Information Sciences 181 (2011) 308–328

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Reversible data hiding based on block median preservation

Hao Luo, Fa-Xin Yu *, Hua Chen, Zheng-Liang Huang, Hui Li, Ping-Hui Wang

School of Aeronautics and Astronautics, Zhejiang University, Hangzhou 310027, PR China

ARTICLE INFO

Article history:
Received 3 September 2009
Received in revised form 5 September 2010
Accepted 19 September 2010

Keywords:
Reversible data hiding
Block median preservation
Image subsampling
Histogram shifting

ABSTRACT

This paper proposes a reversible data hiding scheme for gray level images. It exploits the high correlation among image block pixels to produce a difference histogram. Secret data is embedded based on a multi-level histogram shifting mechanism with reference to the integer median of each block. The image blocks are divided into four categories due to four corresponding embedding strategies, aiming at preserving the medians during data embedding. In decoder, the median pixels are retrieved first followed by the hidden data extraction, and the host image can be accurately recovered via an inverse histogram shifting mechanism after removing the secret data from the marked image. Experimental results validate the effectiveness of our scheme and demonstrate that it outperforms several previous methods in terms of capacity and marked image's quality.

1. Introduction

Data hiding plays an important role in information security and many data hiding methodologies in the form of diverse algorithms are presented in [1,33–35,39–41]. In recent years, reversible data hiding for digital images has drawn much attention among researchers. The reversibility means not only secret data but also host image can be precisely recovered in the decoding stage, and thus the applications are broadened in comparison with those irreversible schemes. Reversible data hiding [37,38] is applicable to any kind of scenarios where the host image must be perfectly recovered after data extraction including military remote sensing imaging, diagnostic medical imaging, precious arts (e.g., painting, calligraphy) protection, and any imaging with legal problems involved.

Nowadays a variety of reversible data hiding schemes are reported in literatures. Most of them can be classified into transform domain, compressed domain and spatial domain methods. In the transform domain schemes [37], the host image is transformed into a set of coefficients first, and then these coefficients are modified according to secret bits. After that, the modified coefficients are inversely transformed into marked pixels. For example, Yang et al. [30,31] proposed a reversible data hiding scheme based on integer discrete cosine transform, while Xuan et al.'s scheme [29] is based on integer wavelet transform. The compressed domain reversible data hiding are designed for images compressed by JPEG [2], vector quantization [3,17,25,32], block truncation coding [4], etc. The early stage spatial domain method is based on lossless data compression whereas the subsequent schemes can be further divided into two subcategories. One is based on difference expansion (DE) and the other is based on histogram shifting. The DE method is developed by Tian [21] with the idea of expanding the difference between a pair of adjacent pixels, and then embedding data in the expanded versions. For example, Chang et al. [5] proposed a just noticeable distortion (JND) based method exploiting pair-difference correlations among DCT domain subimages. In [1], Alattar extended the DE method via generalized integer transform for capacity enhancement, and as a result, $K-1$ bits can be hidden in a set of K adjacent pixels. In addition, various improved DE methods are proposed in

* Corresponding author. Address: No. 38, ZheDa Road, Yuquan Campus, Zhejiang University, School of Aeronautics and Astronautics, Hangzhou 310027, PR China. Tel.: +86 571 87951581x814; fax: +86 571 87951581x815.
E-mail address: aeeizju@gmail.com (F.-X. Yu).

0020-0255/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.ins.2010.09.022

[6,8,13,16,18,23,26–28]. In [19], the histogram shifting principle was first proposed which usually consists of two stages during data embedding. The first stage is to find peak and zero points of the host image histogram. Then the bins between the peak and zero points are shifted with one level, and hence the peak point is emptied. In the second stage, the secret bit is embedded by adjusting the new peak point and the emptied one. Hereafter, many reversible data hiding schemes based on histogram shifting are presented in [7,9–12,14,15,20,22,24]. Generally speaking, the computational complexities of the transform and compressed domains methods are higher than those spatial domain methods. In [36], both histogram shifting and DE based methods are unified due to their same essence inherently. Given their great simplicity and extended use, they have been the dominant techniques in reversible data hiding.

Many activities for the overall performance improvement are currently ongoing. More recently, Kim et al. [11] proposed a reversible data hiding scheme based on histogram shifting. Its idea is to subsample the host image into several subimages and then employ the correlation among subimages to hide message. Actually, this correlation also can be expressed by high redundancy among pixels inside a small block for their pixel values are quite similar. The reference pixel in [11] is fixed as the center of each block. However, this selection strategy is not always optimal. In this work, a novel reversible data hiding scheme based on histogram shifting is proposed, where the median pixel of each block is employed as the reference one. The basic idea behind our method is to take the host image content into account for reference pixel selection. Consequently, the histogram peak points are higher than those in [11], which is beneficial to a larger capacity. Meanwhile, as more differences are concentrated around the zero point, lower distortions are introduced.

The rest part of this paper is organized as follows. Section 2 reviews the reversible data hiding scheme proposed by Kim et al. [11]. Section 3 extensively describes our method including data embedding, extraction and image recovery procedures. Section 4 discusses the overflow and underflow prevention and the PSNR lower bounds of marked images. Experimental results and performance comparisons with other algorithms are shown in Section 5. Finally, conclusions are given in Section 6.

