigma(1)} = 0$ and $d_{\min} < 0$, and $LM(u, v) = 0$, otherwise. Blocks with an LM value equal to 1 are likely to cause overflow/ underflow.

TABLE 1. Embedding rule at $x_{\sigma(n)}$ when $d_{\max} \in \{0, 1\}$

Cases	Embedded bits	$\boldsymbol{x'_{\sigma(n)}}$	$\boldsymbol{x''_{\sigma(n)}}$
1	0	$x_{\sigma(n)}$	$x_{\sigma(n)}$
2	100	$x_{\sigma(n)} + 1$	$x_{\sigma(n)}$
3	101	$x_{\sigma(n)}$	$x_{\sigma(n)} + 1$
4	110	$x_{\sigma(n)} + 2$	$x_{\sigma(n)}$
5	111	$x_{\sigma(n)}$	$x_{\sigma(n)} + 2$

TABLE 2. Embedding rule at $x_{\sigma(n)}$ when $d_{\max} > 1$

Cases	Embedded bits	$\boldsymbol{x'_{\sigma(n)}}$	$\boldsymbol{x''_{\sigma(n)}}$
1	0	$\boldsymbol{x_{\sigma(n)}}$	$\boldsymbol{x_{\sigma n)}}$
2	10	$\boldsymbol{x_{\sigma n)} + 1}$	$\boldsymbol{x_{\sigma(n)}}$
3	11	$\boldsymbol{x_{\sigma(n)}}$	$\boldsymbol{x_{\sigma(n)} + 1}$

2.2. **Method of Niu et al.** Like in the *PVO* method [20], Niu et al. [43] partition an initial image X of size $R \times C$ into distinct chunks bX of $r \times c$ size that do not overlap. The conversion of each block bX into a sequence sX is performed. Then sX is sorted as in the *PVO* method to get $sX_\sigma = \left(x_{\sigma(1)}, x_{\sigma(2)}, \ldots, x_{\sigma(n)}\right)$, where $n = r \times c$. Next, Niu et al. calculate a difference d_{max} by the formula: $d_{max} = x_{\sigma(n)} - x_{\sigma(n-1)}$. Embedding data bits in $x_{\sigma(n)}$ to get two stego pixels $x'_{\sigma(n)}$ and $x''_{\sigma(n)}$ is implemented differently in the two cases for d_{max} as follows. In the case $d_{max} \in \{0, 1\}$, embedding data bits in $x_{\sigma(n)}$ is performed according to Table 1.

In the case $d_{max} > 1$, embedding data bits in $x_{\sigma(n)}$ is performed according to Table 2.

Note that embedding rules satisfy the following two important conditions:

Condition 1: $\min\left\{x'_{\sigma(n)}, x''_{\sigma(n)}\right\} = x_{\sigma(n)}$.

Condition 2: *PVO* of stego pixel sequences sX' and sX'' are unchanged. It means the *PVO* of sX' as well as the *PVO* of sX'' is the same as the *PVO* of sX.

One can easily recover the value of $x_{\sigma(n)}$ from condition 1. Next, when knowing $x_{\sigma(n)}$ can extract embedded bits according to the above tables. Finally, based on condition 2, can restore the original pixel sequence sX.

Similarly, for the left side, first, calculate: $d_{\min} = x_{\sigma(1)} - x_{\sigma(2)}$. Then embedding data bits in $x_{\sigma(1)}$ to get two stego pixels $x'_{\sigma(1)}$ and $x''_{\sigma(1)}$ is implemented by rules in Tables 3 and 4.

TABLE 3. Embedding rule at $x_{\sigma(1)}$ when $dmin \in \{0, -1\}$

Cases	Embedded bits	$x'_{\sigma(1)}$	$x''_{\sigma(1)}$
1	0	$x_{\sigma(1)}$	$x_{\sigma(1)}$
2	100	$x_{\sigma(1)} - 1$	$x_{\sigma(1)}$
3	101	$x_{\sigma(1)}$	$x_{\sigma(1)} - 1$
4	110	$x_{\sigma(1)} - 2$	$x_{\sigma(1)}$
5	111	$x_{\sigma(1)}$	$x_{\sigma(1)} - 2$

3. **The proposed method.** Niu et al.'s method [43] only uses the largest and smallest pixels of a block to embed data. So its embedding capacity is limited. Besides, in some

TABLE 4. Embedding rule at $x_{\sigma(1)}$ when $dmin < -1$

Cases	Embedded bits	$x'_{\sigma(1)}$	$x''_{\sigma(1)}$
1	0	$x_{\sigma(1)}$	$x_{\sigma(1)}$
2	10	$x_{\sigma(1)} - 1$	$x_{\sigma(1)}$
3	11	$x_{\sigma(1)}$	$x_{\sigma(1)} - 1$

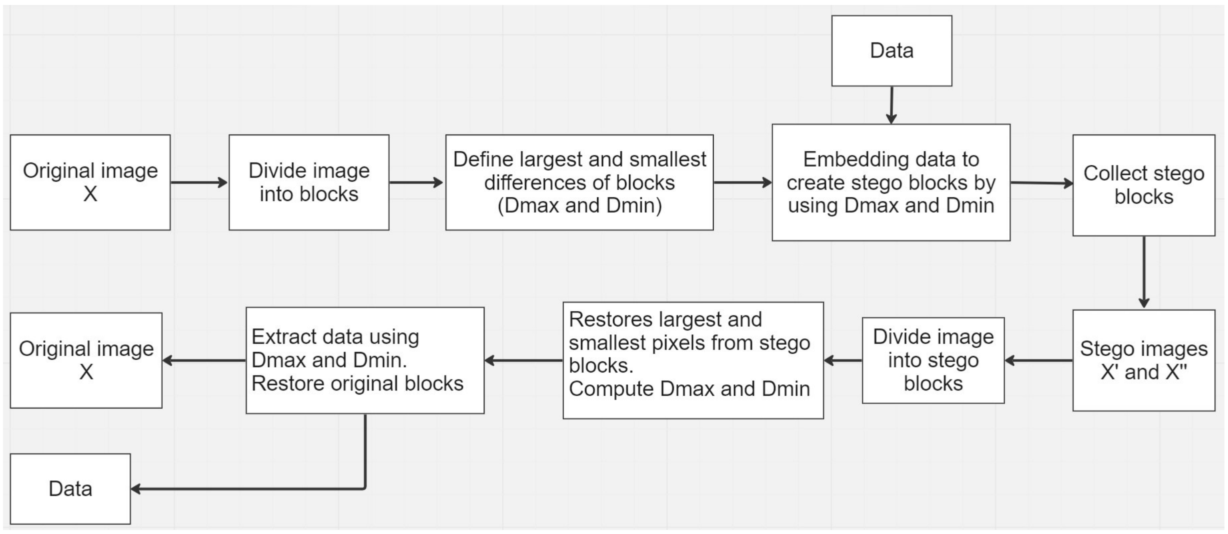
Data
Original image X
Divide image into blocks
Define largest and smallest differences of blocks (Dmax and Dmin)
Embedding data to create stego blocks by using Dmax and Dmin
Collect stego blocks
Stego images X' and X''
Divide image into stego blocks
Restores largest and smallest pixels from stego blocks. Compute Dmax and Dmin
Extract data using Dmax and Dmin. Restore original blocks
Original image X
Data

FIGURE 1. The flowchart of the proposed method

cases of tables 1 and 3, the value of $x_{\sigma(n)}$ and $x_{\sigma(1)}$ are modified by two units. Hence, the method proposed by Niu et al. exhibits a rather low level of stego image quality. In this part, a novel PVO -based RDH approach is introduced for two images. The suggested method allows for the concurrent embedding of data in both the largest and second-largest pixels, as well as the smallest and second-smallest pixels, so it has a greater potential for embedding in comparison to the approach given by Niu et al. In addition, the proposed method only modifies the pixel values by a maximum of one unit, resulting in greater stego image quality in comparison to the approach presented by Niu et al.

This section considers an initial 8-bit image X with dimensions $R \times C$. Image X partitioned into chunks bX_t of $r \times c$ size with $= 1, \ldots, \left\lfloor \frac{R}{r} \right\rfloor \times \left\lfloor \frac{c}{c} \right\rfloor$:

$$bX_t = \begin{pmatrix} x_{t,1,1} & \cdots & x_{t,1,c} \\ \vdots & \ddots & \vdots \\ x_{t,r,1} & \cdots & x_{t,r,c} \end{pmatrix}$$

The block bX_t is converted to a pixel sequence $sX_t = (x_{t,1}, \ldots, x_{t,n})$ where $n = r \times c$. Next, sX_t is arranged in ascending order to obtain a sequence $X_{t,\sigma} = \left(x_{t,\sigma(1)}, \ldots, x_{t,\sigma(n)}\right)$. Below, when considering only one block, the t-index will be removed for compactness.

