Journal of Information Hiding and Multimedia Signal Processing ©2022 ISSN 2073-4212
Ubiquitous International Volume **13**, Number **1**, March **2022**

Efficient reversible data hiding using block histogram shifting with invariant peak points

Nguyen Kim Sao, Cao Thi Luyen and Pham Van At

Department of Information Technology
University of Transport and Communications
No.3 Cau Giay Street, Lang Thuong ward, Dong Da District, Hanoi, Vietnam
saonkoliver@utc.edu.vn, luyenct@utc.edu.vn and phamvanat@utc.edu.vn

Received June 2021; revised December 2021

Abstract. *In the pixel value ordering (PVO) based reversible data hiding method, there are only two pixels in a block are used for embedding data. It limits the data embedding capacity (payload) of the PVO method. To overcome this drawback, lately, Weng et al. proposed new development, in which all pixels in a block, except for one or two in middle positions, can be involved in embedding data, so this method has a large embedding capacity. This paper proposed a new method (HistC) using histogram shifting on blocks, in which selected peak points are pixels at the center of the histogram distribution. These peak points are invariant in the embedding process, so they do not need to be recorded for extracting stage. Because in natural images, many pixels usually appear at the center of the histogram, so our proposed method has a high payload. Moreover, the second method (HistE) in which peak points are selected at the edges of the histogram distribution is also presented. Experimental results show that both proposed methods have a large payload and high stego image quality. Especially, the data embedding capacity of the HistC method is higher than Weng et al.'s method and other PVO-based methods.*
Keywords: PVO, data hiding, histogram shifting

1. **Introduction.** Nowadays, digital documents are usually transmitted on the Internet, they can be risked by various types of attacks. So protecting the integrity of digital documents is an urgent problem in the field of information security. Two main approaches for protecting digital documents are encryption and data hiding. Data hiding is the technology in which secret data, such as authentication or copyright information is embedded in a digital image. Traditional data hiding techniques [5, 25, 28, 32] usually cause permanent distortions in the digital image, and this limits their application in some sensitive fields, such as medical image processing and military communication. So new data hiding methods called reversible (or lossless) data hiding are proposed to overcome this limitation. Reversible data hiding (RDH) not only can extract hidden data but also restore the original image.

The first RDH proposed by [21] is performed based on modulo addition 256. The disadvantage of this method is that the stego image quality is very low. Shortly, some RDH methods based on lossless compression [6, 7, 11] are developed. These methods acquire embedding space through lossless compression of a specific part of the digital original image. As a result, they usually have low embedding capacity.

In 2003, Tian proposed an important reversible data hiding algorithm called "difference expansion" (DE) [35], it overcomes the limitation on the embedding capacity of the above methods. In the DE method, the difference of each pixel pair (x, y) is calculated $h = x - y$,

then h is expanded to the left and one bit is inserted in the right. In fact, a bit b is embedded in h as follows $h' = 2h + b$. After having h', a stego pixel pair (x', y') is determined:

$$x' = l + \left\lfloor \frac{h'+1}{2} \right\rfloor,\ y' = l - \left\lfloor \frac{h'}{2} \right\rfloor,\ where\ l = \left\lfloor \frac{x+y}{2} \right\rfloor.$$

If (x', y') pair is still in pixel domain ([0, 255] for grayscale image) then embedding is successful, and (x, y) is called "expandable". In DE-method, a binary map is established to mark expandable pixel pairs, it is called a location map. Since the compression code of the location map must be embedded in the original image along with the data, so it reduces the embedding capacity. An important improvement of DE method is to consider pixel vectors (PV) instead of pixel pairs [1, 2, 3, 15, 17, 36]. In each n-sized pixel vector, a fixed pixel (basic pixel) is selected, and (n-1) differences are computed by subtracting the basic pixel from other pixels. Thus the number of differences obtained in the PV method is increased almost twice than the DE-method. Moreover, the location map is constructed in blocks so its size is decreased significantly. Two above advantages make remarkably increasing the embedding capacity of the PV method in comparison with the DE-method. To enlarge the embedding capacity of the DE-algorithm, other improved methods try to decrease the size of the location map [13, 14, 20, 29]. The next important development of the DE-method is prediction error expansion methods (PEE) [4, 33, 34] in which each pixel is predicted by a predictor, and a difference between pixel and its prediction value is computed. Each difference is expanded for embedding one bit. Because the absolute value of prediction errors often is smaller than one of pixel pair differences, and the number of prediction errors is almost twice the number of pixel pair differences, so the embedding capacity of PEE methods is significantly larger than DE.

In addition to the DE technique, Ni et al. [22] proposed another important RDH method called histogram shifting (HS). Two bins (denoted PeakL and PeakR) at which histogram reaches largest values are selected, pixels are shifted to the left and the right sides of $PeakL$ and $PeakR$, respectively, such that bins $(PeakL - 1)$ and $(PeakR + 1)$ become empty, finally data bits are embedded at pixels having a value of $PeakL$ or $PeakR$. In HS method, each pixel must be modified at most one, so this method has a good stego image quality, however, its capacity is not very high. To overcome this drawback, prediction error histogram shifting methods (PEH) are proposed [8, 9, 10, 16, 24, 31, 39], in which prediction errors of pixels are evaluated, then HS method is applied for prediction errors. In comparison with the HS method, PEH methods have significantly higher embedding capacity.

Recently, Li et al. [19] proposed a new RDH method based on pixel value ordering (PVO), in which, the pixel values of each block are sorted in ascending order. Then the largest pixel value is predicted by the second-largest pixel, and the smallest pixel is predicted by the second-smallest pixel. The prediction errors $dmax \geq 0$ and $dmin \leq 0$ at the largest and the smallest side, respectively, are calculated. When $dmax = 1$, a bit is embedded by increasing the largest pixel by 1 or keeping it unchanged depending on whether the to-be-embedded bit is 1 or 0.

Prediction error $dmin$ is treated similarity, but the smallest pixel is decreased by one. Thus, after embedding, PVO is unchanged, it guarantees reversibility.

It is noted that since the local pixel values are correlative, so there are many prediction errors of 0, which are ignored by the PVO method. Peng et al. [27] proposed an improved PVO method, called IPVO, in which a bit will be embedded when errors $dmax$ or $dmin$ equal 0. So the IPVO method has an embedding capacity higher than the PVO method. In [12], He et al. commented that the data embedding efficiency of the IPVO method in a

block of pixels is highly dependent on the sequence of pixels obtained from this block. So, unlike in [27] where only a fixed transformation is applied, He et al. use for each different block of pixels a different way to convert the block into a sequence of pixels with as many inversion pairs as possible. By this improvement, the embedding capacity of He et al.'s method is greater than that of the IPVO method. Another method, known as PVO-K, proposed in [23] is also an improvement of PVO. The PVO-K method considers all largest pixels or all smallest pixels as a unit to embed data. That means, for embedding a bit, increase all largest pixels by 1 or keep them unchanged depending on the to-be-embedded bit is 1 or 0. The smallest pixels are treated similarly.

GEPVO-K [18] is a remarkable development of PVO-K, it can embed k bits in all k largest pixels and l bit in all l smallest pixels of a block. However this method has two drawbacks: after embedding, pixels can be altered at most by two and each block is marked by two binary bits in the location map. These drawbacks limit both embedding capacity and stego image quality. The iGePVO-K method [30], an improvement of GePVO-K overcame the above drawbacks. The length of the location map in iGePVO-K is the same as in IPVO or PVO-K methods. After embedding by iGePVO-K, pixels are modified at most by one, so this method has high embedding capacity and good stego image quality.

In [26], it is noted that the maximum embedding capacity is highly dependent on the number of sub-blocks because the prediction errors for every sub-block are fixed by 2. For this purpose, the three-pixel embedding structure (TPES) is used in [26] to create more prediction errors by dividing the original image into 3-sized blocks. Thus, two pixels are used to embed data over three pixels, and the maximal embedding rate is increased to $\frac{2}{3} = 66.7\%$.

To enlarge the embedding capacity, the method proposed by [37] used the three largest pixels and three smallest pixels for embedding. Suppose a sorted n-sized block is $X_\sigma = (x_{\sigma(1)}, \ldots, x_{\sigma(n)})$. Method [37] used $x_{\sigma(n-3)}$ for predicting the three largest pixels and $x_{\sigma(4)}$ for predicting the three smallest pixels. Then three prediction errors at the largest side and three prediction errors at the smallest side are created for embedding at most six bits in every block.