2. Related work

This section briefly reviews the principle of Kim et al.'s method [11]. Suppose the host image I is a $W \times H$ gray level image and subsampled into four equal-sized subimages s_1, s_2, s_3, s_4 as

$$\begin{cases} s_1(i,j) = I(2i-1, 2j-1) \\ s_2(i,j) = I(2i-1, 2j) \\ s_3(i,j) = I(2i, 2j-1) \\ s_4(i,j) = I(2i, 2j) \end{cases} \tag{1}$$

where (i,j) denotes the ith row, jth column pixel in the subimages with $i = 1,2,\ldots,W/2$, $j = 1,2,\ldots,H/2$. Then s_1 is selected as the reference image and the differences relative to s_2, s_3, s_4 are computed as

$$\begin{cases} d_2(i,j) = s_2(i,j) - s_1(i,j) \\ d_3(i,j) = s_3(i,j) - s_1(i,j) \\ d_4(i,j) = s_4(i,j) - s_1(i,j) \end{cases} \tag{2}$$

where d_2, d_3 and d_4 are difference matrices with elements belonging to the range of $[-255,255]$. Obviously there are totally $W/2 \times H/2 \times 3 = (W \times H \times 3)/4$ differences obtained and a difference histogram can be constructed based on them. Assume the histogram bins are denoted by $b_{-255},\ldots,b_0,\ldots,b_{255}$, respectively.

The data embedding processing is based on multi-level histogram shifting. Here a parameter termed embedding level (EL for short) is involved which is established depending on the capacity requirement. For simplicity, suppose $EL = 2$, then the first step is to empty $b_{\pm3}$, $b_{\pm4}$, $b_{\pm5}$ by shifting the bins between $[b_{-3}, b_{-255}]$ leftward and $[b_3, b_{255}]$ rightward by three levels, respectively. This operation can be expressed as

$$d'_k(i,j) = \begin{cases} d_k(i,j) + EL & \text{if } d_k(i,j) > EL \\ d_k(i,j) - EL & \text{if } d_k(i,j) < -EL \\ d_k(i,j) & \text{otherwise} \end{cases} \tag{3}$$

where d'_k (i,j) $(k = 2,3,4)$ denotes the shifted difference. Next, examine each d'_k one by one and three cases should be addressed for one bit embedding. Specifically, repeat Eq. (4) with $EL = 2$ and $EL = 1$, respectively.

$$d''_k(i,j) = \begin{cases} d'_k(i,j) + EL + w & \text{if } d'_k(i,j) = EL \\ d'_k(i,j) - EL - w & \text{if } d_k(i,j) = -EL \end{cases} \tag{4}$$

where d''_k (i,j) and w denote the marked difference and the current processing secret bit, respectively. Next, if $d'_k(i,j) = 0$, execute Eq. (5) for $EL = 0$.

$$d''_k(i,j) = d'_k(i,j) + w \tag{5}$$

That is, if $d'_k(i,j) = 0$ and the current secret bit is 1, $d'_k(i,j)$ is added by 1, otherwise $d'_k(i,j)$ is not changed. In fact, if coming across a $d'_k(i,j)$ belonging to the set {±2,±1,0}, one bit data can be hidden. In this way, the marked subimages s'_2, s'_3 and s'_4 are obtained as

$$\begin{cases} s'_2(i,j) = s_1(i,j) + d''_2(i,j) \\ s'_3(i,j) = s_1(i,j) + d''_3(i,j) \\ s'_4(i,j) = s_1(i,j) + d''_4(i,j) \end{cases} \tag{6}$$

where $s'_2(i,j), s'_3(i,j), s'_4(i,j)$ are associated with $s_2(i,j)$, $s_3(i,j)$, $s_4(i,j)$, respectively. Finally, the marked image I' is obtained by recomposing s_1, s'_2, s'_3 and s'_4.

The data extraction and image recovery is an inverse processing of data embedding. First, I' is subsampled into s_1, s'_2, s'_3 and s'_4 and the difference is computed with reference to s_1 as

$$\begin{cases} d''_2(i,j) = s''_2(i,j) - s_1(i,j) \\ d''_3(i,j) = s''_3(i,j) - s_1(i,j) \\ d''_4(i,j) = s''_4(i,j) - s_1(i,j) \end{cases} \tag{7}$$

Thus a difference histogram can be constructed and the secret bit can be extracted as

$$w = \begin{cases} 1 & \text{if } d''_k(i,j) \in \{\pm 5, \pm 3, 1\} \\ 0 & \text{if } d''_k(i,j) \in \{\pm 4, \pm 2, 0\} \end{cases} \tag{8}$$

where $k = 2, 3, 4$. In brief, if $d'_k(i,j)$ belongs to the set of {±5,±3,1}, the extracted bit is "1"; while if belongs to {±4,±2,0}, "0" is extracted.

To recover the host image, set EL as 0 first and inverse shift histogram as

$$d'_k(i,j) = \begin{cases} d''_k(i,j) - 1 & \text{if } d''_k(i,j) = 1 \\ d''_k(i,j) & \text{if } d''_k(i,j) = 0 \end{cases} \tag{9}$$

Then repeat the operations in Eq. (10) with $EL = 1$ and $EL = 2$ respectively.

$$d'_k(i,j) = \begin{cases} d''_k(i,j) - EL - 1 & \text{if } d''_k(i,j) = 2EL + 1 \\ d''_k(i,j) - EL & \text{if } d''_k(i,j) = 2EL \\ d''_k(i,j) + EL + 1 & \text{if } d''_k(i,j) = -2EL - 1 \\ d''_k(i,j) + EL & \text{if } d''_k(i,j) = -2EL \end{cases} \tag{10}$$

Hence, $b_{\pm 5}$, $b_{\pm 4}$, $b_{\pm 3}$ are emptied. Next, the histogram is inversely shifted as

$$d_k(i,j) = \begin{cases} d'_k(i,j) - EL & \text{if } d'_k(i,j) > EL \\ d'_k(i,j) + EL & \text{if } d'_k(i,j) < -EL \\ d'_k(i,j) & \text{otherwise} \end{cases} \tag{11}$$

Thus the original histogram is recovered and s_2, s_3, s_4 can be recovered as

$$\begin{cases} s_2(i,j) = s_1(i,j) + d_2(i,j) \\ s_3(i,j) = s_1(i,j) + d_3(i,j) \\ s_4(i,j) = s_1(i,j) + d_4(i,j) \end{cases} \tag{12}$$

Finally, the host image I is recovered by recomposing s_1, s_2, s_3 and s_4.