The flowchart of the proposed method is shown in Fig. 1.

3.1. **Embed data bits in a block.** Considering a block bX, scanning row by row to convert bX into a sequence $sX = (x_1, \ldots, x_n)$, next, the sX is arranged in ascending order to obtain $sX_\sigma = \left(x_{\sigma(1)}, \ldots, x_{\sigma(n)}\right)$ such that $x_{\sigma(1)} \leq \cdots \leq x_{\sigma(n)}$.

3.1.1. *Embed data bits in the two largest pixels.* The process involves embedding two to four bits in pixels $x_{\sigma(n-1)}$ and $x_{\sigma(n)}$ to create two stego pixel-pairs $x'_{\sigma(n-1)}, x'_{\sigma(n)}$ and $x''_{\sigma(n-1)}, x''_{\sigma(n)}$ by the following steps:

Step 1: Compute: $d_{\max} = x_{\sigma(n)} - x_{\sigma(n-1)}$. Consider two cases: $d_{\max} = 0$ and $d_{\max} > 0$ and use different embedding rules in each case.

In case $d_{\max} = 0$, it means $x_{\sigma(n)} = x_{\sigma(n-1}$, put: $\alpha = x_{\sigma(n)} = x_{\sigma(n-1}$, and go to Step 2 .

In the case of $d_{\max} > 0$, it means $x_{\sigma(n)} > x_{\sigma(n-1)}$, put: $\alpha = x_{\sigma(n-1)}, \beta = x_{\sigma(n)}$, and go to Step 3

Step 2: Embed two or three bits in pixel-pair $x_{\sigma(n-1)}$ and $x_{\sigma(n)}$ to get two new pixel-pairs $x'_{\sigma(n-1)}, x'_{\sigma(n)}$ and $x''_{\sigma(n-1)}, x''_{\sigma(n)}$ according to the rules shown in Table 5, where the second column contains embedded bits and the embedding formulas are presented in the next four columns.

Step 3: Embed three or four bits in $x_{\sigma(n-1)}$ and $x_{\sigma(n)}$ to get $x'_{\sigma(n-1)}, x'_{\sigma(n)}$ and $x''_{\sigma(n-1)}, x''_{\sigma(n)}$ according to the rules shown in Table 6.

Table 5. The embedding rule in the case $x_{\sigma(n-1)} = x_{\sigma(n)}$ $(\alpha = x_{\sigma(n-1)})$

Case	Embedded bit sequence	$x'_{\sigma(n-1)}$	$x'_{\sigma(n)}$	$x''_{\sigma(n-1)}$	$x''_{\sigma(n)}$	$x'_{\sigma(n-1)} - \alpha$	$x'_{\sigma(n)} - \alpha$	$x''_{\sigma(n-1)} - \alpha$	$x''_{\sigma(n)} - \alpha$
1	00	α	α	α	α	0	0	0	0
2	01	α	α	α	$\alpha+1$	0	0	0	1
3	10	α	α	$\alpha+1$	$\alpha+1$	0	0	1	1
4	11	α	$\alpha+1$	α	α	0	1	0	0
5	100	$\alpha+1$	$\alpha+1$	α	α	1	1	0	0

Remark 1

From Tables 5 and 6, deduce that the values of pixels $x_{\sigma(n-1)}$ and $x_{\sigma(n)}$ can be restored from $x'_{\sigma(n-1)}, x'_{\sigma(n)}, x''_{\sigma(n-1)}$ and $x''_{\sigma(n)}$ by formulas:

$$x_{\sigma(n-1)} = \min\left(x'_{\sigma(n-1)}, x''_{\sigma(n-1)}\right), x_{\sigma(n)} = \min\left(x'_{\sigma(n)}, x''_{\sigma(n)}\right). \tag{1}$$

After determining the values of $x_{\sigma(n-1)}$ and $x_{\sigma(n)}$, need to locate them in the original block by using the positions of $x'_{\sigma(n-1)}$ and $x'_{\sigma(n)}$ (or of $x''_{\sigma(n-1)}$ and $x''_{\sigma(n)}$) if the order of $x'_{\sigma(n-1)}$ and $x'_{\sigma(n)}$ (or of $x''_{\sigma(n-1)}$ and $x''_{\sigma(n)}\Big)$ is the same as the order of $x_{\sigma(n-1)}$ and $x_{\sigma(n)}$.

Table 6. The embedding rule in the case $x_{\sigma(n-1)} < x_{\sigma(n)}$ $(\alpha = x_{\sigma(n-1)}, \beta = x_{\sigma(n)})$

Cases	Embedded bit sequence	$x'_{\sigma(n-1)}$	$x'_{\sigma(n)}$	$x''_{\sigma(n-1)}$	$x''_{\sigma(n)}$	$x'_{\sigma(n-1)} - \alpha$	$x'_{\sigma(n)} - \beta$	$x''_{\sigma(n-1)} - \alpha$	$x''_{\sigma(n)} - \beta$
1	000	α	β	α	β	0	0	0	0
2	001	α	β	α	$\beta+1$	0	0	0	1
3	010	α	β	$\alpha+1$	β	0	0	1	0
4	011	α	β	$\alpha+1$	$\beta+1$	0	0	1	1
5	100	$\alpha+1$	β	α	$\beta+1$	1	0	0	1
6	101	$\alpha+1$	$\beta+1$	α	β	1	1	0	0
7	110	$\alpha+1$	β	α	β	1	0	0	0
8	111	α	$\beta+1$	α	β	0	1	0	0
9	1000	α	$\beta+1$	$\alpha+1$	β	0	1	1	0

This condition is always true except for the following two cases:

1. When $x_{\sigma(n)} = x_{\sigma(n-1)} + 1(\beta = \alpha + 1)$, but $x'_{\sigma(n-1)} = x'_{\sigma(n)}$ as shown in case 5 or case 7 of Table 6. In this case, the order of $x'_{\sigma(n-1)}$ and $x'_{\sigma(n)}$ maybe different from the order of $x_{\sigma(n-1)}$ and $x_{\sigma(n)}$. However, since $x''_{\sigma(n-1)} < x''_{\sigma(n)}$, the order of $x''_{\sigma(n-1)}$ and $x''_{\sigma(n)}$ is the same as the order of $x_{\sigma(n-1)}$ and $x_{\sigma(n)}$.
2. When $x_{\sigma(n)} = x_{\sigma(n-1)} + 1(\beta = \alpha + 1)$, but $x''_{\sigma(n-1)} = x''_{\sigma(n)}$ as shown in case 3 or case 9 of Table 6.In a similar way, the positions of $x'_{\sigma(n-1)}$ and $x'_{\sigma(n)}$ can be utilized to determine the locations of $x_{\sigma(n-1)}$ and $x_{\sigma(n)}$ within the original block.

This remark will be used to restore $x_{\sigma(n-1)}$ and $x_{\sigma(n)}$ as well as to extract embedded bits.

3.1.2. *Embed data bits in the two smallest pixels.* The process involves embedding two to four bits in pixels $x_{\sigma(1)}$ and $x_{\sigma(2)}$ to create two pixel-pairs $x'_{\sigma(1)}, x'_{\sigma(2)}$ and $x''_{\sigma(1)}, x''_{\sigma(2)}$ by the following steps:

Step 1: Compute: $d_{\min} = x_{\sigma(2)} - x_{\sigma(1)}$. Consider two cases $d_{\min} = 0$ and $d_{\min} > 0$ and use different embedding rules in each case.

In case $d_{\min} = 0$, it means $x_{\sigma(2)} = x_{\sigma(1)}$. Put: $\rho = x_{\sigma(1)} = x_{\sigma(2)}$, and go to Step 2.

In the case of $d_{min} > 0$, it means $x_{\sigma(2)} > x_{\sigma(1)}$. Put: $\rho = x_{\sigma(1)}, \tau = x_{\sigma(2)}$, and go to Step 3

Step 2: Embed two or three bits in pixel-pair $x_{\sigma(1)}$ and $x_{\sigma(2)}$ to get two new pixel-pairs $x'_{\sigma(1)}, x'_{\sigma(2)}$ and $x''_{\sigma(1)}, x''_{\sigma(2)}$ according to the rules shown in Table 7.

Step 3: Embed three or four bits by computing $x'_{\sigma(1)}, x'_{\sigma(2)}$ and $x''_{\sigma(1)}, x''_{\sigma(2)}$ according to the rules shown in Table 8.

Similar to subsection 3.1.1, we can prove the following remark 2.