Method [38] is a development of [37] to can embed more than six bits in a smooth block by selecting median pixel values for predicting remaining pixels. In fact, if n is even, two median pixels $x_{\sigma(\frac{n}{2})}, x_{\sigma(\frac{n}{2}+1)}$ are chosen, while when n odd, one median value $x_{\sigma(\frac{n+1}{2})}$ is chosen. From this, $(n-2)$ prediction errors when n is even, and $(n-1)$ prediction errors when n odd are created. Thus, at most $(n-1)$ bits can be embedded in a n-sized block. Perhaps, this method achieves the highest embedding capacity among the PVO-based methods. In [40], Zheng et al. applied the ideas of He et al. to Weng et al.'s method [37] to achieve higher embedding capacity.

In this paper, we will analyze the relation between the PVO of a block and the histogram of this block and show that can use histogram shifting in each block with two reasonably defined peak points to achieve high embedding capacity. Two methods are presented. The first method, called HistC (center histogram), selects two peak points at the center of the histogram distribution. While, the second method, called HistE (Edge histogram), selects two peak points at the Edge of the histogram. Each method is performed in two modes: Left mode and right mode. The first method has an embedding capacity larger than the second method. While the second method has a better stego image quality.

Experiment results show that the HistC method achieves the embedding capacity larger in comparison with all above mentioned PVO-based methods.

The rest of this paper is organized as follows. Section 2 presents some related works. The idea of the new methods is introduced in Section 3. In Section 4, two new methods

are proposed. Experimental results are presented in Section 5. Finally, the conclusion is made at the end of the paper.

2. **Related works.**

2.1. **IPVO.** Given a host 8-bit grayscale image I of size $R \times C$, it is divided into non-overlap $r \times c$-sized blocks. Each block is converted to a pixel sequence $X = (x_1, x_2, \ldots, x_n)$ where $n = r \times c$, then X is sorted in ascending order to get $X_\sigma = (x_{\sigma(1)}, \ldots, x_{\sigma(n)})$:

$x_{\sigma(1)} \leq \ldots \leq x_{\sigma(n)}$.

Moreover if $x_{\sigma(i)} = x_{\sigma(i+1)}$ then $\sigma(i) < \sigma(i+1)$. Value d_{max} is computed as:

$$d_{max} = x_u - x_v, \tag{1}$$

where

$$\begin{cases} u = min(\sigma(n), \sigma(n-1)), \\ v = max(\sigma(n), \sigma(n-1)). \end{cases} \tag{2}$$

Then $x_{\sigma(n)}$ is modified to $x'_{\sigma(n)}$ for embedding a bit b as follows:

$$x'_{\sigma(n)} = \begin{cases} x_{\sigma(n)} + b & if\ d_{max} = 0\ or\ d_{max} = 1, \\ x_{\sigma(n)} + 1 & if\ d_{max} < 0\ or\ d_{max} > 1. \end{cases} \tag{3}$$

It is noted that pixel value order of block is unchanged, so values $\sigma(1), \ldots, \sigma(n)$ and u, v are retained. Therefore, at the decoder side, extracting bit b and restoring $x_{\sigma(n)}$ can perform as follows:

$$d'_{max} = x'_u - x'_v, \tag{4}$$

$$b = \begin{cases} -d'_{max}, & if\ d'_{max} \in \{-1, 0\}, \\ d'_{max} - 1, & if\ d'_{max} \in \{1, 2\}, \end{cases} \tag{5}$$

and

$$x_{\sigma(n)} = \begin{cases} x'_{\sigma(n)} - b, & if\ d'_{max} \in \{-1, 0, 1, 2\}, \\ x'_{\sigma(n)} - 1, & otherwise. \end{cases} \tag{6}$$

Similarly, $x_{\sigma(1)}$ is altered to $x'_{\sigma(1)}$ to embeded a bit b according to formulas:

$$d_{min} = x_s - x_t, \tag{7}$$

$$\begin{cases} s = min(\sigma(1), \sigma(2)), \\ t = max(\sigma(1), \sigma(2)), \end{cases} \tag{8}$$

$$x'_{\sigma(1)} = \begin{cases} x_{\sigma(1)} - b, & if\ d_{min} = 0\ or\ d_{min} = 1, \\ x_{\sigma(1)} - 1, & if\ d_{min} < 0\ or\ d_{min} > 1. \end{cases} \tag{9}$$

Extracting hidden bit and restoring $x_{\sigma(1)}$ are carried out as follows:

$$d'_{min} = x'_s - x'_t, \tag{10}$$

$$b = \begin{cases} -d'_{min}, & if\ d'_{min} \in \{-1, 0\}, \\ d'_{min} - 1, & if\ d'_{min} \in \{1, 2\}, \end{cases} \tag{11}$$

$$x_{\sigma(1)} = \begin{cases} x'_{\sigma(1)} + b, & if\ d'_{min} \in \{-1, 0, 1, 2\}, \\ x'_{\sigma(1)} + 1, & otherwise. \end{cases} \tag{12}$$

To solve the over/underflow problem, the IPVO method uses a block-based location map. Each block is assigned a position number in the location map. The block containing at least a pixel value of 0 or 255, has a position number of 1, the other block's position number equals 0. The blocks with position number 1 will not be used for embedding data, because they can cause over/underflow.

Although, the IPVO method has a higher embedding capacity than the PVO method, it is still limited by the fact that it is only possible to embed at most 2 bits in each pixel block.

2.2. **GePVO-K.** In the PVO-K method [23], one bit is embedded in k largest pixels of an image block. GePVO-K [18] is a generation of PVO-K to embed k bits in k these pixels.

In the GePVO-K, each block $X = (x_1, x_2, \ldots, x_n)$ is divided into three types.

(a) Block having at least one pixel value of 0, 1, 254 or 255 will not be used for embedding data, because it may cause under/overflow. The position number of this block in the location map is set as 2 ($LM(X) = 2$).

(b) Block with all pixel values are equal (flat block): $x_1 = x_2 = \ldots = x_n = \alpha$ with α differs from 0, 1, 254, 255 will be used to embed $n - 1$ bits. The position number of the block is set as 1 ($LM(X) = 1$).

(c) Remaining blocks (rough blocks) will be used to embed data. The position number of blocks is set as 0 ($LM(X) = 0$).

It is noted that the position number of each block is two binary bits, so the location map is a binary sequence having the length of twice the number of blocks.

Next, the authors deal with each block depending on its position number on the map:

Case 1: If the position number of the block $LM(X) = 2$, the block is not used to embed data and it is skipped

Case 2: If the position number of block $LM(X) = 1$, i.e, all pixel values in the block are equal, keep the first pixel value unchanged and then embed the data in the remaining pixels as follows:

$$x'_i = \begin{cases} x_i, & if\ i = 1 \\ x_i + b_{i-1}, & if\ i = 2, 3, \ldots, n, \end{cases}$$

where $b_i, i = 1, \ldots, n-1$ are embedded bits.

Case 3: If the position number of the block $LM(X) = 0$, X is sorted in ascending order to get $(x_{\sigma(1)}, x_{\sigma(2)}, \ldots, x_{\sigma(n)})$ as in subsection 2.1. Assume sorted block has k largest pixels and l second largest pixels. The difference: $d_{max} = x_{\sigma(n-k+1)} - x_{\sigma(n-k)}$ is calculated and embedding is performed as follows:

Case 3.1: if $d_{max} > 1$: No data is embedded, all largest pixel values are increased by one.

Case 3.2: if $d_{max} = 1$, all largest and second largest pixel values are modified for embedding k bits as follows.

$$x'_{\sigma(i)} = \begin{cases} x_{\sigma(i} + 1, & if\ n - k - l + 1 \le i \le n - k \\ x_{\sigma(i)} + 1 + b_j, & if\ n - k + 1 \le i \le n;\ j = 1, \ldots, k. \end{cases}$$

Embedding data bits in the smallest pixel values is performed similarly.