3. Proposed scheme

3.1. Motivation

The subsampling principle in [11] is equivalent to exploiting the redundancy among image blocks. In other words, the correlation among subimages is considered from a global viewpoint, whereas the redundancy in image blocks is considered from a local viewpoint. In the context, the proposed scheme is described mainly from the latter one. For example, each 2×2 host image block is equivalent to the set of $s_1(i,j)$, $s_2(i,j)$, $s_3(i,j)$, $s_4(i,j)$.

Suppose I is partitioned into non-overlapped $u \times v$ sized blocks. Then $u \times v$ subimages $s_1, s_2, \ldots, s_{u \times v}$ are obtained, each with size of $W/u \times H/v$. In [11], the reference image s_r is redetermined as Eq. (13) and cannot used for data embedding.

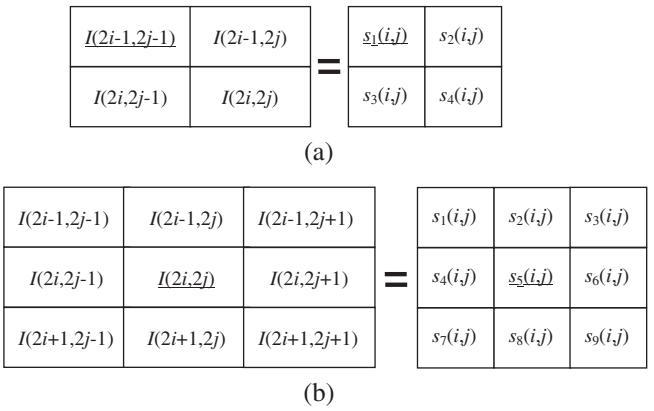


Fig. 1. Reference pixel (the underlined) selection in Kim et al.'s scheme [11], (a) 2 × 2 block partition and (b) 3 × 3 block partition.

$$s_r = s_{\lceil \frac{u}{2} - 1 \rceil \times v + \lceil \frac{v}{2} \rceil} \tag{13}$$

where $\lceil \cdot \rceil$ denotes rounding the element to the nearest integer no less than it. As shown in Fig. 1, if I is partitioned into 2 × 2 blocks, i.e., $s_1, s_2, \ldots, s_4$ generated, then $s_r = s_1$. Likewise, $s_r = s_5$ for 3 × 3 block partition.

As shown in Fig. 1(b), it is reasonable that the eight neighboring pixels are quite similar to the central $s_5(i,j)$ in statistics. That is, the differences between $s_5(i,j)$ and its neighbors are equal or close to zero. Actually, if we select the median $s_m(i,j)$ among $s_1(i,j)$, $s_2(i,j), \ldots, s_9(i,j)$ as the reference pixel $s_r(i,j)$, the differences between $s_m(i,j)$ and the others are more prone to zero in most cases. In this way, more secret bits may be embedded due to the higher peak points obtained than those in [11].

In our scheme, it is necessary to keep the reference pixel $s_m(i,j)$ unchanged all the time for it is not used for data embedding. As long as the median pixel is preserved, i.e., its position can be still accurately located and pixel value can be maintained, not only the secret data can be extracted, but also the host image can be perfectly recovered. This is the main idea of the proposed scheme.

Fortunately, this essential requirement can be achieved. An example of 2 × 2 block partition is shown in Fig. 2. Suppose $s_1(i,j) < s_4(i,j) < s_2(i,j) < s_3(i,j)$, then $s_m(i,j) = s_4(i,j)$. Thus $d_1(i,j) = s_1(i,j) - s_m(i,j) < 0$, $d_2(i,j) = s_2(i,j) - s_m(i,j) > 0$, $d_3(i,j) = s_3(i,j) - s_m(i,j) > 0$. The negative and positive differences fall into the left (L for short) and right (R for short) region of zero, respectively. In [11], the absolute values of both negative and positive differences become larger after data embedded. Nevertheless, the magnitude order of all differences is not changed. Moreover, since $s'_1(i,j) < s_4(i,j) < s'_2(i,j) < s'_3(i,j)$, we find that $s_m(i,j) = s_4(i,j)$ is still preserved. In Fig. 2, "→" denotes the rightward embedding direction, i.e., an original pixel larger than $s_m(i,j)$ is modified to be larger. On the contrary, "←" denotes the leftward embedding direction, i.e., an original pixel smaller than $s_m(i,j)$ is modified to be smaller.

3.2. Data embedding

The data embedding process is illustrated in Fig. 3 with steps described as follows.

Step 1: Image partition and subsampling.

Partition I into a set of $u \times v$ blocks. At the same time, subsample I into subimages $s_1, s_2, \ldots, s_{u \times v}$ with each sized $\lfloor \frac{W}{u} \rfloor \times \lfloor \frac{H}{v} \rfloor$.

Step 2: Median image computation.

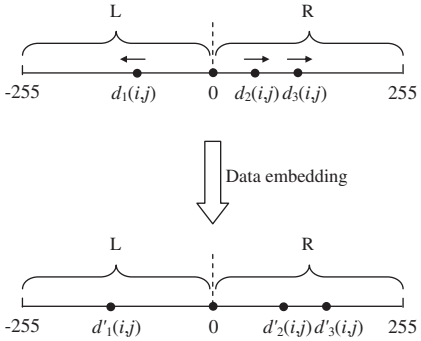


Fig. 2. Data embedding strategy based on block median preservation.