Remark 2. From Tables 7 and 8, it deduced that the values of the pixels $x_{\sigma(1)}$ and $x_{\sigma(2)}$ can be restored from $x'_{\sigma(1)}, x'_{\sigma(2)}, x''_{\sigma(1)}$ and $x''_{\sigma(2)}$ by formulas:

$$x_{\sigma(1)} = \max\left(x'_{\sigma(1)}, x''_{\sigma(1)}\right), x_{\sigma(2)} = \max\left(x'_{\sigma(2)}, x''_{\sigma(2)}\right). \tag{2}$$

After determining the values of $x_{\sigma(1)}$ and $x_{\sigma(2)}$, need to locate them in the original block by using the order of $x'_{\sigma(1)}$ and $x'_{\sigma(2)}$ (or order of $x''_{\sigma(1)}$ and $\left.x''_{\sigma(2)}\right)$ if the order of $x'_{\sigma(1)}$ and $x'_{\sigma(2)}$ (or of $x''_{\sigma(1)}$ and $x''_{\sigma(2)}$) is the same as the order of $x_{\sigma(1)}$ and $x_{\sigma(2)}$.

TABLE 7. The embedding rule in the case $x_{\sigma(1)} = x_{\sigma(2)}$ $(\rho = x_{\sigma(1)})$

Cases	Embedded bit sequence	$x'_{\sigma(1)}$	$x'_{\sigma(2)}$	$x''_{\sigma(1)}$	$x''_{\sigma(2)}$	$\rho - x'_{\sigma(1)}$	$\rho - x'_{\sigma(2)}$	$\rho - x''_{\sigma(1)}$	$\rho - x''_{\sigma(2)}$
1	00	ρ	ρ	ρ	ρ	0	0	0	0
2	01	ρ	ρ	ρ	$\rho-1$	0	0	0	1
3	10	ρ	ρ	$\rho-1$	$\rho-1$	0	0	1	1
4	11	ρ	$\rho-1$	ρ	ρ	0	1	0	0
5	100	$\rho-1$	$\rho-1$	ρ	ρ	1	1	0	0

This condition is always true except for the following two cases:

1. When $x_{\sigma(2)} = x_{\sigma(1)} + 1$, but $x'_{\sigma(1)} = x'_{\sigma(2)}$ as shown in case 8 or case 9 of Table 8. In this case, the order of $x'_{\sigma(1)}$ and $x'_{\sigma(2)}$ maybe different from the order of $x_{\sigma(1)}$ and $x_{\sigma(2)}$. However, since $x''_{\sigma(1)} < x''_{\sigma(2)}$, the order of $x''_{\sigma(1)}$ and $x''_{\sigma(2)}$ is the same as the order of $x_{\sigma(1)}$ and $x_{\sigma(2)}$.

TABLE 8. The embedding rule in the case $x_{\sigma(1)} < x_{\sigma(2)}$ ($\rho = x_{\sigma(1)}$, $\tau = x_{\sigma(2)}$)

Cases	Embedded bit sequence	$x'_{\sigma(1)}$	$x'_{\sigma(2)}$	$x''_{\sigma(1)}$	$x''_{\sigma(2)}$	$\rho - x'_{\sigma(1)}$	$\tau - x'_{\sigma(2)}$	$\rho - x''_{\sigma(1)}$	$\tau - x''_{\sigma(2)}$
1	000	ρ	τ	ρ	τ	0	0	0	0
2	001	ρ	τ	ρ	$\tau-1$	0	0	0	1
3	010	ρ	τ	$\rho-1$	τ	0	0	1	0
4	011	ρ	τ	$\rho-1$	$\tau-1$	0	0	1	1
5	100	$\rho-1$	τ	ρ	$\tau-1$	1	0	0	1
6	101	$\rho-1$	$\tau-1$	ρ	τ	1	1	0	0
7	110	$\rho-1$	τ	ρ	τ	1	0	0	0
8	111	ρ	$\tau-1$	ρ	τ	0	1	0	0
9	1000	ρ	$\tau-1$	$\rho-1$	τ	0	1	1	0

2. When $x_{\sigma(2)} = x_{\sigma(1)} + 1$, but $x''_{\sigma(1)} = x''_{\sigma(2)}$ as shown in case 2 or case 5 of Table 8. Similarly, in this case, can use the positions of $x'_{\sigma(1)}$ and $x'_{\sigma(2)}$ to locate $x_{\sigma(1)}$ and $x_{\sigma(2)}$ in an original block.

This remark will be used to restore $x_{\sigma(1)}$ and $x_{\sigma(2)}$ as well as to extract embedded bits.

After embedding data bits in a pixel sequence sX_σ on both sides, sequences sX'_σ and sX''_σ are obtained. Next, need to rearrange sX'_σ and sX''_σ by positions in the vector σ to get sX', sX''. Finally, convert sX' and sX'' into blocks bX' and bX'' of size $r \times c$, respectively.

3.1.3. *Example.* Give an image X of 2×4 size:

$$X = \begin{bmatrix} 4 & 6 & 5 & 8 \\ 6 & 3 & 7 & 5 \end{bmatrix}$$

and a bit sequence $D = \begin{bmatrix} 100011 & 110100 \end{bmatrix}$. Split X into $2{\times}2$ blocks $(r = c = 2, n = 4)$ and process each block.

1. Consider block bX_1

$$bX_1 = \begin{bmatrix} 4 & 6 \\ 6 & 3 \end{bmatrix}$$

On the right side, $d_{\max} = 6 - 6 = 0, \alpha = 6$ (Table 5). The three bits from D: 100. By case 5, $x'_{\sigma(3)} = x'_{\sigma(4)} = \alpha + 1 = 7, x''_{\sigma(3)} = x''_{\sigma(4)} = \alpha = 6$. On the left side, $\rho = 3, \tau = 4, d_{min} = 4 - 3 = 1$ (Table 8). The next three bits from D: 011. By case 4, $x'_{\sigma(1)} = \rho = 3$, $x'_{\sigma(2)} = \tau = 4$, and $x''_{\sigma(1)} = \rho -\ 1 = 2, x''_{\sigma(2)} = \tau - 1 = 3$. Thus obtain:

$$bX'_1 = \begin{bmatrix} 4 & 7 \\ 7 & 3 \end{bmatrix}, bX''_1 = \begin{bmatrix} 3 & 6 \\ 6 & 2 \end{bmatrix}$$

2. Consider block bX_2

$$bX_2 = \begin{bmatrix} 5 & 8 \\ 7 & 5 \end{bmatrix}$$

On the right, $d_{\max} = 8 - 7 = 1, \alpha = 7, \beta = 8$ (Table 6). The next three bits from D: 110. By case 7, $x'_{\sigma(3)} = \alpha + 1 = 8, x'_{\sigma(4)} = \beta = 8$, and $x''_{\sigma(3)} = \alpha = 7, x''_{\sigma(4)} = \beta = 8$. On the left, $d_{\min} = 5 -\ 5 = 0, \rho = 5$ (Table 7). The next three bits from D: 100, by case 5, $x'_{\sigma(1)} = x'_{\sigma(2)} = \rho - 1 = 4$, $x''_{\sigma(1)} = x''_{\sigma(2)} = \rho = 5$. Thus obtain:

$bX_2' = \begin{bmatrix} 4 & 8 \\ 8 & 4 \end{bmatrix}, bX_2'' = \begin{bmatrix} 5 & 8 \\ 7 & 5 \end{bmatrix}.$

In summary, we obtain:

$X' = \begin{bmatrix} 4 & 7 & 4 & 8 \\ 7 & 3 & 8 & 4 \end{bmatrix}, X'' = \begin{bmatrix} 3 & 6 & 5 & 8 \\ 6 & 2 & 7 & 5 \end{bmatrix}.$

3.2. **Extract embedded bits and restore the original block.** Given stego blocks bX' and bX''. To restore the original block bX and extract embedded bits, first, convert these blocks to pixel-sequences sX' and sX' (as in the embedding algorithm). Next, sort sX' and sX'' in ascending order to have sX'_σ and X''_σ : $sX'_\sigma = \left(x'_{\sigma(1)}, \ldots, x'_{\sigma(n)}\right)$, and $sX''_\sigma = \left(x''_{\sigma(1)}, \ldots, x''_{\sigma(n)}\right)$.

In the proposed method only consider $n \geq 4$, thence, there is no overlap between the two largest pixels and the two smallest pixels of sX'_σ and sX''_σ. So, extracting and restoring can be performed independently on the largest and smallest sides.