2.3. **Weng, Pan et al.'s method (WengPan method).** Like the IPVO method, in Weng, Pan et al.'s method [37], each block $(x_1, x_2, \ldots, x_n)$ is sorted in assending order to get $(x_{\sigma(1)}, \ldots, x_{\sigma(n)})$. Then to improve the embedding capacity, Weng, Pan, et al. consider three pixel pairs on the right side $(x_{\sigma(n-3)}, x_{\sigma(n-2)})$, $(x_{\sigma(n-3)}, x_{\sigma(n-1)})$, $(x_{\sigma(n-3)}, x_{\sigma(n)})$, and three pixel pairs on the left side $(x_{\sigma(1)}, x_{\sigma(4)})$, $(x_{\sigma(2)}, x_{\sigma(4)})$, $(x_{\sigma(3)}, x_{\sigma(4)})$. The pixels

$(x_{\sigma(4)}, \ldots, x_{\sigma(n-3)})$ are kept unchanged, while six pixels $x_{\sigma(n-2)}$, $x_{\sigma(n-1)}$, $x_{\sigma(n)}$, $x_{\sigma(1)}, x_{\sigma(2)}$, $x_{\sigma(3)}$ will be used for embedding. In fact, each pair $(x_{\sigma(n-3)}, x_{\sigma(i)}), i = n-2, n-1, n$ will be used for embedding according to formulas (1-3). Extracting data bits and restoring pixels $x_{\sigma(i)}$ are performed as in (4-6). Similarly, each pair $(x_{\sigma(i)}, x_{\sigma(4)}), i = 1, 2, 3$ are used for embedding as in formulas (7-9). Extracting data bits and restoring pixels $x_{\sigma(i)}, i = 1, 2, 3$ are carried out as in formulas (10-12). In general, for flat blocks, this method provides high embedding capacity.

2.4. **Weng, Shi et al.'s method (WengShi method).** Weng and Shi et al.[38] noted that for high-correlation blocks, more than six pixels can be used to carry data. In fact, X is sorted in ascending order to get $X_\sigma = (x_{\sigma(1)}, \ldots, x_{\sigma(n)})$. Two median positions m_1 and m_2 in X_σ are defined as follows: when n is even, $m_1 = \frac{n}{2}, m_2 = m_1 + 1$. In contrast, when n is odd, $m_1 = m_2 = \frac{n+1}{2}$ The values $x_{\sigma(m_1)}$ and $x_{\sigma(m_2)}$ are called median values of X_σ. Then data bits are embedded in pixel pairs $(x_{\sigma(m_2)}, x_{\sigma(m_2+1)}), \ldots, (x_{\sigma(m_2)}, x_{\sigma(n)})$ as in formulas (1-3) and embedded in pixel pairs $(x_{\sigma(1)}, x_{\sigma(m_1)}), \ldots, (x_{\sigma(m_1-1)}, x_{\sigma(m_1)})$ according to formulas (7-9). Thus, when n is odd, (n-1) bits can be embedded and when n is even, $(n-2)$ bits can be embedded. It is noted that if $n \leq 8$ then two methods: WengPan and WengShi method are the same. In general, WengShi method has an embedding capacity higher than WengPan method. Both these methods also use the location map similar to the one in the method IPVO to solve the over/underflow.

3. Idea of proposed methods.

3.1. **Histogram of the pixel block.** Let $X = (x_1, x_2, \ldots, x_n)$ be a n-sized pixel block, then histogram of X, denoted h, is a function defined in segment $[mi, ma]$ where mi and ma are the minimal and the maximal values of X, respectively. Each x belongs to $[mi, ma]$ called a bin and $h(x)$ equals the number of times that pixels having a value of x appear in block X. For example for a block:

$$X = \begin{bmatrix} 25 & 24 & 25 \\ 26 & 27 & 28 \\ 26 & 25 & 26 \end{bmatrix}$$

histogram $h(x)$ is as in Fig.1. It is noted that the shape of the histogram for natural image

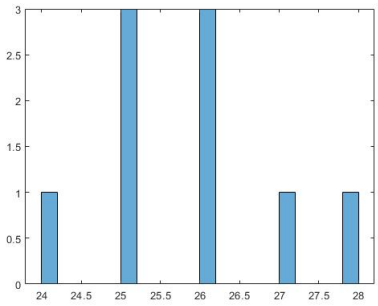

FIGURE 1. Histogram of X in section 3.1

blocks is usually similar to Fig 1: Histogram achieves its maximal value at the center of segment $[mi, ma]$ (the center of the histogram). Then the values of the histogram are decreased in both directions: the left and the right of $[mi, ma]$.

Below it is shown that the histogram shifting method with reason defined peak-points can provide embedding capacity higher than PVO-based reversible data hiding methods.

3.2. **Remark on GePVO-K method.** Consider a n-sized block $X = (x_1, \ldots, x_n)$ which has at least four distinct pixel values. Let k be the number of the largest pixels and l be the number of smallest pixels, d_{max} be the difference between largest and second-largest pixels. Then embedding capacity of the GePVO-K method in the largest pixels denoted by CL is calculated as:

$$CL = \begin{cases} k, & if\ d_{max} = 1 \\ 0, & if\ d_{max} > 1 \end{cases}$$

In other words:

$$CL = h(ma) * sign(h(ma - 1))$$

where $ma = max(X)$,

$$sign(z) = \begin{cases} 1, & if\ z > 0 \\ -1, & if\ z < 0 \\ 0, & if\ z = 0 \end{cases}$$

Similarly, the embedding capacity of the GePVO-K method in the smallest pixels denoted CS can be formulated as:

$$CS = h(mi) * sign(h(mi + 1)).$$

From the property of histogram presented in subsection 3.1, it is very likely that

$$CL + CS \leq h(ma - 1) + h(mi + 1).$$

From this, it follows that the histogram shift method with the left $peak = mi + 1$ and right $peak = ma - 1$ (HistE method) can reach the embedding capacity higher than the GePVO-K method.

Example: Four following 3×3 blocks are randomly extracted from Cabeza and Lena images. The embedding capacity of GePVO-K and HistE methods in these blocks is shown in Table 1.

80	82	83
79	79	80
80	80	80

Cabeza block 1

68	70	70
68	68	69
69	71	69

Cabeza block 2

24	25	27
25	25	24
25	25	25

Lena block 1

68	67	70
70	70	73
68	70	71

Lena block 2

FIGURE 2. Randomly extracted 3×3 blocks

TABLE 1. Capacity comparison between GePVO-K and HistE

Block	Sorted pixel sequence of block	GePVO-K	HistE
Cabeza block 1	79 79 80 80 80 80 80 82 83	2+1=3	5+1=6
Cabeza block 2	68 68 68 69 69 69 70 70 71	3+1=4	3+2=5
Lena block 1	24 24 25 25 25 25 25 25 27	2+0=2	6+0=6
Lena block 2	67 68 68 70 70 70 70 71 73	1+0=1	2+0=2

TABLE 2. Capacity comparison between Wengshi and HistC

Block	Sorted pixel sequence of block	WengShi	HistC
Cabeza block 3	X_σ : 67 67 68 68 69 69 69 69 70 σ : 1 7 4 5 **2** 3 6 9 8	$m_1 = m_2 = 5$ $\sigma(m_1) = \sigma(m_2) = 2$ $x_{\sigma(m_1)} = x_{\sigma(m_2)} = 69$ $2 + 3 = 5$	$LPP = \lfloor \frac{67+70}{2} \rfloor = 68$ $RPP = 69$ $2 + 4 = 6$
Cabeza block 4	X_σ : 24 25 26 28 28 28 32 32 33 σ : 2 3 1 5 **6** 8 7 9 4	$m_1 = m_2 = 5$ $\sigma(m_1) = \sigma(m_2) = 6$ $x_{\sigma(m_1)} = x_{\sigma(m_2)} = 28$ $1 + 1 = 2$	$LPP = \lfloor \frac{24+33}{2} \rfloor = 28$ $RPP = 29$ $3 + 0 = 3$
Lena block 3	X_σ : 95 97 97 99 100 101 101 102 103 105 107 108 σ : 2 4 7 5 1 **6 8** 10 3 11 9 12	$m_1 = 6,\ m_2 = 7$ $\sigma(m_1) = 6,\ \sigma(m_2) = 8$ $x_{\sigma(m_1)} = x_{\sigma(m_2)} = 101$ $0 + 0 = 0$	$LPP = \lfloor \frac{95+108}{2} \rfloor = 101$ $RPP = 102$ $2 + 1 = 3$
Lena block 4	X_σ : 119 120 121 121 121 123 123 123 123 124 124 124 σ : 4 6 1 9 10 **2 3** 5 12 7 8 11	$m_1 = 6,\ m_2 = 7$ $\sigma(m_1) = 2,\ \sigma(m_2) = 3$ $x_{\sigma(m_1)} = x_{\sigma(m_2)} = 123$ $0 + 2 = 2$	$LPP = \lfloor \frac{119+124}{2} \rfloor = 121$ $RPP = 122$ $3 + 0 = 3$