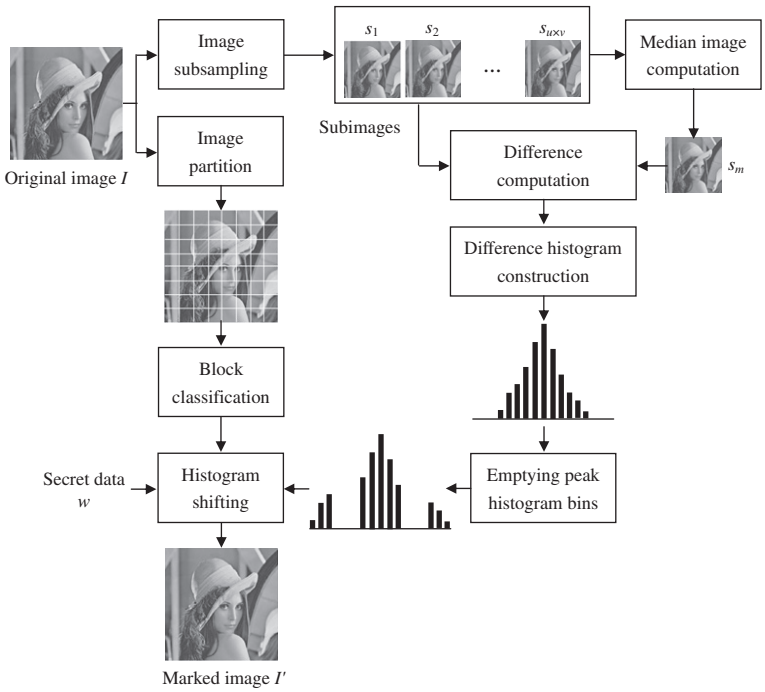


Fig. 3. Block diagram of data embedding.

The median image s_m is employed as the reference image. Retrieve $s_1(i,j), s_2(i,j), \ldots, s_{u\times v}(i,j)$ to form a pixel array, and then sort their pixel values in ascending order such that $p_1(i,j) \leqslant p_2(i,j) \leqslant \cdots \leqslant p_{u\times v}(i,j)$. The median of $p_1(i,j), p_2(i,j), \ldots, p_{u\times v}(i,j)$ is defined as

$$s_m(i,j) = p_{\lfloor \frac{u\times v+1}{2} \rfloor}(i,j) \tag{14}$$

where the subscript $\lfloor\cdot\rfloor$ means the nearest integer no larger than its element. For example, if I is subsampled into s_1, s_2, s_3, s_4, then $s_m(i,j) = p_2(i,j)$; while if nine subimages $s_1, s_2, \ldots, s_9$ produced, $s_m(i,j) = p_5(i,j)$. Repeat this operation for all pixels of the subimages and then $s_m$is obtained.

In some cases, there are more than one pixel values among $s_1(i,j), s_2(i,j), \ldots, s_{u\times v}(i,j)$ equaling $s_m(i,j)$. For simplicity, the first encountered one, i.e., with the smallest subscript, is regarded as the median pixel, while the others are regarded as candidate pixels. For example, suppose I is subsampled into four subimages and $s_1(i,j) = 161$, $s_2(i,j) = 160$, $s_3(i,j) = 160$, $s_4(i,j) = 158$. Accordingly $p_1(i,j) = s_4(i,j) = 158$, $p_2(i,j) = s_2(i,j) = 160$, $p_3(i,j) = s_3(i,j) = 160$, $p_4(i,j) = s_1(i,j) = 161$. In this case $s_m(i,j) = p_2(i,j) = s_2(i,j) = 160$, and $s_1(i,j), s_3(i,j), s_4(i,j)$ are candidate pixels. In a word, the median pixel $s_m(i,j)$ is unique for each block, although sometimes more than one (at most $u \times v$) pixel values equal it.

Step 3: Block classification.

As each block is equivalent to the set of $s_1(i,j), s_2(i,j), \ldots, s_{u\times v}(i,j)$, the block classification is reduced to classify the set of $s_1(i,j), s_2(i,j), \ldots, s_{u\times v}(i,j)$. In our case, all blocks are classified into two types, Type I and Type II. Suppose n_l, n_0 and n_r represent the numbers of pixel values smaller, equal and larger than $s_m(i,j)$ in a block, respectively. Obviously, $n_l + n_r + n_0 = u \times v$. As shown in Table 1, Type I is defined as those blocks with only one pixel value equaling $s_m(i,j)$, i.e., $n_0 = 1$, whereas those with at least two pixel values equaling $s_m(i,j)$ belong to Type II, i.e., $n_0 \geqslant 2$. Furthermore, the Type II blocks are divided into three subcategories. Type II-1 refers to those blocks with $n_l = n_r$. A special case of is that, in some Type II-1 blocks both the number of pixel values larger and smaller than $s_m(i,j)$ are zeros. That is, all the pixels are exactly the same. In a Type II-2 block, n_l is

Table 1
Block classification based on median.

Block type		Definition
Type I		$n_0 = 1$
Type II	Type II-1	$n_0 \geqslant 2$, $n_l = n_r$
	Type II-2	$n_0 \geqslant 2$, $n_l < n_r$
	Type II-3	$n_0 \geqslant 2$, $n_l > n_r$

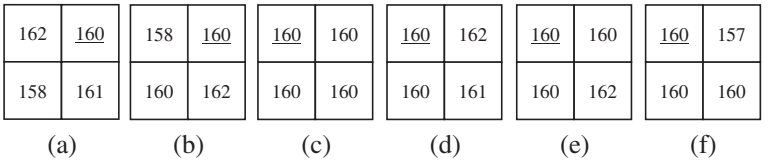


Fig. 4. Examples of different 2×2 block types, (a) Type I, (b) and (c) Type II-1, (d) and (e) Type II-2, (f) Type II-3.

Table 2
Number of different types of blocks with 2×2 block partition.