3.2.1. *Extracting and restoring on the largest side.* Extracting embedded bits and restoring $x_{\sigma(n-1)}$ and $x_{\sigma(n)}$ are done according to Remark 1 by the following steps:

Step 1: Define the values of the original pixels $x_{\sigma(n-1)}$ and $x_{\sigma(n)}$ by formula (1) as follows:

$$\begin{aligned} \alpha = x_{\sigma(n-1)} &= \min\left\{x'_{\sigma(n-1)}, x''_{\sigma(n-1)}\right\}, \\ \beta = x_{\sigma(n)} &= \min\left\{x'_{\sigma(n)}, x''_{\sigma(n)}\right\}. \end{aligned} \tag{3}$$

Step 2: Determine the positions of $x_{\sigma(n-1)}$ and $x_{\sigma(n)}$ in the original pixel block.

Case 1: $x_{\sigma(n-1)} = x_{\sigma(n)}$. The position of $x_{\sigma(n-1)}$ (or $x_{\sigma(n)}$) is chosen to be the position of $x'_{\sigma(n-1)}$ (or $x'_{\sigma(n)}$).

Case 2: $x_{\sigma(n-1)} < x_{\sigma(n)}$.

(a) If $x'_{\sigma(n-1)} < x'_{\sigma(n)}$, the position of $x_{\sigma(n-1)}$ (or $x_{\sigma(n)}$) is chosen to be the position of $x'_{\sigma(n-1)}$ $\left(\text{or}\, x'_{\sigma(n)}\right)$.

(b) If $x'_{\sigma(n-1)} = x'_{\sigma(n)}$, the position of $x_{\sigma(n-1)}$ (or $x_{\sigma(n)}$) is chosen to be the position of $x''_{\sigma(n-1)}$ $\left(\text{ or } x''_{\sigma(n)}\right)$

To extract embedded bits, consider the two following cases: $\alpha = \beta$ and $\alpha < \beta$. In the first case, go to Step 3; otherwise, go to Step 4.

Step 3: When $\alpha = \beta$ (corresponding to Table 5), calculate the differences $x'_{\sigma(n-1)} - \alpha, x'_{\sigma(n)} - \alpha, x''_{\sigma(n-1)} - \alpha$, and $x''_{\sigma(n)} - \alpha$ and compare them with corresponding differences in the last four columns of Table 5 to get the embedded data bits.

Step 4: When $\alpha < \beta$ (corresponding to Table 6), calculate the differences $x'_{\sigma(n-1)} - \alpha, x'_{\sigma(n)} - \beta, x''_{\sigma(n-1)} - \alpha$, and $x''_{\sigma(n)} - \beta$ and compare them with corresponding differences in Table 6 to get embedded data bits.

3.2.2. *Extracting and restoring on the smallest side.* Similar to subsection 3.2.1, remark 2 will be used to extract embedded bits and to restore $x_{\sigma(1)}$ and $x_{\sigma(2)}$ by the following steps:

Step 1: Define the values of the original pixels $x_{\sigma(1)}$ and $x_{\sigma(2)}$ by the formula (2) as follows:

$$\begin{aligned} \rho = x_{\sigma(1)} = \max\left(x'_{\sigma(1)}, x''_{\sigma(1)}\right), \\ \tau = x_{\sigma(2)} = \max\left(x'_{\sigma(2)}, x''_{\sigma(2)}\right). \end{aligned} \tag{4}$$

Step 2: Determine the positions of $x_{\sigma(1)}$ and $x_{\sigma(2)}$ in the original pixel block.

Case 1: $x_{\sigma(1)} = x_{\sigma(2)}$. The position of $x_{\sigma(1)}$ $\left(\text{ or } x_{\sigma(2)}\right)$ is chosen to be the position of $x'_{\sigma(1)}$ (or $x'_{\sigma(2)}$).

Case 2: $x_{\sigma(1)} < x_{\sigma(2)}$.

(a) If $x'_{\sigma(1)} < x'_{\sigma(2)}$, the position of $x_{\sigma(1)}$ $\left(\text{ or } x_{\sigma(2)}\right)$ is chosen to be the position of $x'_{\sigma(1)}$ (or $x'_{\sigma(2)}$).

(b) If $x'_{\sigma(1)} = x'_{\sigma(2)}$, the position of $x_{\sigma(1)}$ $\left(\text{ or } x_{\sigma(2)}\right)$ is chosen to be the position of $x''_{\sigma(1)}$ (or $x''_{\sigma(2)}$).

To extract embedded bits, consider the two following cases: $\rho = \tau$ and $\rho < \tau$. In the first case, go to Step 3; otherwise, go to Step 4.

Step 3: When $\rho = \tau$ (corresponding to Table 7), calculate the differences $\rho - x'_{\sigma(1)}, \rho - x'_{\sigma(2)}, \rho - x''_{\sigma(1)}$, and $\rho - x''_{\sigma(2)}$ and compare them with corresponding differences in Table 7 to get embedded data bits

Step 4: When $\rho < \tau$ (corresponding to Table 8), calculate the differences $\rho - x'_{\sigma(1)}, \tau - x'_{\sigma(2)}, \rho - x''_{\sigma(1)}$, and $\tau - x''_{\sigma(2)}$ and compare them with corresponding differences in Table 8 to get embedded data bits.

After restoring the pixels $x_{\sigma(1)}, x_{\sigma(2)}, x_{\sigma(n-1)}$, and $x_{\sigma(n)}$, the original block bX is obtained. In addition, embedded bits are obtained by collecting bits from the largest and smallest sides.

3.2.3. *Example.* Given stego images:

$$X' = \left[\begin{array}{cccc} 4 & 7 & 4 & 8 \\ 7 & 3 & 8 & 4 \end{array}\right], X'' = \left[\begin{array}{cccc} 3 & 6 & 5 & 8 \\ 6 & 2 & 7 & 5 \end{array}\right]$$

To extract embedded bits and restore the original image, divide stego images into 2×2 blocks $(r = c = 2, n = 4)$ and process each pair of blocks.

1.Consider bX'_1 and bX''_1 :

$$bX'_1 = \left[\begin{array}{cc} 4 & 7 \\ 7 & 3 \end{array}\right], bX''_1 = \left[\begin{array}{cc} 3 & 6 \\ 6 & 2 \end{array}\right]$$

On the right, $x'_{\sigma(3)} = 7, x'_{\sigma(4)} = 7, x''_{\sigma(3)} = 6, x''_{\sigma(4)} = 6, x_{\sigma(3)} = \min\left(x'_{\sigma(3)}, x''_{\sigma(3)}\right) = 6, x_{\sigma(4)} = \min\left(x'_{\sigma(4)}, x''_{\sigma(4)}\right) = 6$ (Table 5 with $\alpha = 6$). Compute $x'_{\sigma(3)} - \alpha = 1, x'_{\sigma(4)} - \alpha = 1, x''_{\sigma(3)} - \alpha = 0, x''_{\sigma(4)} - \alpha = 0$. Comparing these differences with the values in the last four columns of Table 5, infer that the embedded bits are 100. On the left, $x'_{\sigma(1)} = 3, x'_{\sigma(2)} = 4, x''_{\sigma(1)} = 2, x''_{\sigma(2)} = 3, x_{\sigma(1)} = \max\left(x'_{\sigma(1)}, x''_{\sigma(1)}\right) = 3, x_{\sigma(2)} = \max\left(x'_{\sigma(2)}, x''_{\sigma(2)}\right) = 4$, (Table 8 with $\rho = 3, \tau = 4$). Compute: $\rho - x'_{\sigma(1)} = 0, \tau - x'_{\sigma(2)} = 0, \rho - x''_{\sigma(1)} = 1, \tau - x''_{\sigma(2)} = 1$. Comparing these differences with the corresponding values of Table 8, deduce that embedded bits are 011. Locate $x_{\sigma(3)}$ and $x_{\sigma(4)}$ by $x'_{\sigma(3)}$ and $x'_{\sigma(4)}$, locate $x_{\sigma(1)}$ and $x_{\sigma(2)}$ by $x'_{\sigma(1)}$ and $x'_{\sigma(2)}$, obtain $bX_1 = \left[\begin{array}{cc} 4 & 6 \\ 6 & 3 \end{array}\right]$, and bits $= 1000011$.