3.3. **Remark on WengShi method.** WengShi method selects a left median pixel $x_{\sigma(m1)}$ and a right median pixel $x_{\sigma(m2)}$ of block X. Then uses all pairs $(x_{\sigma(i)}, x_{\sigma(m_1)}), 1 \le i \le m_1 - 1$ and $(x_{\sigma(m_2)}, x_{\sigma(i)}), m_2 + 1 \le i \le n$ for embedding data. So the embedding capacity of this method is very large. Now consider histogram shifting in block X with peak points are selected at the center of the segment $[mi, ma]$ (HistC method). In other words, values LPP (Left Peak Point) and RPP (Right Peak Point) are defined as

$$LPP = \left\lfloor \frac{mi + ma}{2} \right\rfloor, RPP = LPP + 1,$$

where $mi = min(X), ma = max(X)$. According to the remark in the subsection 3.1, the values $h(LPP)$ and $h(RPP)$ are usually greater than or equal to the values $h(x_{\sigma(m_1))}$ and $h(x_{\sigma(m_2))}$, where $h(x)$ is the histogram of the block X. From this, it follows that most likely the embedding capacity of HistC method is greater one of WengShi method. Below present some examples for demonstrating the above remarks.

Example: Four following blocks are randomly extracted from Cabeza and Lena images. The embedding capacity of WengShi and HistC methods in these blocks is shown in Table 2

67	69	69
68	68	69
67	70	69

Cabeza block 3

26	24	25
33	28	28
32	28	32

Cabeza block 4

100	95	103	97
99	101	97	101
107	102	105	108

Lena block 3

121	123	123	119
123	120	124	124
121	121	124	123

Lena block 4

FIGURE 3. Randomly extracted 3×3 and 3×4 blocks

4. **Proposed methods.** In this section, two proposed methods are presented. Both methods are performed according to the histogram shifting scheme on pixel blocks, but selecting peak points is different. The first method is called HistCenter (HistC) because its two peak points are at the center of the histogram distribution. While two peak points of the second method are at the two edges of the histogram, so it is called HistEdge (HistE). Each method is performed in two modes and what mode that gives higher embedding capacity will be selected.

HistC method is presented in subsections from Subsection 4.1 to Subsection 4.5, while HistE method will be considered briefly in Subsection 4.6.

4.1. **Embedding algorithm in a block.** In each of the 4 cases below pixels of a block will be scanned in rows from the top to bottom and left to right. Consider an image block of size $r \times c$

$$X = \begin{bmatrix} x_{11} & \dots & x_{1c} \\ \dots & \dots & \dots \\ x_{r1} & \dots & x_{rc} \end{bmatrix}$$

First, let $mi = min(X)$ and $ma = max(X)$, establish the histogram h of X: $h(x)$ with $mi \le x \le ma$. If denote $s = ma - mi + 1$, then histogram h consists of s bins.

Depending on value s the following cases are considered:

Case 1. If $s = 1$, i.e. histogram consists of only one bin, X is flat (all pixels of X are equal). The embedding is done as follows: The first pixel x_{11} is unchanged, $(r \times c - 1)$ remaining pixels are modified to embed $(r \times c - 1)$ bits by the formula:

$$x'_{ij} = \begin{cases} x_{11} + b_t, & if\ x_{11} \le 254 \\ x_{11} - b_t, & if\ x_{11} = 255, \end{cases}$$

with $(ij) \ne (1,1), i = 1, \dots, r, j = 1 \dots, c, t = 1, \dots, r \times c - 1$. It is noted that when $s = 1$, histogram h' of X' will consist of one or two bins.

Case 2: If $s = 2$, i.e. histogram consists of two bins: mi and ma. Denote $h1 = h(mi)$ and $h2 = h(ma)$. We shift bin mi to the left and bin ma to the right, that means pixels with a value of mi are decreased by 1 and the pixels with a value of ma are increased by 1. Then histogram consists of 4 bins that are $mi - 1, mi, ma$ and $ma + 1$, where mi and ma are empty bins.

Embed $(h_1 - 1)$ bits at bin $(mi - 1)$ and $(h_2 - 1)$ bits at bin $(ma + 1)$ as follows

$$x'_{ij} = \begin{cases} x_{ij} + b, & if\ x_{ij} = (mi - 1)\ and\ (i,j) <> (p,q) \\ x_{ij} - b, & if\ x_{ij} = (ma + 1)\ and\ (i,j) <> (u,v) \\ x_{ij}, & if\ otherwise, \end{cases}$$

where (p, q) is the position of the last pixel among pixels with value $(mi - 1)$, (u, v) is the position of the last pixel among pixels with value $(ma + 1)$, b is a to-be-embedded bit.

Case 3. If $s = 3$, i.e. histogram h consist of three bins: $mi, mi + 1, ma$. Denote $h1 = h(mi)$, $h2 = h(mi+1)$, $h3 = h(ma)$, shift bin mi to the left and bin ma to the right. Then histogram has five bins: $mi - 1, mi, mi + 1, ma$ and $ma + 1$. Here, the algorithm is performed in two modes: left and right.

Left mode: Embed $h_1 - 1$ bits at bin $mi - 1$ and h_2 bits at bin $mi + 1$ by the formula:

$$x'_{ij} = \begin{cases} x_{ij} + b, & if\ x_{ij} = (mi - 1)\ and\ (i,j) <> (p,q) \\ x_{ij} + b, & if\ x_{ij} = (mi + 1), \\ x_{ij}, & if\ otherwise \end{cases}$$

where (p, q) and b have the same meaning as in case 2.

Right mode: Embed h_2 bits at bin $mi+1$ and h_3-1 bits at bin $ma+1$ as follows:

$$x'_{ij} = \begin{cases} x_{ij}-b, & if\ x_{ij} = (ma+1)\ and\ (i,j) <> (u,v), \\ x_{ij}-b, & if\ x_{ij} = (mi+1) \\ x_{ij}, & if\ otherwise, \end{cases}$$

where (u,v) and b have the same meaning as in case 2.

Case 4: $s \geq 4$. First, determine LPP (Left Peak Point) and RPP (Right Peak Point) depending on s is even or odd and left mode or right mode as follows

$$LPP = \begin{cases} \frac{mi+ma-1}{2}, & if\ s\ is\ even, \\ \frac{mi+ma}{2}-1, & if\ s\ is\ odd\ and\ left\ mode, \\ \frac{mi+ma}{2}, & if\ s\ is\ odd\ and\ right\ mode, \end{cases} \tag{13}$$

$$RPP = LPP + 1. \tag{14}$$

Then $[h(LPP)+h(RPP)]$ bits are embedded at bins LPP and RPP as follows

$$x'_{ij} = \begin{cases} x_{ij}-1, & if\ x_{ij} < LPP, \\ x_{ij}+1, & if\ x_{ij} > RPP, \\ x_{ij}-b, & if\ x_{ij} = LPP, \\ x_{ij}+b, & if\ x_{ij} = RPP, \end{cases} \tag{15}$$

where b is a to-be-embedded bit.

4.2. **Examples for illustrating embedding algorithm in a block.** Input: an original block X, an array of embed bits b. Output: a stego block X'.