Test image	Type I	Type II-1	Type II-2	Type II-3	Total number
Lena	42,648	9662	11,688	1538	65,536
Barbara	49,382	6914	8292	948	65,536
Airplane	37,983	10,425	13,848	3280	65,536
Aerial	57,101	4262	3941	232	65,536
Truck	50,963	6804	6869	900	65,536

smaller than n_r. Type II-3 refers to those with $n_l > n_r$. Fig. 4 shows some examples of different types for 2×2 block partition with the median pixels underlined.

Table 2 lists the number of each type of blocks in several 512×512 images (see Fig. 12). In each image, a large quantity of Type I blocks exists and can be easily processed as the principle shown in Fig. 2. However, the embedding strategies for Type II blocks must be designed elaborately. Tables 3 and 4 show the median pixels distribution with 2×2 and 3×3 block partition, respectively. Clearly, the median pixels do not always lie in the center of blocks.

Step 4. Difference computation.

Compute the difference between $s_m(i, j)$ and the other $(u \times v) - 1$ pixels as

$$d_k(i,j) = s_k(i,j) - s_m(i,j) \tag{15}$$

where $1 \leqslant k \leqslant u \times v$, $k \neq m$.

Step 5: Difference histogram construction.

As $(u \times v) - 1$ differences produced in each block, the total number of differences n_d used for histogram statistics can be computed as

$$n_d = W \times H - \left\lfloor \frac{W}{u} \right\rfloor \times \left\lfloor \frac{H}{v} \right\rfloor \tag{16}$$

where $W \times H$ is the total number of differences, and $\lfloor \frac{W}{u} \rfloor \times \lfloor \frac{H}{v} \rfloor$ is the number of blocks. Clearly, a smaller block partition corresponds to a smaller n_d. Although n_d is equal to the counterpart in [11], the distribution of difference magnitudes is quite different. As shown in Table 5, the upper and bottom rows for each test image are several peak points obtained by Kim et al. [11] and our scheme, respectively. It is easy to find that the bins around b_0 (e.g., from b_{-2} to b_2) obtained by our scheme are averagely larger than those obtained by Kim et al. [11].

Table 3
Median pixel distribution with 2×2 block partition.

Test image	$s_1(i,j)$	$s_2(i,j)$	$s_3(i,j)$	$s_4(i,j)$	Total number
Lena	22,310	18,603	14,723	9900	65,536
Barbara	20,140	18,668	14,973	11,755	65,536
Airplane	26,249	17,689	12,521	9077	65,536
Aerial	19,389	16,183	14,765	15,199	65,536
Truck	20,781	17,235	14,508	13,012	65,536

Table 4
Median pixel distribution with 3×3 block partition.

Test image	$s_1(i,j)$	$s_2(i,j)$	$s_3(i,j)$	$s_4(i,j)$	$s_5(i,j)$	$s_6(i,j)$	$s_7(i,j)$	$s_8(i,j)$	$s_9(i,j)$	Total number
Lena	5313	5909	3707	2939	5415	1779	1442	1608	788	28,900
Barbara	4776	4536	3597	3333	5230	2373	1829	1707	1519	28,900
Airplane	6731	5330	3433	3434	4838	1803	1252	1240	839	28,900
Aerial	3949	3589	3187	3423	4904	2880	2375	2379	2214	28,900
Truck	4668	4262	3490	4024	4417	2848	1854	1828	1509	28,900

Table 5
Comparison of several peak histogram bins obtained by Kim et al.'s scheme [11] and our scheme.

Test image	Scheme	Histogram bins										
		−5	−4	−3	−2	−1	0	1	2	3	4	5
Lena	Kim et al.	6678	9079	12,133	15,654	18,472	19,755	18,071	15,154	11,999	8724	6237
	Our	7874	11316	15,828	21,561	26,799	26,465	18,505	12,091	7302	4319	2731
Barbara	Kim et al.	5256	7224	9149	11,503	12,913	13,914	13,471	11,796	9830	7712	5801
	Our	6725	9332	12,602	15,983	18,620	18,281	13,817	9719	6582	4456	3108
Airplane	Kim et al.	4809	7134	10,448	14,663	22,071	28,991	22,139	14,071	10,017	6765	4396
	Our	5493	8493	12,815	19,291	31,552	36,221	18,263	9935	6163	3713	2371
Aerial	Kim et al.	5095	5562	5776	6267	6475	6473	6441	6132	5696	5445	5131
	Our	7177	7737	8439	8862	9533	8909	7444	6594	5640	5056	4395
Truck	Kim et al.	6159	6040	6652	8789	5360	12,885	5514	8841	7018	6347	6551
	Our	8897	8754	9786	12,500	7861	16,423	6293	8497	6009	5388	4841

The histograms are illustrated in Fig. 5. It seems that more differences in our histograms concentrate around b_0 in comparison with those in [11], while fewer fall into the bins far from it.

According to Table 5 and Fig. 5, our histograms seem a little lean to left with respect to b_0. For example, the b_{-1} (i.e., 26,799) of Lena is higher than b_0 (i.e., 26,465). This is because the median pixel in a 2×2 block defined as p_2 is not always the accurate median. In practical, as shown in Fig. 6 the accurate median lies in the center of two pixels p_2 and p_3. It is usually a floating point number and thus cannot be directly used. In our case, we use the integer p_2 as the block median to approximate the accurate one.

Step 6: Empty peak points.