2. Consider bX'_2 and bX''_2 :

$$bX_2' = \begin{bmatrix} 4 & 8 \\ 8 & 4 \end{bmatrix}, bX_2'' = \begin{bmatrix} 5 & 8 \\ 7 & 5 \end{bmatrix}$$

On the right, $x'_{\sigma(3)} = 8, x'_{\sigma(4)} = 8, x''_{\sigma(3)} = 7, x''_{\sigma(4)} = 8, x_{\sigma(3)} = \min\left(x'_{\sigma(3)}, x''_{\sigma(3)}\right) = 7, x_{\sigma(4)} = \min\left(x'_{\sigma(4)}, x''_{\sigma(4)}\right) = 8$ (Table 6 with $\alpha = 7, \beta = 8$). Compute: $x'_{\sigma(3)} - \alpha = 1$, $x'_{\sigma(4)} - \beta = 0, x''_{\sigma(3)} - \alpha = 0, x''_{\sigma(4)} - \beta = 0$. Comparing these values with corresponding differences in Table 6, we deduce that the embedded bits are 110. On the left, $x'_{\sigma(1)} = 4, x'_{\sigma(2)} = 4, x''_{\sigma(1)} = 5, x''_{\sigma(2)} = 5, x_{\sigma(1)} = \max\left(x'_{\sigma(1)}, x''_{\sigma(1)}\right) = 5, x_{\sigma(2)} = \max\left(x'_{\sigma(2)}, x''_{\sigma(2)}\right) = 5$ (Table 7 with $\rho = 5$). Compute: $\rho - x'_{\sigma(1)} = 1, \rho - x'_{\sigma(2)} = 1, \rho - x''_{\sigma(1)} = 0, \rho - x''_{\sigma(2)} = 0$. Comparing these values with the corresponding differences in Table 7, infer that embedded bits are 100. Locate $x_{\sigma(3)}$ and $x_{\sigma(4)}$ by $x''_{\sigma(3)}$ and $x''_{\sigma(4)}$, locate $x_{\sigma(1)}$ and $x_{\sigma(2)}$ by $x'_{\sigma(1)}$ and $x'_{\sigma(2)}$, obtain

$$bX_2 = \begin{bmatrix} 5 & 8 \\ 7 & 5 \end{bmatrix}, \text{ and bits } = 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0.$$

In summary, have

$$X = \begin{bmatrix} 4 & 6 & 5 & 8 \\ 6 & 3 & 7 & 5 \end{bmatrix},$$
$$D = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \end{bmatrix},$$

3.3. **Conceal data bits in an initial image.** Given an original image X of size $R \times C$ and a data bit sequence D. To obtain two stego images, X' and X'', D must be concealed in X. The embedding is done step by step as follows:

Step 1: Divide X into K non-overlapped blocks bX_k of size $r \times c, n = r \times c \geq 4$ with $k = 1, \ldots, K$, where $K = \left\lfloor \frac{R}{r} \right\rfloor \times \left\lfloor \frac{C}{c} \right\rfloor$

Set $k = 1$

Step 2: Compute the minimum and maximum values of the pixels in X_k : $\min_k = \min(bX_k), \max_k = \max(bX_k)$. If $\min_k = 0$ or $\max_k = 255$, embedding data bits in bX_k can cause an overflow or underflow. So in this case, no bit is embedded, set $bX'_k = bX''_k = bX_k$, and go to Step 4. In the opposite case, go to Step 3.

Step 3: Embed from 4 to 8 bits selected from D in the block bX_k to get stego pixel blocks bX'_k and bX''_k according to the algorithm presented in subsection 3.1.

Step 4: Classify into three cases to consider:

1.The value of k is equal to K,

2.The embedding of enough data bits is completed,

3.Another case.

In case 1, go to Step 5.

In case 2, copy the remaining original blocks into the respective stego blocks: $bX'_t = bX''_t = bX_t$, with $t = k + 1, \ldots, K$.

Then go to Step 5

In case 3, increase k by one: $k = k + 1$, and go back to Step 2.

Step 5: Create a stego image X' (and X'') consists of blocks bX'_k (and bX''_k) with $k = 1, \ldots, K$.

3.4. **Extract embedded bits and restore the original image.** Given two stego images X' and X'' with dimension $R \times C$. Require the extraction of a sequence D consisting of embedded bits and the restoration of an exact image X. The procedure of extracting and restoring is done step by step as follows:

Step 1: Divide X' and X'' into K non-overlapping chunks bX'_k and bX''_k of size $r \times c$ (as in Subsection 3.3) with $k = 1, \ldots, K$, where $K = \left\lfloor \frac{R}{r} \right\rfloor \times \left\lfloor \frac{c}{c} \right\rfloor$. Set $k = 1$.

Step 2: Compute the minimum and maximum values of the pixels in blocks bX'_k and bX''_k :

$\min'_k = \min\left(bX'_k\right), \max'_k = \max\left(bX'_k\right)$

$\min''_k = \min\left(bX''_k\right), \max''_k = \max\left(bX''_k\right)$.

If $\min'_k = \min''_k = 0$ or $\max'_k = \max''_k = 255$, go to Step 3. In the opposite case, go to Step 4.

Step 3: No bit is extracted, define an original pixel block bX_k by the formula: $bX_k = bX'_k$.

Go to Step 5.

Step 4: Extract 4 to 8 bits from bX'_k and bX''_k and restore an original pixel block bX_k according to the algorithm presented in subsection 3.2 (formulas (3-4)).

Step 5: Classify into three cases to consider:

1.The value of k is equal to K,

2.The extracting of enough data bits is completed,

3.Another case.

In case 1, go to Step 6.

In case 2, copy the remaining stego blocks into the respective original blocks: $bX_t = bX'_t$, with $t = k + 1, \ldots, K$. Then go to Step 6

In case 3, increase k by one: $k = k + 1$, and go back to Step 2.

Step 6: Restore the original image X consisting of blocks bX_k with $= 1, \ldots, K$. The sequence D of embedded data bits can be obtained by concatenating the extracted bits.

4. **Experiment and discussion.**

This part conducts experiments and discussions to compare the suggested method with other contemporary approaches. Experiments were performed on a set of 12 standard 512×512 grayscale images from the database https://ccia.ugr.es/cvg/ (Fig. 2) to assess the suggested method in relation to the existing methods of Li et al. [17], Peng et al. [22], Ou et al. [21], Li et al. [23], Sao et al. [24], Kumar et al. [29], Nguyen et al. [32], Niu et al. [43], Lu et al. [38], Kim et al. [40] and Chen et al. [41]. The software applications are written utilizing the Matlab platform and subsequently performed on the Lenovo IdeaPad S410p computer.

4.1. **A comparative analysis of the suggested approach and methods using PVO.**

4.1.1. *Comparative analysis of embedding the capacity.* First, compare the proposed method with the method of Niu et al. and PVO-based methods to see the advantages of using PVO in two images compared to using PVO in one image. Table 9 presents the embedding capacity of all compared methods. Dividing the last row of this table by 512×512 gives the result that the average number of embedded bits per pixel (bpp) of PVO-based methods in an image: [17], [22], [21], [23], [24], [29], [32] is 0.128, 0.173, 0.164, 0.175, 0.193, 0.260, 0.286, respectively. Meanwhile, this ratio is 0.865 and 1.432 for PVO-based methods in two images: method [43] and the proposed method. Additionally, this finding demonstrates that the suggested method has a higher embedding capacity compared to alternative methods. The embedding capacity of the suggested technique rises by 11.160, 8.297, 8.728, 8.199, 7.435, 5.512, 5.001, and 1.655 times, respectively, when compared to [17], [22], [21], [23], [24], [29], [32] and [43].

(A) Airplane (B) Barbara (C) Couple (D) Boat

(E) Lena (F) Pepper (G) Goldhill (H) Cabeza

(I) Cameraman (J) Blob (K) Car (L) Tiffany

Figure 2. Experimental images

Table 9. Embedding capacity comparison between methods based on *PVO*

Image	[**17**]	[**22**]	[**21**]	[**23**]	[**24**]	[**29**]	[**32**]	[**43**]	Propose
Airplane	38454	53055	51505	56305	61566	79582	88526	231408	372210
Barbara	29490	48066	40202	41008	51775	72116	65065	226869	372338
Couple	21599	29950	25127	26868	28044	45001	54215	216957	382643
Boat	27392	35159	32783	33321	37044	52810	57069	220585	381972
Lena	32108	38713	38951	384141	459709	58069	73345	223482	381121
Pepper	28093	30886	32422	34853	35023	46292	58015	218736	386252
Goldhill	25955	28662	29918	31210	32183	42993	52871	217013	387481
Cabeza	54263	71946	73416	79198	81389	107951	121482	244549	362416
Cameraman	43458	71097	63378	69771	79650	106612	104444	240561	358931
Blob	41495	53326	52773	56473	61386	79989	90400	232693	372650
Car	31378	47327	41594	44644	56457	70990	77183	227166	373529
Tiffany	29995	34776	34091	34418	38684	54966	58260	222245	373664
Average	33640	45247	43013	45791	50493	68114	75073	226855	375434

4.1.2. *Comparison of Stego image quality.* The comparison of stego image quality between methods with embedded bit string lengths of 10000, 20000 and 30000 bits is presented in Table 10.