Case 1: $s = 1$

$$X = \begin{bmatrix} 255 & 255 & 255 \\ 255 & 255 & 255 \end{bmatrix};\ b = (1,0,0,1,0)\ \Rightarrow$$

$$X' = \begin{bmatrix} 255 & 254 & 255 \\ 255 & 254 & 255 \end{bmatrix}$$

$$X = \begin{bmatrix} 250 & 250 & 250 \\ 250 & 250 & 250 \end{bmatrix};\ b = (1,0,0,1,0)\ \Rightarrow$$

$$X' = \begin{bmatrix} 250 & 251 & 250 \\ 250 & 251 & 250 \end{bmatrix}$$

Case 2: $s = 2$

$$X = \begin{bmatrix} 41 & 40 & 40 \\ 40 & 40 & 41 \end{bmatrix};\ b = (1,0,0,1).$$

$$h(40) = 4,\ h(41) = 2;\ h1 = 4,\ h2 = 2,\ (h1+h2)-2 = 4;$$

Histogram shifting:

$$\Rightarrow X = \begin{bmatrix} 42 & 39 & 39 \\ 39 & 39 & 42 \end{bmatrix}$$

For this block, we have $(p,q) = (2,2)$ and $(u,v) = (2,3)$. Embedding 3 bits on bin 39 and 1 bit on bin 42:

$$X' = \begin{bmatrix} 41 & 39 & 39 \\ 40 & 39 & 42 \end{bmatrix}$$

Case 3: $s = 3$

$$X = \begin{bmatrix} 40 & 42 & 40 \\ 42 & 41 & 40 \end{bmatrix}$$

Histogram shifting:

$$\Rightarrow X = \begin{bmatrix} 39 & 43 & 39 \\ 43 & 41 & 39 \end{bmatrix} \tag{16}$$

$h(39) = 3,\ h(40) = 0,\ h(41) = 1,\ h(42) = 0,\ h(43) = 2;\ (p, q) = (2, 3);\ (u, v) = (2, 1).$

Left mode: Embed $(3 - 1)$ bits on bin 39 and 1 bit on bin 41, suppose $b = (1, 0, 1)$, then from (16) have

$$X' = \begin{bmatrix} 40 & 43 & 39 \\ 43 & 42 & 39 \end{bmatrix}$$

Right mode: Embed 1 bit on bin 41 and $(2 - 1)$ bit on bin 43, suppose $b = (1, 0)$ then from (16)

$$X' = \begin{bmatrix} 39 & 42 & 39 \\ 43 & 41 & 39 \end{bmatrix}$$

Case 4: $s \geq 4, s = 5$:

$$X = \begin{bmatrix} 41 & 40 & 43 \\ 42 & 41 & 41 \\ 41 & 44 & 43 \end{bmatrix};\ b = (1, 0, 1, 0, 1). \tag{17}$$

$mi = 40,\ ma = 44,\ s = ma - mi + 1 = 5$ is a odd number.
Left mode: From (13), (14), have

$$LPP = \frac{mi + ma}{2} - 1 = 41,\ RPP = 42$$

Amount of embedded bits: $h(LPP) + h(RPP) = 5,\ b = (1, 0, 1, 0, 1)$.
Embedding b into X (see (17)) at bin 41 and 42 by formula (15), obtain

$$X' = \begin{bmatrix} 40 & 39 & 44 \\ 42 & 40 & 41 \\ 40 & 45 & 44 \end{bmatrix}$$

Right mode: From (13), (14), have

$$LPP = \frac{mi + ma}{2} = 42,\ RPP = 43$$

Amount of embedded bits: $h(LPP) + h(RPP) = 3,\ b = (1, 0, 1)$.
Embedding b into X (see (17)) at bin 42 and 43 according to (15), obtain

$$X' = \begin{bmatrix} 40 & 39 & 44 \\ 42 & 40 & 40 \\ 40 & 45 & 44 \end{bmatrix}$$

4.3. **Extracting and restoring algorithm in a block.** Given an $r \times c$-sized stego block X' which is obtained from algorithm 4.1. We need to extract data bits and restore the original block X.

First, establish the histogram h' of X': $h'(x)$ with $mi' \leq x \leq ma', mi' = min(X'), ma' = max(X')$. If denote $s' = ma' - mi' + 1$ then histogram h' consists of s' bins. Depending on value s', the algorithm is performed in the following cases.

Case 1: $s' \leq 2$ then according to subsection 4.1, $s = 1$ and block X is flat. Extracting bits b_t and restoring X are done as follows:

$$b_t = \begin{cases} x'_{ij} - x'_{11}, & if\ x'_{11} \leq 254 \\ x'_{11} - x'_{ij}, & if\ x'_{11} = 255 \end{cases}$$

$x_{ij} = x'_{11}, i = 1, \ldots, r, j = 1, \ldots, c, t = 1, \ldots, r * c - 1.$

Case 2: $s' = 4$ i.e. the histogram h' consist of 4 bins: $mi', mi' + 1, ma' - 1, ma'$. In this case, $s = 2$, set $h1 = h'(mi') + h'(mi' + 1)$ and $h2 = h'(ma' - 1) + h'(ma')$.

Denote X'_1 as the set of $(h_1 - 1)$ first pixels $x'_{ij} \in \{mi', mi' + 1\}$ and X'_2 as the set of $(h_2 - 1)$ first pixels $x'_{ij} \in \{ma' - 1, ma'\}$. Then extracting $h1 + h2 - 2$ bits is done as follows.

Scan all pixels in the union $X_1 \cup X_2$. For each $x'_{ij} \in X_1 \cup X_2$, a bit b is extracted as:

$$b = \begin{cases} 0, & if\ x'_{ij} = mi'\ or\ x'_{ij} = ma' \\ 1, & if\ x'_{ij} = mi' + 1,\ or\ x'_{ij} = ma' - 1. \end{cases}$$

In this case X consists of only two distinct pixel values and they are returned as follows:

$$x_{ij} = \begin{cases} mi' + 1, & if\ x'_{ij} = mi'\ or\ x'_{ij} = mi' + 1 \\ ma' - 1, & if\ x'_{ij} = ma',\ or\ x'_{ij} = ma' - 1. \end{cases}$$

Case 3. $s' = 5$. In this case, $s = 3$. Histogram h' consists of five bins: $mi', mi' + 1, mi' + 2, ma' - 1, ma'$.

3.1. Left mode: Let $h1 = h'(mi') + h(mi' + 1), h2 = h(mi' + 2) + h(ma' - 1)$. Denote X'_1 as the set of first $(h_1 - 1)$ pixels $x'_{ij} \in \{mi', mi' + 1\}$, and X'_2 as the set of h_2 pixels $x'_{ij} \in \{mi' + 2, ma' - 1\}$. For each $x'_{ij} \in X'_1 \cup X'_2$,, a bit b is extracted as follows:

$$b = \begin{cases} 0, & if\ x'_{ij} = mi'\ or\ x'_{ij} = mi' + 2 \\ 1, & if\ x'_{ij} = mi' + 1,\ or\ x'_{ij} = ma' - 1. \end{cases}$$

Then pixels x_{ij} are restored as:

$$x_{ij} = \begin{cases} mi' + 1, & if\ x'_{ij} = mi'\ or\ x'_{ij} = mi' + 1 \\ mi' + 2, & if\ x'_{ij} = mi' + 2,\ or\ x'_{ij} = ma' - 1 \\ ma' - 1, & if\ x'_{ij} = ma'. \end{cases}$$

3.2. Right mode:

Let $h_1 = h'(mi' + 1) + h(mi' + 2)$ and $h_2 = h(ma' - 1) + h(ma')$. Denote X'_1 as the set of h_1 pixels $x_{ij} \in \{mi' + 1, mi' + 2\}$ and X'_2 as the set of first $h_2 - 1$ pixels $x'_{ij} \in \{ma - 1, ma\}$. For each $x'_{ij} \in X_1 \cup X_2$, a bit b is extracted as follows:

$$b = \begin{cases} 0, & if\ x'_{ij} = mi' + 2\ or\ x'_{ij} = ma' \\ 1, & if\ x'_{ij} = mi' + 1,\ or\ x'_{ij} = ma' - 1. \end{cases}$$

Then pixels x_{ij} are restored as:

$$x_{ij} = \begin{cases} mi' + 2, & if\ x'_{ij} \in \{mi' + 1, mi' + 2\} \\ ma' - 1, & if\ x'_{ij} \in \{ma' - 1, ma'\} \\ mi' + 1, & if\ x'_{ij} = mi'. \end{cases}$$