Before data embedding, the bins in the range of $[b_{-2EL-1}, b_{-EL-1}]$ and $[b_{EL+1}, b_{2EL+1}]$ are emptied as

$$d'_k(i,j) = \begin{cases} d_k(i,j) + EL & \text{if } d_k(i,j) > EL \\ d_k(i,j) - EL & \text{if } d_k(i,j) < -EL \\ d_k(i,j) & \text{otherwise} \end{cases} \tag{17}$$

where $1 \leqslant k \leqslant u \times v$, $k \neq m$. For example, given $EL = 2$ and the original histogram is shown in Fig. 7(a), all bins smaller than b_{-2} and larger than b_2 are shifted leftward and rightward respectively, as shown in Fig. 7(b). Consequently, the bins between $[b_{-5}, b_{-3}]$ and $[b_3, b_5]$ are emptied. To the block composed by $s_1(i,j)$, $s_2(i,j), \ldots, s_{u\times v}(i,j)$, this operation can be given as

$$s'_k(i,j) = \begin{cases} s_k(i,j) + EL & \text{if } s_k(i,j) > s_m(i,j) + EL \\ s_k(i,j) - EL & \text{if } s_k(i,j) < s_m(i,j) - EL \\ s_k(i,j) & \text{otherwise} \end{cases} \tag{18}$$

where $1 \leqslant k \leqslant u \times v$, $k \neq m$ and $s'_k(i,j)$ denotes the results after emptying peak bins.

Step 7: Histogram shifting.

This is the key operation of data embedding, and different types of blocks are tackled with different shifting strategies. For simplicity, we take the example of $EL = 2$ and 2×2 block partition as shown in Fig. 8. These strategies are described below and the corresponding example blocks can be found in Fig. 4.

(1) Type I.

In a Type I block, it is easy to find that

$$\begin{cases} n_l = \left\lfloor \frac{u\times v}{2} \right\rfloor - 1, n_0 = 1, n_r = \left\lfloor \frac{u\times v}{2} \right\rfloor & \text{if } (u \times v) \bmod 2 = 0 \\ n_l = n_r = \left\lfloor \frac{u\times v}{2} \right\rfloor, n_0 = 1 & \text{if } (u \times v) \bmod 2 = 1 \end{cases} \tag{19}$$

For example, in Fig. 8(a), $n_l = n_0 = 1$, $n_r = 2$. The secret bit is embedded by repeating Eq. (20) for $EL = 2$ and $EL = 1$ respectively.

$$s''_k(i,j) = \begin{cases} s'_k(i,j) + EL + w & \text{if } s'_k(i,j) - s_m(i,j) = EL \\ s'_k(i,j) - EL - w & \text{if } s'_k(i,j) - s_m(i,j) = -EL \end{cases} \tag{20}$$

where $1 \leqslant k \leqslant u \times v$, $k \neq m$.

(2) Type II-1.

According to the definition of block Type II-1, we have

$$\begin{cases} n_0 \geqslant 2 \\ 0 \leqslant n_r = n_l \leqslant \left\lfloor \frac{(u\times v) - n_0}{2} \right\rfloor \end{cases} \tag{21}$$

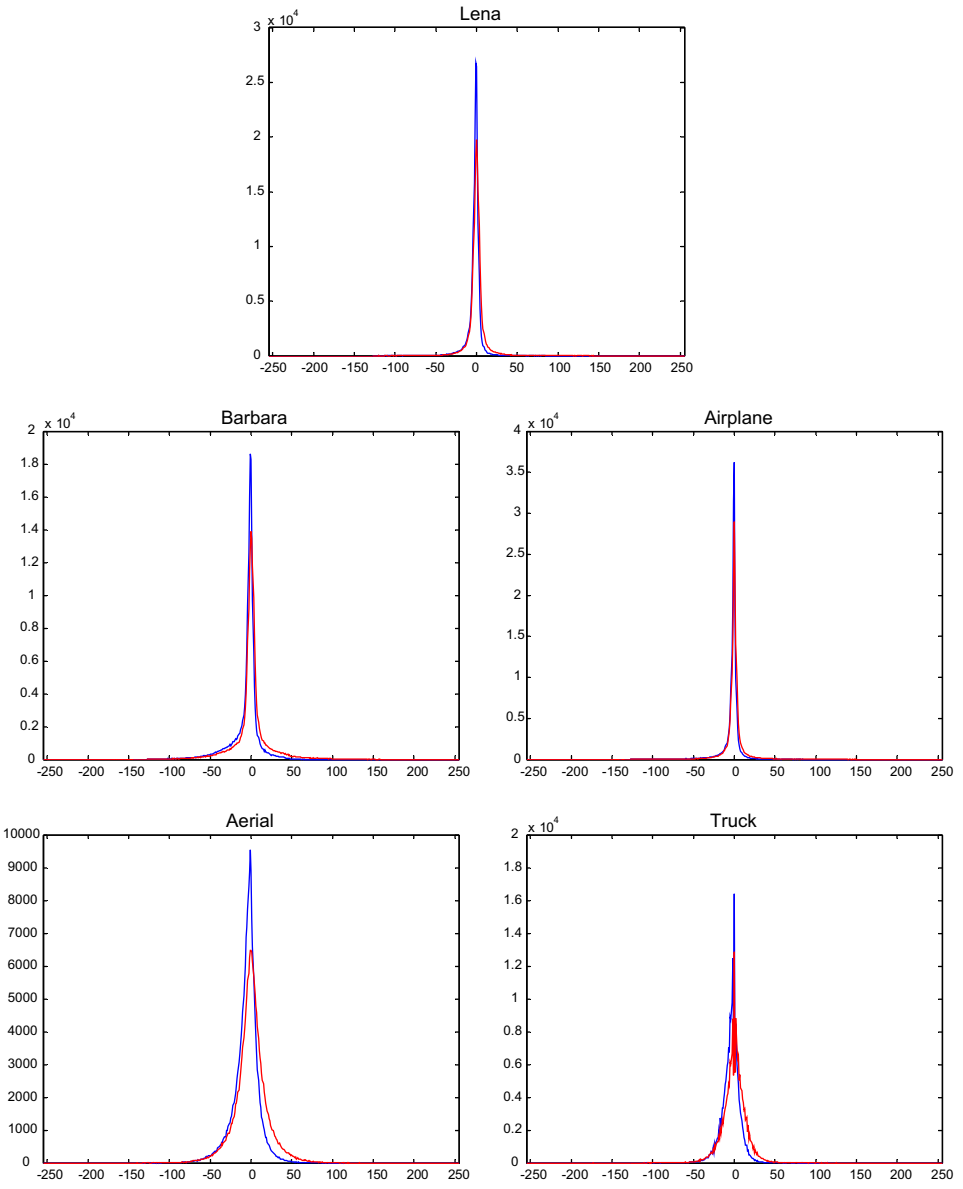


Fig. 5. Histogram comparisons of Kim et al.'s scheme [11] (red curve) and our scheme (blue curve) with 2×2 block partition. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