Table 10. Comparison of $PSNR$ between methods

Payload	[**17**]	[**22**]	[**21**]	[**23**]	[**24**]	[**29**]	[**32**]	[**43**]	Proposed
10000	58.588	58.965	57.708	53.313	55.497	55.268	55.955	68.222	68.984
20000	55.272	55.604	54.421	0.962	53.018	52.092	52.770	65.220	65.993
30000	54.891	54.07	53.007	49.706	51.727	50.024	50.74	63.488	64.24

Table 11. Comparing the value of $PSNR$ between dual-image methods

Payload	Kim [40]	Chen [41]	Lu [38]	Niu [43]	Proposed
100000	47.291	50.875	58.329	58.367	59.038
200000	43.431	48.133	55.309	55.415	56.043
300000	40.719	46.369	53.564		54.289

The $PSNR$ values in Table 10 are the average values obtained from the 12 standard test images in Fig. 1. For the approach [43] and the suggested approach, the value of $PSNR$ refers to the mean of the $PSNR$ values obtained from two stego images.

In the case of the embedded data size of 10000 bits, the mean $PSNR$ of the proposed method is higher than that of traditional PVO-based methods [17], [22], [21], [23], [24], [29] and [32] are approximately 10.396, 10.019, 11.276, 15.671, 13.487, 13.716 and 13.029(dB). Compared with the method of Niu et al., the peak signal-to-noise ratio $PSNR$ of the suggested approach exceeds 0.762dB. When the embedded data size is 20000 bits, these values will be slightly larger. Specifically, the mean $PSNR$ of the proposed method is higher than that of the traditional PVO - based methods by about 10.721, 10.389, 11.572, $15.031, 12.975, 13.901, 13.223(dB)$ and is greater than the method of Niu et al. 0.773dB. In the case of an embedded data size of 30000 bits, these values are 9.349, 10.170, 11.233, 14.534, 2.5130, 14.216, 13.500 and 0.752, respectively.

The reason that PVO methods on two images have higher $PSNR$ values than traditional PVO methods can be explained as follows. To embed the same number of bits, the previous methods modify two images simultaneously, while the methods below modify only one image. Therefore, the image distortion of PVO methods on two images will be less than that of traditional PVO methods.

4.2. **Evaluating the suggested approach and dual-image methodologies.** The block size used for the approaches of Kim et al., Niu et al., and the proposed method is 2×2. The value of the threshold k in the method of Lu et al. is equal to 2. The suggested method is evaluated in comparison to the approach offered by Niu et al. [43] as well as other dual-image reversible data-hiding approaches proposed by Kim et al. [40], Chen et al. [41], and Lu et al. [38]. The comparison results are presented in Table 11 and Fig. 3. The $PSNR$ values in Table 11 are calculated as follows. The $PSNR$ values in Table 11 are calculated as follows. First, calculate the average $PSNR$ value of the stego images in the 12 standard test images. Then, $PSNR$ is the average value of $PRSN$ values of the two stego images.

It should be noted that the embedding capacity of the approach proposed by Niu et al. is below 300000 bits, so the lowest cell of Niu [43] is empty. From Table 11 and Fig. 3, it can be inferred that the stego image quality of the proposed method is superior to other methods, including the method of Niu et al.

4.3. **Analyzing the stego image quality of the dual-image methods.** Suppose, after embedding data into pixel x, will receive stego pixels x' and x''. To determine the

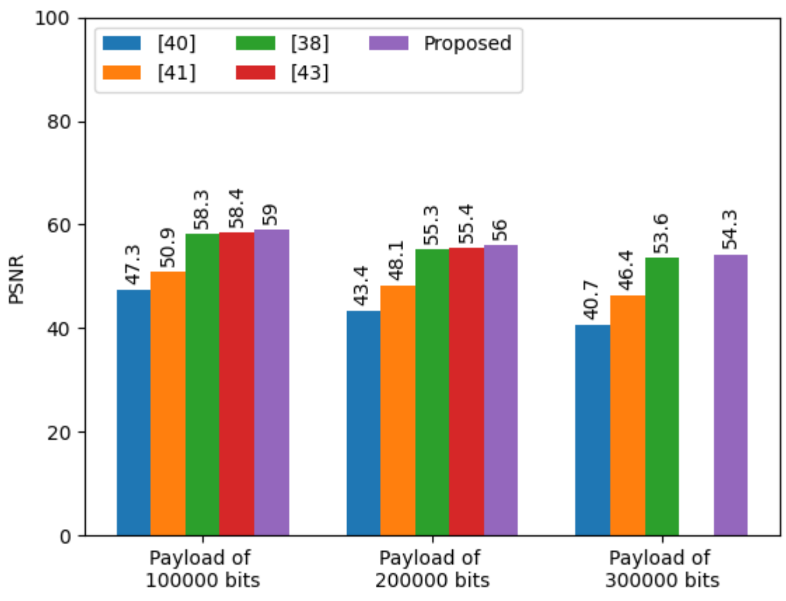

FIGURE 3. *PSNR* comparison of the proposed method with dual-image RDH methods

degree of distortion of an image, it is necessary to evaluate the average value of $(x-x')^2$ and $(x-x'')^2$, denoted by $RM\left((x-x')^2\right)$ and $RM\left((x-x'')^2\right)$, when embedding a bit in x.

In the method of Chen et al. [41], for embedding about 3 bits in a pixel $x \in [5, 251]$, a stego pixel x' or x'' is selected in the segment $[x-5, x-1]$ or $[x, x+4]$, so the value of $(x-x')^2$ and $(x-x'')^2$ can be equal to 25. So this method has a large image distortion.

In the method of Kim et al. [40], the values of $(x-x')^2$ and $(x-x'')^2$ can be reached 64^2, so the stego image quality of this method is lower than that of Chen et al.'s method

In the method of Niu et al. [43], the absolute values of $(x-x')$ and $(x-x'')$ are almost less than or equal to 1, only equal to 2 in very rare cases. Especially, in the method of Lu et al. [38] and the proposed method, this absolute value is one at most. So the *PSNR* value of these three methods is significantly higher than the two methods above.

The estimation of the degree of image distortion in the latter three methods is performed below.

4.3.1. *Consider Lu et al.'s approach with a threshold condition of* $k = 2$. Estimating the *PSNR* in subsections 4.3.1, 4.3.2 and 4.3.3 is based on the assumption that embedded bits follow the uniform distribution.

The method of Lu et al. embeds data into each pixel separately and follows the same embedding rules. Therefore, one will evaluate the distortion degree of a pixel after embedding.

When the value of threshold k is 2, the method of Lu et al. will embed two bits b_1, b_2 in each pixel x as follows. Convert b_1, b_2 into a decimal number d with $0 \le d \le 3$. Compute $\bar{d} = d - 2, u = \left\lfloor \frac{\bar{d}}{2} \right\rfloor, v = \left\lceil \frac{\bar{d}}{2} \right\rceil$. Finally, The formulas that define stego pixels x' and x'' are as follows: $x' = x + u$, and $x'' = x - v$ (see Table 12).

Since the value of d is uniformly distributed from 0 to 3, from Table 12, the mean value of $(x-x')^2$ can be evaluated as follows: $M\left((x-x')^2\right) = \frac{2}{4} = \frac{1}{2}$. According to Lu

TABLE 12. Computing x' and x'' in Lu et al.'s method

Case	$\boldsymbol{b}_1, \boldsymbol{b_2}$	**d**	$\overline{\boldsymbol{d}}$	**u**	**v**	$\boldsymbol{x'}$	$\boldsymbol{x''}$
1	00	0	-2	-1	-1	$x-1$	$x+1$
2	01	1	-1	-1	0	$x-1$	x
3	10	2	0	0	0	x	x
4	11	3	1	0	1	x	$x-1$

et al.'s method, the number of embedded bits in x is 2, resulting in a ratio per bit of $M\left((x-x')^2\right)$ are equal to $RM\left(x,x'\right) = \frac{1}{2} : 2 = \frac{1}{4} = 0.25$. This has a meaning that if embedding one bit in a pixel x, the average value of $M\left((x-x')^2\right)$ is 0.25. Similarly, it follows that

$$RM\left(x,x''\right) = RM\left(x,x'\right) = 0.25. \tag{5}$$

4.3.2. *Analyzing the approach proposed by Niu et al.* The method of Niu et al. classifies the pixel blocks into four types corresponding to the four tables 1 to 4. Embedding data into blocks of the same type is done according to the same rules. Therefore, one will evaluate the distortion degree of each block in a type after embedding.