Case 4. $s' \geq 6$. First, set

$$mi' = min(X'),$$
$$ma' = max(X'),$$
$$s' = ma' - mi' + 1.$$

Then similar as in case 4 of subsection 4.1, values LPP and RPP are defined as

$$LPP = \begin{cases} \frac{mi'+ma'-1}{2}, & if\ s'\ is\ even \\ \frac{mi'+ma'}{2} - 1, & if\ s'\ is\ odd\ and\ left\ mode \\ \frac{mi'+ma'}{2}, & if\ s'\ is\ odd\ and\ right\ mode, \end{cases}$$

$$RPP = LPP + 1$$

After having LPP and RPP, extracting and restoring are performed by formulas:

$$b = \begin{cases} 0, & if\ x'_{ij} \in \{LPP, RPP\} \\ 1, & if\ x'_{ij} \in \{LPP - 1, RPP + 1\}. \end{cases}$$

$$x_{ij} = \begin{cases} x'_{ij} + 1, & if\ x'_{ij} < LPP, \\ x'_{ij} - 1, & if\ x'_{ij} > RPP, \\ x'_{ij}, & if\ otherwise. \end{cases}$$

It is noted that the number of extracted bits equals $h(LPP) + h(RPP)$, where h is the histogram of X

Table 3. Extra information

	Information	Notation	Purpose	Size in bits
1	Size of a block $r \times c$	r, c	Divide host image into blocks	6
2	The size of data bits B	DS	To embed data bits	18
3	The size of compressed location map	MS	To embed compressed location map	14
4	The compressed location map	CLM	To extract data bits	MS
5	Embedding mode: left:0, right: 1	Mode	To know embedding algorithm	1
6	The ordinal number of the last block which is used to embed data	END	To read extra information	18
	Size of extra information			**57+MS**

4.4. **Data embedding procedure.** Here, the embedding process is described in detail.

Input: A 8-bit grayscale host image I of size $R \times C$ and the data bits B.

Output: A stego image I'.

Step 1: Divide I into non-overlapped $r \times c$-sized blocks and obtain a block sequence $X = \{X_1, \ldots, X_N\}$, where $N = \lfloor \frac{R}{r} \rfloor . \lfloor \frac{C}{c} \rfloor$ is the number of blocks. Establish a location map for blocks to distinguish two block categories: The over/underflow blocks and remaining blocks. Position number in the map for X_k having pixels values of 0 or 255 (over/underflow block) is set as 1 ($LM(X_k) = 1$) and for other blocks is set as 0 ($LM(X_k) = 0$). Applying

the algorithm in subsection 4.1 for all blocks X_k with $LM(X_k) = 0$ to define the embedding capacity in the image I for modes: Left and right, and select a mode providing larger capacity.

Step 2. Visit from the first block, for block X_k if $LM(X_k) = 1$, skip (do nothing). In the case $LM(X_k) = 0$, embed data bits B in X_k according to subsection 4.1.

Step 3. When embedding data bits are completed, defined the ordinal number of the last block used for embedding, denoted as END, then establish extra information as in Table 3.

Define the sequence consisting of the least significant bits (LSBs) of $(73 + MS)$ first pixels in the host image and embed this sequence in blocks from a block with the ordinal number $(END + 1)$ by using the algorithm in subsection 4.1.

Step 4: Insert value $(57 + MS)$ into LSB of 16 first pixels of the image. Continue inserting $(57 + MS)$ bits of the extra information into LSB of the $(57 + MS)$ next pixels. After finishing this step, the stego image I' is obtained.

4.5. **Data extracting procedure.** Here, the data extracting and the host image restoring process is presented in detail.

Input: A 8-bit grayscale stego image I' of size $R \times C$ caries data bits B.

Output: Data bits B and host image I.

The extracting procedure is carried out in inverse order with the embedding process as follows.

Step1. Take 16 LSB bits from 16 first pixels of the stego image I' to get the value equal to $(57 + MS)$, then continue to extract $(57 + MS)$ LSB bits of the following pixels to obtain extra information as shown in Table 3. Decoding the compressed map to get location map LM.

Step 2. Divide I' into $r \times c$-sized blocks as the same in the data embedding procedure to obtain a block sequence $X' = \{X'_1, \ldots, X'_N\}$.

Step 3. Extracting $(73 + MS)$ embedded bits from blocks X'_k, beginning from the block with an ordinal number $(END + 1)$ at the same time restoring host blocks according to the algorithm as in subsection 4.3. Inserting extracted bits into $(73 + MS)$ LSB bits of the $(73 + MS)$ first pixels of I'.

Step 4. Extracting data bits B and restoring host blocks beginning from the first block by using the algorithm as in subsection 4.3.

After finishing step 4, data bits B are extracted and host image I is restored.

4.6. **HistE method.** As noted above, two methods HistC and HistE differ only in how LPP and RPP are defined. In the HistC method, LPP and RPP are selected at the center of histogram $h(X)$, while in HistE these two values are determined at two edges of the histogram. In fact, the HistE method is absolutely similar to the HistC method except for defining values LPP and RPP when $s \geq 4$ in embedding algorithm and when $s' \geq 6$ in extracting algorithm for a block as follows:

1. When $s \geq 4$ (in embedding algorithm)

$$LPP = mi + 1, RPP = ma - 1$$

2. When $s' \geq 6$ (in extracting algorithm)

$$LPP = mi' + 2, RPP = ma' - 2.$$

FIGURE 4. Experimental images

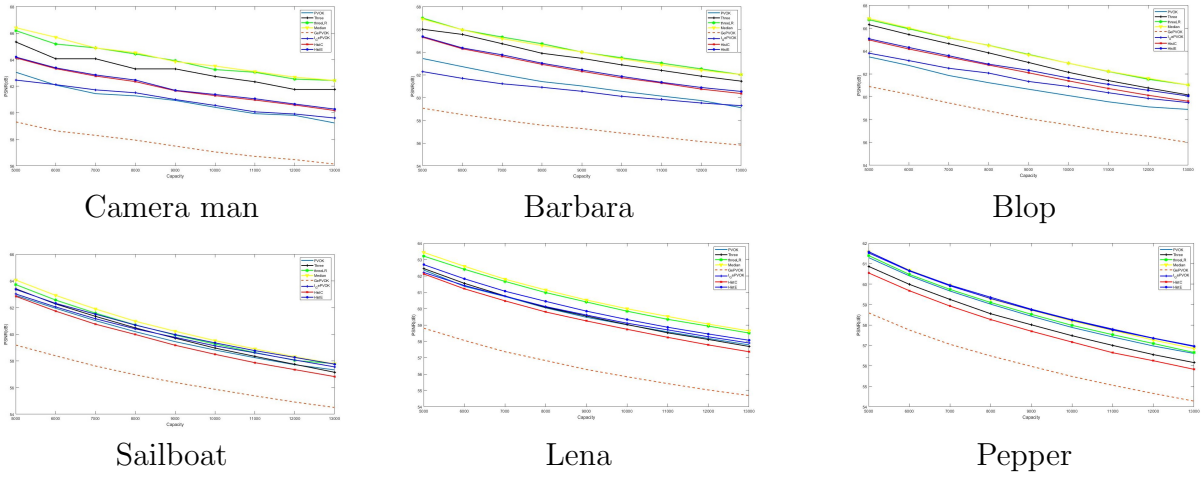

FIGURE 5. Comparisons in term of the stego quality

5. **Experimental results.** To illustrate the results of theoretical analysis, we perform experiments on 16 standard grayscale images of size 512 × 512 selected as shown in Figure 4 for comparing 10 methods: PVO [19], IPVO [27], PVO-K [23], GePVO-K [18], iGePVO-K [30], TRES [26], WengShi [38], WengPan [37], and HistC, HistE (two proposed methods). The embedded data is a random bit sequence. Programs are written in the Matlab platform and run on IdeaPad S410p Lenovo computer.