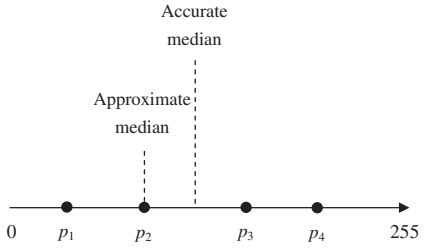


Fig. 6. Approximate and accurate median of a 2×2 block.

For example, in Fig. 8(b), $n_l = n_r = 2$, $n_0 = 2$; while in Fig. 8(c), $n_l = n_r = 0$, $n_0 = 4$. When $EL > 0$, repeat Eq. (20) for $EL = 2$ first and then for $EL = 1$. When $EL = 0$, implement

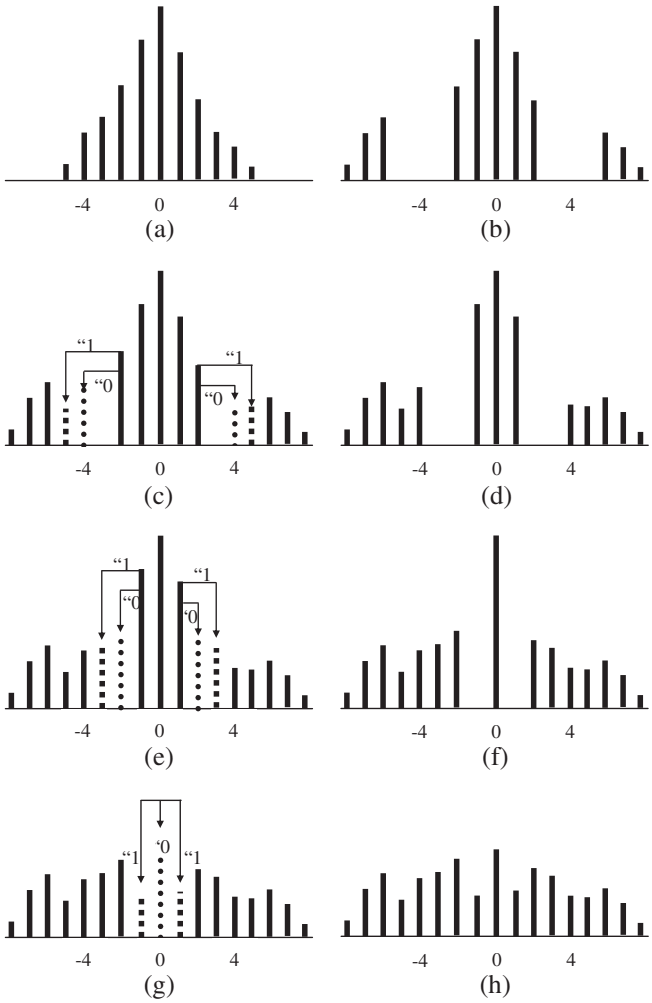


Fig. 7. Difference histogram shifting mechanism with $EL = 2$.

$$s''_k(i,j) = \begin{cases} s'_k(i,j) + (-1)^{q+1} & \text{if } s'_k(i,j) = s_m(i,j), w = 1 \\ s'_k(i,j) & \text{if } s'_k(i,j) = s_m(i,j), w = 0 \end{cases} \tag{22}$$

where $1 \leqslant k \leqslant u \times v$, $k \neq m$, and q denotes the qth encountered $s_k(i,j)$ equaling $s_m(i,j)$ $(k \neq m)$. Unlike Type I, the pixels equaling $s_m(i,j)$ are specially processed. In particular, scan $s'_1(i,j)$, $s'_2(i,j), \ldots, s'_{u\times v}(i,j)$, if coming across the first pixel value which is equal to the block median, it is tagged as $s_m(i,j)$; when another pixel $s'_k(i,j) = s_m(i,j)$, $q = 1$ is assigned, and thus $s''_k(i,j) = s'_k(i,j) + 1$ if the current secret bit is 1. When a third pixel $s'_k(i,j) = s_m(i,j)$, $q = 2$. In a word, if $s'_k(i,j) = s_m(i,j)$ $(k \neq m)$ and the current secret bit is 1, the $s''_k(i,j)$ is obtained by interleavedly adding or subtracting 1 from $s'_k(i,j)$. If $s'_k(i,j) = s_m(i,j)$ $(k \neq m)$ and the current secret bit is 0, then $s'_k(i,j)$ is not changed, i.e., $s''_k(i,j) = s'_k(i,j)$.

(3) Type II-2.