First, consider the embedding rules according to Table 1 ($d_{\max} \le 1$). Since the embedded bit sequence follows a uniform distribution, the probability of case 1 is 0.5, and the probability of cases from 2 to 5 (in Table 1) is 1/8. Then the mean value of $\left(x_{\sigma(n)} - x'_{\sigma(n)}\right)^2$ is equal: $M\left(\left(x_{\sigma(n)} - x'_{\sigma(n)}\right)^2\right) = \frac{1}{8} \times 5$. In this case, two bits can be embedded, so

$$RM\left(x_{\sigma(n)}, x'_{\sigma(n)}\right) = \frac{1}{8} \times 5 : 2 = \frac{5}{16} = 0.313, \text{ if } d_{\max} \le 1$$

Using the same arguments from Table 2, infer that:
$RM\left(x_{\sigma(n)}, x'_{\sigma(n)}\right) = \frac{1}{4} : \frac{3}{2} = 0.167$, if $d_{\max} \ge 2$.

By computing over 12 test images, the number of blocks with $d_{\max} \le 1$ is 30581 and the number of blocks with $d_{\max} \ge 2$ is 34955. From this, it follows that
$RM\left(x_{\sigma(n)}, x'_{\sigma(n)}\right) = \frac{30581 \times 0.313 + 34955 \times 0.17}{30581 + 34955} = 0.235$
Similarly, have

$$RM\left(x_{\sigma(n)}, x'_{\sigma(n)}\right) = RM\left(x_{\sigma(n)}, x''_{\sigma(n)}\right) = 0.235. \tag{6}$$

Using Tables 3- 4 and the above arguments, deduce that

$$RM\left(x_{\sigma(1)}, x'_{\sigma(1)}\right) = RM\left(x_{\sigma(1)}, x''_{\sigma(1)}\right) = 0.235. \tag{7}$$

Comparing formulas (5) with formulas (6) and (7), it can be deduced that the level of image distortion in the approach proposed by Niu et al. is comparatively lower than that in the approach proposed by Lu et al.

4.3.3. *Analyzing the proposed method.* The proposed method classifies the pixel blocks into four types corresponding to the four tables 5 to 8. Embedding data into blocks of the same type is done according to the same rules. Therefore, one will evaluate the distortion degree of each block in a type after embedding.

First, consider the embedding rules according to Table 5 $\left(d_{\max} = x_{\sigma(n-1)}\text{-}x_{\sigma(n)} = 0\right)$. Since the embedded bit sequence follows a uniform distribution, the probabilities of the occurrence of the subsequences $000, 001, 010, 011, 100, 101, 110, 111$ are equal. Therefore the probabilities of cases $1, 2, 3, 4$, and 5 (in Table 5) are $\frac{1}{4}, \frac{1}{4}, \frac{1}{8}, \frac{1}{4}$ and $\frac{1}{8}$. So $M\left(\left(x_{\sigma(n-1)} - x'_{\sigma(n-1)}\right)^2 + \left(x_{\sigma(n)} - x'_{\sigma(n)}\right)^2\right) = \frac{1}{4} \times 1 + \frac{1}{8} \times 2 = \frac{1}{2}$.

Where $M(Z)$ denotes the average value of Z. Because in this case, $\left(2 + \frac{1}{8}\right)$ bits can be embedded, so

$$RM\left(\left(x_{\sigma(n-1)} - x'_{\sigma(n-1)}\right)^2 + \left(x_{\sigma(n)} - x'_{\sigma(n)}\right)^2\right) = \frac{1}{2} : \left(2 + \frac{1}{8}\right) = \frac{4}{17} = 0.235, \text{ if } d_{\max} = 0$$

Similarly, from Table 6 it can be inferred

$$RM\left(\left(x_{\sigma(n-1)} - x'_{\sigma(n-1)}\right)^2 + \left(x_{\sigma(n)} - x'_{\sigma(n)}\right)^2\right) = \frac{5}{8} : \left(3 + \frac{1}{16}\right) = \frac{10}{49} = 0.204, \text{ if } d_{\max} > 0$$

Statistics from 12 test images show that the number of blocks with $d_{\max} = 0$ is 13557, and the number of blocks with $d_{\max} > 0$ is 51979, so we have

$$RM\left(\left(x_{\sigma(n-1)} - x'_{\sigma(n-1)}\right)^2 + \left(x_{\sigma(n)} - x'_{\sigma(n)}\right)^2\right) = \frac{13557 \times 0.235 + 51979 \times 0.204}{13557 + 0.204} = 0.2104. \tag{8}$$

By similar reasoning, it is deduced that

$$\begin{gathered} RM\left(\left(x_{\sigma(n-1)} - x'_{\sigma(n-1)}\right)^2 + \left(x_{\sigma(n)} - x'_{\sigma(n)}\right)^2\right) = \\ RM\left(\left(x_{\sigma(n-1)} - x''_{\sigma(n-1)}\right)^2 + \left(x_{\sigma(n)} - x''_{\sigma(n)}\right)^2\right) = 0.2104. \end{gathered} \tag{9}$$

Based on formulas (5-9), it can be deduced that the suggested approach exhibits a reduced level of image distortion compared to the techniques presented by Lu et al. and Niu et al. In other words, the $PSNR$ value of the proposed method is higher than that of these two methods. This conclusion is completely consistent with the results presented in Table 11.

5. **Conclusion.**

In this paper, we propose a PVO-based RDH method in two images. The initial image is partitioned into distinct blocks that do not overlap, with each block containing a minimum of four pixels. Each block's pixels are transformed into a sequence of pixels and subsequently arranged in ascending order. Two stego blocks are formed by concurrently embedding four to eight bits in the second-largest and largest pixels, as well as in the second-smallest and smallest pixels. During the process of embedding, the maximum alteration made to each pixel is one unit. Hence, in contrast to the approach employed by Niu et al. and other conventional PVO-based techniques, the suggested method exhibits superior embedding capacity and enhanced stego image quality. In comparison to the approach employed by Niu et al., the method described in this study exhibits a significantly greater embedding capacity of over 1.65 times and a $PSNR$ value that is $0.72dB$ higher throughout all conducted tests. Furthermore, both empirical findings and theoretical examination indicate that the $PSNR$ (Peak Signal-to-Noise Ratio) of the suggested approach surpasses that of certain existing dual-image techniques.

Acknowledgment. This work is funded by the Vietnam Academy of Science and Technology (VAST) under grant number CSCL02.05/22-23.
The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation.

REFERENCES

[1] Pan, H.-K., Chen, Y.-Y., Tseng, Y.-C.: A secure data hiding scheme for two-color images. *In: Proceedings ISCC 2000. Fifth IEEE Symposium on Computers and Communications*, pp. 750-755 (2000). https://doi.org/10. 1109/ISCC.2000.860731

[2] Podilchuk, C.I., Delp, E.J.: Digital watermarking: algorithms and applications. *IEEE Signal Processing Magazine* 18(4), 33-46 (2001). https: //doi.org/10.1109/79.939835

[3] Wu, D.-C., Tsai, W.-H.: A steganographic method for images by pixelvalue differencing. *Pattern Recognition Letters* 24(9), 1613-1626 (2003). https://doi.org/10.1016/S0167-8655(02)00402-6

[4] Mielikainen, J.: Lsb matching revisited. *IEEE Signal Processing Letters* 13(5), 285-287 (2006). https://doi.org/10.1109/LSP.2006.870357

[5] Zhang, X., Wang, S.: Efficient steganographic embedding by exploiting modification direction. *IEEE Communications Letters* 10(11), 781 – 783 (2006).
https://doi.org/10.1109/LCOMM.2006.060863

[6] Tian, J.: Reversible data embedding using a difference expansion. *IEEE Transactions on Circuits and Systems for Video Technology* 13(8), 890896 (2003).
https://doi.org/10.1109/TCSVT.2003.815962

[7] Alattar, A.M.: Reversible watermark using the difference expansion of a generalized integer transform. *IEEE Transactions on Image Processing* 13(8), 1147-1156 (2004). https://doi.org/10.1109/TIP.2004.828418

[8] Lee, C.-C., Wu, H.-C., Tsai, C.-S., Chu, Y.-P.: Adaptive lossless steganographic scheme with centralized difference expansion. *Pattern Recognition* 41(6), 2097-2106 (2008). https://doi.org/10.1016/j.patcog.2007.11.018

[9] K B, D., Chang, C.-C., Yang, H.-R., Wang, Z.-H.: Efficient pixel prediction algorithm for reversible data hiding. *Int. J. Netw. Secur.* 18, 750-757 (2016). https://doi.org/10.6633/IJNS.201607.18(4). 15

[10] Dragoi, I.-C., Coltuc, D.: Local-prediction-based difference expansion reversible watermarking. *IEEE Transactions on Image Processing* 23(4), 1779-1790 (2014). https://doi.org/10.1109/TIP.2014.2307482