5.1. **Data hiding capacity comparison.** To obtain hiding capacity as many as possible, all blocks are used for embedding. The difference between embedding capacity and the length of the compressed location map is called data hiding capacity or payload. It is noted that for each method, the payload depends on the size of the blocks. For this reason, in the experiments, for every method and each image, the size which gets maximal payload is selected from sizes of $r \times c$ with $r, c = 1, 2, 3, 4$. The optimal size, embedding capacity, the length of compressed location map and the maximal payload of the methods

TABLE 4. Comparisons in terms of the max payload (continue to Table 5)

Images	[19]	[27]	[23]	[18]	[30]	[26]	[38]	[37]	HistC	HistE
Lena	2×2 32108 0 32108	2×2 38173 0 38173	2×2 38951 0 38951	2×2 43845 2428 41417	2×2 42709 0 42709	3×1 45263 0 45263	3×3 49319 0 49319	4×2 48937 0 48937	$3 \times 2L$ 52288 0 52288	$2 \times 2R$ 44501 0 44501
Baboon	2×2 13211 65 13146	2×2 13730 268 13462	2×2 14208 268 13940	2×2 14994 448 14546	2×2 14929 268 14661	1×3 14669 228 14441	3×3 16295 0 16295	2×4 15494 170 15324	$2 \times 2R$ 17462 268 17194	$2 \times 2L$ 15145 268 14877
Barbara	2×2 29525 35 29490	2×2 48101 35 48066	2×2 40237 35 40202	3×2 47049 5218 41831	2×2 51810 35 51775	3×1 47135 0 47135	4×2 52173 33 52140	4×2 52173 33 52140	$3 \times 2R$ 53002 34 52968	$2 \times 2R$ 51407 35 51372
Blob	2×2 41495 0 41495	2×2 53326 0 53326	2×2 52733 0 52733	2×2 64624 8151 56473	2×2 61386 0 61386	3×1 59182 0 59182	3×3 68614 0 68614	4×2 65751 0 65751	$3 \times 2L$ 69183 0 69183	$3 \times 2L$ 62753 0 62753
Cabeza	2×2 54327 64 54263	2×2 72101 64 71946	2×2 73480 64 73416	2×2 86457 7259 79198	2×2 81453 64 81389	3×1 77542 0 77542	3×3 96610 47 96563	4×2 91260 61 91199	$4 \times 2L$ 102949 61 102888	$2 \times 3R$ 90419 63 90356
Boat	2×2 27575 183 27392	2×2 35485 326 35159	2×2 33109 326 32783	2×2 39113 5792 33321	2×2 37370 326 37044	3×1 36826 0 36826	4×2 39687 216 39471	4×2 39687 216 39471	$3 \times 2L$ 39717 235 39482	$2 \times 2L$ 37772 326 37446
Camera man	2×2 44168 710 43458	2×2 72138 1041 71097	2×2 64419 1041 63378	2×3 74892 5089 69803	2×2 80691 1041 79650	1×3 76019 1398 74621	3×3 90974 794 90180	4×2 85197 765 84432	$3 \times 3L$ 91867 794 91073	$2 \times 3L$ 87927 976 86951
Goldhill	2×2 25955 0 25955	2×2 28662 0 28662	2×2 29918 0 29918	2×2 32722 1512 31210	2×2 32183 0 32183	1×3 31899 0 31899	3×3 35520 0 35520	2×4 34402 0 34402	$2 \times 3R$ 36073 0 36073	$2 \times 2L$ 32959 0 32959
Pepper	2×2 28143 50 28093	2×2 30936 50 30886	2×2 32472 50 32422	2×2 35511 658 34853	2×2 35073 50 35023	3×1 33576 0 33576	3×3 42060 47 42013	4×2 39359 47 39312	$3 \times 2L$ 43146 49 43097	$2 \times 2L$ 35940 50 35890
Sailboat	2×2 23569 0 23569	2×2 26314 0 26314	2×2 27462 0 27462	2×2 30314 1321 28993	2×2 29628 0 29628	1×3 28240 0 28240	3×3 35316 0 35316	2×4 32948 0 32948	$2 \times 3R$ 35905 0 35905	$2 \times 2L$ 30459 0 30459
Tiffany	2×2 33957 3962 29995	2×2 41775 6979 34776	2×2 41070 6979 34091	2×2 47642 13224 34418	2×2 45552 6868 38684	3×1 41965 0 41965	3×3 46716 4008 42708	4×2 45509 4592 40917	$3 \times 2R$ 50612 4866 45746	$2 \times 2R$ 46969 6868 40101

TABLE 5. Comparisons in terms of the max payload (continued from Table 4)

Images	[19]	[27]	[23]	[18]	[30]	[26]	[38]	[37]	HistC	HistE
Shirt	2×2	2×2	2×2	3×4	2×2	1×3	3×3	2×4	$2 \times 3R$	$2 \times 2R$
	17567	23405	21368	12856	28401	23904	26668	25955	28368	27988
	4126	10482	10482	11867	6545	13295	6108	6535	5618	6545
	13441	12923	10886	989	21496	10609	20560	19420	22750	21443
Chest	2×2	2×2	2×2	2×2	2×2	3×1	3×3	4×2	$2 \times 3L$	$3 \times 2L$
	45781	63502	60470	74200	69117	63280	71815	69260	74215	72090
	0	0	0	10049	0	0	0	0	0	0
	45781	63502	60470	64151	69117	63280	71815	69260	74215	72090
Things	2×2	2×2	2×2	2×3	2×2	1×3	3×3	2×4	$2 \times 3L$	$2 \times 2R$
	33931	49589	44392	50076	55265	52239	57991	56816	58336	55448
	159	171	171	5770	171	190	157	159	165	171
	33772	49418	44221	44306	55094	52049	57834	56657	58171	55277
Tank	2×2	2×2	2×2	2×2	2×2	1×3	3×3	2×4	$2 \times 2R$	$2 \times 2L$
	11187	20469	12426	13752	13609	21388	24236	23753	24252	14196
	0	0	0	681	0	0	0	0	0	0
	11187	20469	12426	13071	13609	21388	24236	23753	24252	14196
Map	2×2	2×2	2×2	2×2	2×2	1×3	3×3	2×4	$2 \times 3R$	$2 \times 2R$
	26693	32492	32852	38215	36498	34538	38234	36931	37759	37463
	0	2324	2434	5420	2434	1793	1521	1225	1622	2434
	26693	30058	30418	32795	34064	32745	36713	35706	36137	35029
Average	30575	40637	38723	44141	44730	42979	49514	47715	50946	46465
	585	1359	1366	5305	1113	1057	808	863	857	1109
	29990	39265	37357	38836	43595	41923	48706	46852	**50089**	45356

for each image are written in Table 4, 5. Moreover, for proposed methods, the mode (left and right) selected in experiments is also shown.

From Table 4, Table 5, we deduce some following conclusions:

Three methods PVO, IPVO, PVO-K where each block can embed at most two bits, have the same optimal size of 2×2, method GePVO-K where smallest and largest pixels are used for embedding has the optimal size from $2 \times 2, 2 \times 3, 3 \times 2, 3 \times 4$. The optimal size of three methods WengShi, WengPan, and histC is larger than 2×2. Meanwhile, methods HistE and iGePVO-K often reach the largest payload at the size of $2 \times 2, 2 \times 3, 3 \times 2$.

The proposed HistC has the largest payload, followed by WengShi, WengPan, HistE, and other methods.

5.2. **Stego image quality comparison.** To get high stego image quality with each given payload, only more smooth blocks are selected to embed enough desired data. To evaluate the smoothness of blocks, a context is defined for each block as shown in Figure 6. In fact, the context of the block B containing rows $r, \ldots, s$ and columns $u, \ldots, v$ is constituted from pixels (i, j) with $i = s+1, s+2; j = u, \ldots, v$ and $i = r, \ldots, s, s+1, s+2; j = v+1, v+2$ (gray region in Figure 6). The context of the block B is denoted by $\mathbb{C}(B)$. Then smoothness of B is evaluated by the local variance of the context $\mathbb{C}(B)$ as follows:

$$\nu(B) = \sqrt{\frac{\sum_{(i,j)\in\mathbb{C}(B)} (P(i,j) - \bar{P})^2}{L}},$$

where L is the number of pixels of $\mathbb{C}(B)$ and $\bar{P} = \frac{\sum_{(i,j)\in\mathbb{C}(B)} P(i,j)}{L}$.

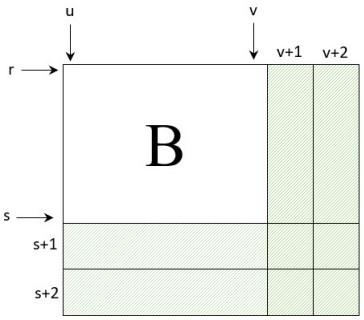

FIGURE 6. Context of blocks

To be equal, $\nu(B)$ is used for all compared methods. In other words, for each method, each original image, and each given payload, a threshold T is defined such that if only blocks having the local variance less than T are used for embedding, then obtained payload is just greater than or equal to the given payload. Then image quality is measured by coefficient PSNR between the image obtained after embedding and the original image.

The image quality comparison results of all methods are shown in Figure 5. From this, it follows that both proposed methods HistC and HistE have image quality a little less than TRES [26], WengPan [37] and WengShi [38] methods, but much better than GePVO-K [18]. While, the payloads presented in Table 4,5 show that the proposed methods, especially, HistC method outperforms other methods in all cases.

6. **Conclusion.** In this paper, we present two new reversible data hiding methods that use histogram shifting in blocks. In the first method (HistC), the center pixels of the histogram distribution are selected to be peak points, while for the second one (HistE), the edge pixels are selected. It is noted that peak points defined as above are invariant after embedding, and they can be easily restored in extracting stage. Both proposed methods have a large payload and high image quality. Especially, the data embedding capacity of the HistC method is larger than one of all existing PVO-based reversible data hiding methods.

REFERENCES

[1] A. Al-Jaber and M. K. Yaqub, "Reversible watermarking using modified difference expansion," *International Journal of Computing & Information Sciences*, vol. 4, no. 3, pp. 134–142, 2006.

[2] A. Alattar, "Reversible watermark using the difference expansion of a generalized integer transform," *IEEE transactions on image processing*, vol. 13, no. 8, pp. 1147–1156, 2004.

[3] A. M. Alattar, "Reversible watermark using difference expansion of triplets," in *Proceedings 2003 International Conference on Image Processing (Cat. No. 03CH37429)*, vol. 1. IEEE, 2003, pp. I–501.

[4] K. Bharanitharan, C.-C. Chang, Y. H. Rui, and Z.-H. Wang, "Efficient pixel prediction algorithm for reversible data hiding." *IJ Network Security*, vol. 18, no. 4, pp. 750–757, 2016.

[5] G. Bhatnagar and B. Raman, "A new robust reference watermarking scheme based on dwt-svd," *Computer Standards & Interfaces*, vol. 31, no. 5, pp. 1002–1013, 2009.

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

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

[8] S. Chen, X. Chen, and H. Fu, "General framework of reversible watermarking based on asymmetric histogram shifting of prediction error," *Advances in Multimedia*, vol. 2017, 2017.

[9] X. Chen, X. Sun, H. Sun, L. Xiang, and B. Yang, "Histogram shifting based reversible data hiding method using directed-prediction scheme," *Multimedia Tools and Applications*, vol. 74, no. 15, pp. 5747–5765, 2015.

[10] X. Chen, X. Sun, H. Sun, Z. Zhou, and J. Zhang, "Reversible watermarking method based on asymmetric-histogram shifting of prediction errors," *Journal of Systems and Software*, vol. 86, no. 10, pp. 2620–2626, 2013.

[11] J. Fridrich, M. Goljan, and R. Du, "Lossless data embedding—new paradigm in digital watermarking," *EURASIP Journal on Advances in Signal Processing*, vol. 2002, no. 2, p. 986842, 2002.

[12] W. He, Z. Cai, and Y. Wang, "Flexible spatial location-based pvo predictor for high-fidelity reversible data hiding," *Information Sciences*, vol. 520, pp. 431–444, 2020.

[13] Y. Hu, H. K. Lee, and J. Li, "De-based reversible data hiding with improved overflow location map," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 19, no. 2, pp. 250–260, 2008.

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

[15] M. Khodaei and K. Faez, "Reversible data hiding by using modified difference expansion," in *2010 2nd International Conference on Signal Processing Systems*, vol. 3. IEEE, 2010, pp. V3–31.

[16] S. Kukreja, S. S. Kasana, and G. Kasana, "Histogram based multilevel reversible data hiding scheme using simple and absolute difference images," *Multimedia Tools and Applications*, vol. 78, no. 5, pp. 6139–6162, 2019.

[17] C. C. Lee, H. C. Wu, C. S. Tsai, and Y. P. Chu, "Adaptive lossless steganographic scheme with centralized difference expansion," *Pattern Recognition*, vol. 41, no. 6, pp. 2097–2106, 2008.

[18] J. Li, Y. H. Wu, C. F. Lee, and C. C. Chang, "Generalized pvo-k embedding technique for reversible data hiding." *IJ Network Security*, vol. 20, no. 1, pp. 65–77, 2018.

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

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

[21] B. Macq, "Lossless multiresolution transform for image authenticating watermarking," in *2000 10th European Signal Processing Conference*. IEEE, 2000, pp. 1–4.

[22] Z. Ni, Y.-Q. Shi, N. Ansari, and W. Su, "Reversible data hiding," *IEEE Transactions on circuits and systems for video technology*, vol. 16, no. 3, pp. 354–362, 2006.

[23] B. Ou, X. Li, Y. Zhao, and R. Ni, "Reversible data hiding using invariant pixel-value-ordering and prediction-error expansion," *Signal processing: image communication*, vol. 29, no. 7, pp. 760–772, 2014.

[24] B. Ou, Y. Zhao, and R. Ni, "Reversible watermarking using prediction error histogram and blocking," in *International Workshop on Digital Watermarking*. Springer, 2010, pp. 170–180.

[25] G. Pan, Y. Wu, and Z. Wu, "A novel data hiding method for two-color images," in *International Conference on Information and Communications Security*. Springer, 2001, pp. 261–270.

[26] Z. Pan and E. Gao, "Reversible data hiding based on novel embedding structure pvo and adaptive block-merging strategy," *Multimedia Tools and Applications*, vol. 78, no. 18, pp. 26 047–26 071, 2019.

[27] F. Peng, X. Li, and B. Yang, "Improved pvo-based reversible data hiding," *Digital Signal Processing*, vol. 25, pp. 255–265, 2014.

[28] S. Rawat and B. Raman, "A chaos-based robust watermarking algorithm for rightful ownership protection," *International Journal of Image and Graphics*, vol. 11, no. 04, pp. 471–493, 2011.

[29] V. Sachnev, H. J. Kim, J. Nam, S. Suresh, and Y. Q. Shi, "Reversible watermarking algorithm using sorting and prediction," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 19, no. 7, pp. 989–999, 2009.

[30] N. K. Sao, N. N. Hoa, and P. Van At, "An effective reversible data hiding method based on pixel-value-ordering," *Journal of Computer Science and Cybernetics*, vol. 36, no. 2, pp. 139–158, 2020.

[31] S. Y. Shin, H. M. Yoo, and J. W. Suh, "Reversible watermarking based on histogram shifting of difference image between original and predicted images." IARIA, 2014, Springer, Berlin, Heidelberg, 2014, pp. 147–150.

[32] Q. Su and B. Chen, "Robust color image watermarking technique in the spatial domain," *Soft Computing*, vol. 22, no. 1, pp. 91–106, 2018.

[33] D. M. Thodi and J. J. Rodriguez, "Expansion embedding techniques for reversible watermarking," *IEEE transactions on image processing*, vol. 16, no. 3, pp. 721–730, 2007.

[34] D. M. Thodi and J. J. Rodriguez, "Prediction-error based reversible watermarking," in *2004 International Conference on Image Processing, 2004. ICIP'04.*, vol. 3, no. 8426938. IEEE, 2004, pp. 1549–1552.

[35] J. Tian, "Reversible data embedding using a difference expansion," *IEEE transactions on circuits and systems for video technology*, vol. 13, no. 8, pp. 890–896, 2003.

[36] S. Weng, J. S. Pan, and J. Deng, "Invariability of remainder based reversible watermarking." *J. Netw. Intell.*, vol. 1, no. 1, pp. 16–22, 2016.

[37] S. Weng, J. S. Pan, and L. Li, "Reversible data hiding based on an adaptive pixel-embedding strategy and two-layer embedding," *Information Sciences*, vol. 369, pp. 144–159, 2016.

[38] S. Weng, Y. Shi, W. Hong, and Y. Yao, "Dynamic improved pixel value ordering reversible data hiding," *Information Sciences*, vol. 489, pp. 136–154, 2019.

[39] S. Weng, W. Tan, B. Ou, and J. S. Pan, "Reversible data hiding method for multi-histogram point selection based on improved crisscross optimization algorithm," *Information Sciences*, vol. 549, pp. 13–33, 2021.

[40] W. Zheng, Y. Zhou, T. Zhang, C. Zhang, F. Zou, and Y. Shi, "Adaptive reversible data hiding based on multiple block scanning ways," 2021.