[11] Hu, Y., Lee, H.-K., Li, J.: De-based reversible data hiding with improved overflow location map. *IEEE Transactions on Circuits and Systems for Video Technology* 19(2), 250-260 (2009). https://doi.org/10.1109/ TCSVT.2008.2009252

[12] Liu, M., Seah, H.S., Zhu, C., Lin, W., Tian, F.: Reducing location map in prediction-based difference expansion for reversible image data embedding. *Signal Processing* 92(3), 819-828 (2012). https://doi.org/10.1016/ j.sigpro.2011.09.028

[13] Ni, Z., Shi, Y.-Q., Ansari, N., Su, W.: Reversible data hiding. *IEEE Transactions on Circuits and Systems for Video Technology* 16(3), 354-362 (2006). https://doi.org/10.1109/TCSVT.2006.869964

[14] Tsai, P., Hu, Y.-C., Yeh, H.-L.: Reversible image hiding scheme using predictive coding and histogram shifting. *Signal Process.* 89(6), 11291143 (2009). https://doi.org/10.1016/j.sigpro.2008.12.017

[15] Kukreja, S., Kasana, S.S., Kasana, G.: Histogram based multilevel reversible data hiding scheme using simple and absolute difference images. *Multimedia Tools and Applications* 78, 6139-6162 (2019), https://link.springer.com/article/10.1007/s11042-018-6169-0

[16] Jia, Y., Yin, Z., Zhang, X., Luo, Y.: Reversible data hiding based on reducing invalid shifting of pixels in histogram shifting. *Signal Processing* 163, 238-246 (2019). https://doi.org/10.1016/j.sigpro.2019.05.020

[17] Li, X., Yang, B., Zeng, T.: Efficient reversible watermarking based on adaptive prediction-error expansion and pixel selection. *IEEE Transactions on Image Processing* 20(12), 3524-3533 (2011). https://doi.org/10. 1109/TIP.2011.2150233

[18] Wang, X., Ding, J., Pei, Q.: A novel reversible image data hiding scheme based on pixel value ordering and dynamic pixel block partition. *Information Sciences* 310, 16-35 (2015). https://doi.org/10.1016/j.ins.2015.03. 022

[19] Weng, S., Shi, Y., Hong, W., Yao, Y.: Dynamic improved pixel value ordering reversible data hiding. *Information Sciences* 489, 136-154 (2019). https://doi.org/10.1016/j.ins.2019.03.032

[20] Li, X., Li, J., Li, B., Yang, B.: High-fidelity reversible data hiding scheme based on pixel-value-ordering and prediction-error expansion. *Signal Processing* 93(1), 198-205 (2013). https://doi.org/10.1016/j.sigpro.2012.07.025

[21] Ou, B., Li, X., Zhao, Y., Ni, R.: Reversible data hiding using invariant pixel-value-ordering and prediction-error expansion. *Signal processing: image communication* 29(7), 760-772 (2014). https://doi.org/10.1016/j.image.2014.05.003

[22] Peng, F., Li, X., Yang, B.: Improved pvo-based reversible data hiding. *Digital Signal Processing* 25, 255-265 (2014). https://doi.org/10.1016/j. dsp.2013.11.002

[23] Li, J., Wu, Y.-H., Lee, C.-F., Chang, C.-C.: Generalized pvo-k embedding technique for reversible data hiding. *Int. J. Netw. Secur.* 20(1), 65-77 (2018). https://doi.org/10.6633/IJNS.201801.20(1).08

[24] Sao, N.K., Hoa, N.N., At, P.V.: An effective reversible data hiding method based on pixel-value-ordering. *Journal of Computer Science and Cybernetics* 36(2), 139-158 (2020). https://doi.org/10.15625/1813-9663/36/2/ 14084

[25] Weng, S., Pan, J.-s., Li, L.: Reversible data hiding based on an adaptive pixel-embedding strategy and two-layer embedding. *Information Sciences* 369, 144-159 (2016). https://doi.org/10.1016/j.ins.2016.05.030

[26] Ou, B., Li, X., Li, W., Shi, Y.-Q.: Pixel-value-ordering based reversible data hiding with adaptive texture classification and modification. *In: Digital Forensics and Watermarking*, pp. 169-179. Springer, Cham (2019), DOI: 10.1007/978-3-030-11389-6_13

[27] Weng, S., Shi, Y., Hong, W., Yao, Y.: Dynamic improved pixel value ordering reversible data hiding. *Information Sciences* 489, 136-154 (2019). https://doi.org/10.1016/j.ins.2019.03.032

[28] Pan, Z., Gao, E.: Reversible data hiding based on novel embedding structure pvo and adaptive block-merging strategy. *Multimedia Tools and Applications* 78, 26047-26071 (2019), https://link.springer.com/article/10.1007/s11042-019-7692-3

[29] Kumar, R., Kumar, N., Jung, K.-H.: I-pvo based high capacity reversible data hiding using bin reservation strategy. *Multimedia Tools and Applications* 79(31-32), 22635-22651 (2020), https://link.springer.com/article/10.1007/s11042-020-09069-0

[30] Qu, X., Kim, H.J.: Pixel-based pixel value ordering predictor for highfidelity reversible data hiding. *Signal Processing* 111, 249-260 (2015). https://doi.org/10.1016/j.sigpro.2015.01.002

[31] Weng, S., Pan, J.-S., Jiehang, D., Zhou, Z.: Pairwise ipvo-based reversible data hiding. *Multimedia Tools and Applications* 77, 13419-13444 (2018), https://link.springer.com/article/10.1007/s11042-017-4959-4

[32] Nguyen, N.-H., Pham, V.-A.: An efficient ipvo based reversible data hiding method using four pixel-pairs. *Multimedia Tools and Applications*, 1-30 (2023), https://link.springer.com/article/10.1007/s11042-023-14669-7

[33] Chang, C.-C., Kieu, T.D., Chou, Y.-C.: Reversible data hiding scheme using two steganographic images. *In: TENCON 2007 - 2007 IEEE Region 10 Conference*, pp. 1-4 (2007). https://doi.org/10.1109/TENCON.2007. 4483783

[34] Chang, C.-C., Chou, Y.-C., Kieu, T.D.: Information hiding in dual images with reversibility. *In: 2009 Third International Conference on Multimedia and Ubiquitous Engineering*, pp. 145-152 (2009). https://doi.org/10. 1109/MUE.2009.35

[35] Chang, C.-C., Lu, T.-C., Horng, G., Huang, Y.-H., Hsu, Y.-M.: A high payload data embedding scheme using dual stego-images with reversibility. *In: 2013 9th International Conference on Information, Communications & Signal Processing*, pp. 1-5 (2013). https://doi.org/10.1109/ICICS. 2013.6782790

[36] Qin, C., Chang, C.-C., Hsu, T.-J.: Reversible data hiding scheme based on exploiting modification direction with two steganographic images. *Multimedia Tools and Applications* 74, 5861-5872 (2015), https://link.springer.com/article/10.1007/s11042-014-1894-5

[37] Lu, T.-C., Tseng, C.-Y., Wu, J.-H.: Dual imaging-based reversible hiding technique using lsb matching. *Signal Processing* 108, 77-89 (2015). https: //doi.org/10.1016/j.sigpro.2014.08.022

[38] Lu, T.-C., Wu, J.-H., Huang, C.-C.: Dual-image-based reversible data hiding method using center folding strategy. *Signal Processing* 115,195213 (2015). https://doi.org/10.1016/j.sigpro.2015.03.017

[39] Yao, H., Qin, C., Tang, Z., Tian, Y.: Improved dual-image reversible data hiding method using the selection strategy of shiftable pixels' coordinates with minimum distortion. *Signal Processing* 135, 26-35 (2017). https://doi.org/10.1016/j.sigpro.2016.12.029

[40] Kim, P.-H., Ryu, K.-W., Jung, K.-H.: Reversible data hiding scheme based on pixel-value differencing in dual images. *International Journal of Distributed Sensor Networks* 16(7), 1550147720911006 (2020). https://doi.org/10.1177/1550147720911006

[41] Chen, X., Hong, C.: An efficient dual-image reversible data hiding scheme based on exploiting modification direction. *Journal of Information Security and Applications* 58, 102702 (2021). https://doi.org/10.1016/j.jisa. 2020.102702

[42] Luyen, C.T., Sao, N.K., At, P.V.: Dual-image reversible data hiding method using maximum embedding ability of each pixel. *Journal of Information Security and Applications* 74, 103470 (2023). https://doi.org/10.1016/j.jisa.2023.103470

[43] Niu, Y., Shen, S.: A novel pixel value ordering reversible data hiding based on dual-image. *Multimedia Tools and Applications* 81(10), 13751-13771 (2022), https://link.springer.com/article/10.1007/s11042-022-12149-y