In a Type II-2 block, we have

$$\begin{cases} n_0 \geqslant 2 \\ 0 \leqslant n_l < n_r \leqslant (u \times v) - n_0 \end{cases} \tag{23}$$

For example, in Fig. 8(d), $n_l = 0$, $n_r = n_0 = 2$; while in Fig. 8(e), $n_l = 0$, $n_r = 1$, $n_0 = 3$. When $EL > 0$, repeat Eq. (20) for $EL = 2$ and $EL = 1$ respectively. When $EL = 0$, implement

$$s''_k(i,j) = \begin{cases} s'_k(i,j) - 1 & \text{if } s'_k(i,j) = s_m(i,j), w = 1,\ q < n_r - n_l \\ s'_k(i,j) + (-1)^{q+1} & \text{if } s'_k(i,j) = s_m(i,j), w = 1,\ q \geqslant n_r - n_l \\ s'_k(i,j) & \text{if } s'_k(i,j) = s_m(i,j),\ w = 0 \end{cases} \tag{24}$$

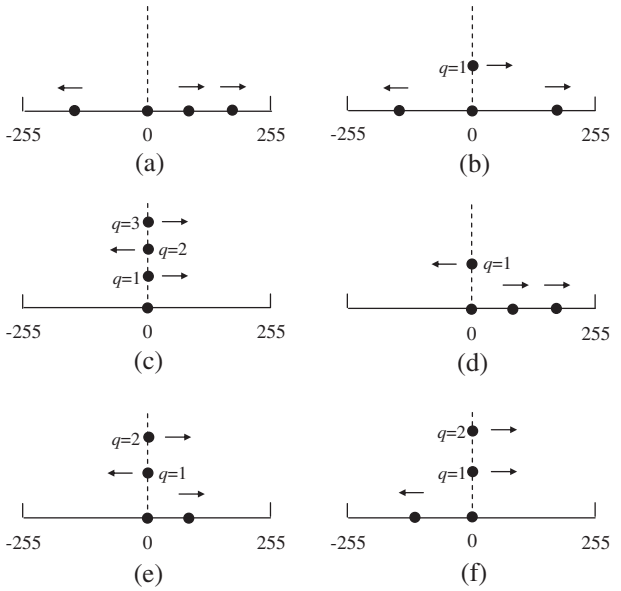


Fig. 8. Data embedding strategies for different types of blocks with 2×2 block partition, (a) Type I, (b) and (c) Type II-1, (d) and (e) Type II-2, (f) Type II-3.

where $1 \leqslant k \leqslant u \times v$, $k \neq m$ and q has the same meaning as that in Type II-1 block data embedding. If $s'_k(i,j) = s_m(i,j)(k \neq m), s'_k(i,j)$ is kept unchanged when $w = 0$, otherwise the following rule is adopted. If $q < n_r - n_l$, let $s''_k(i,j) = s'_k(i,j) - 1$; while if $q \geqslant n_r - n_l$, let $s''_k(i,j) = s'_k(i,j) + (-1)^{q+1}$. In a word, our strategy is to distribute $s''_k(i,j)$ evenly into L and R. First, $n_r - n_l$ pixels with values equaling $s_m(i,j)$ are transformed into L so that there are both n_r points in both L and R. Second, the remained pixels (if any) with values equaling $s_m(i,j)$ are assigned interleavedly to L and R.

For example, in Fig. 8(d), $n_l = 0$, $n_r = n_0 = 2$, $n_r - n_l = 2$, thus the pixel with $q = 1$ is transformed into L if a bit "1" is embedded. In Fig. 8(e), $n_l = 0$, $n_r = 1$, $n_0 = 3$, $n_r - n_l = 1$, and thus the pixel with $q = 1$ is transformed into L, while that with $q = 2$ is transformed into R if two bits "11" are embedded.

(4) Type II-3.

In a Type II-3 block, we have

$$\begin{cases} n_0 \geqslant 2 \\ 0 \leqslant n_r < n_l \leqslant (u \times v) - n_0 \end{cases} \tag{25}$$

For example, in Fig. 8(f), $n_l = 1$, $n_r = 0$, $n_0 = 3$. When $EL > 0$, repeat Eq. (20) for $EL = 2$ and $EL = 1$ respectively. When $EL = 0$, implement

$$s''_k(i,j) = \begin{cases} s'_k(i,j) + 1 & \text{if } s'_k(i,j) = s_m(i,j),\ w = 1, q \leqslant n_l - n_r \\ s'_k(i,j) - (-1)^{q+1} & \text{if } s'_k(i,j) = s_m(i,j),\ w = 1, q > n_l - n_r \\ s'_k(i,j) & \text{if } s'_k(i,j) = s_m(i,j),\ w = 0 \end{cases} \tag{26}$$

where $1 \leqslant k \leqslant u \times v$, $k \neq m$ and the meaning of q is the same as that in Type II-1 block processing. If $s'_k(i,j) = s_m(i,j)(k \neq m)$ and $w = 0$, $s'_k(i,j)$ is kept unchanged. The processing mechanism to the pixels with values $s'_k(i,j) = s_m(i,j)$ when embedding "1"s is contrary to the Type II-2 blocks. If $s'_k(i,j) = s_m(i,j)(k \neq m)$ and $w = 1$, the following rule is adopted. If $q < n_l - n_r$, let $s''_k(i,j) = s'_k(i,j) + 1$; while if $q \geqslant n_l - n_r$, let $s''_k(i,j) = s'_k(i,j) - (-1)^{q+1}$. The strategy is also to distribute $s''_k(i,j)$ evenly in L and R. First, $n_l - n_r$ pixels with values equal to $s_m(i,j)$ are transformed into R so that there are n_l points in both sides. Second, the remained pixels (if any) with values equal to $s_m(i,j)$ are assigned to L and R interleavedly. For example, in Fig. 8(f), $n_l = 1$, $n_r = 0$, $n_0 = 3$, $n_l - n_r = 1$, and thus the pixels with $q = 1$ and $q = 2$ are both transformed into R.

Notice that for Type II-1, Type II-2 and Type II-3 blocks, the transformation of $s'_k(i,j) = s_m(i,j)$ $(k \neq m)$ is only executed when the current secret bit is "1". If the current secret bit is "0", these pixels are kept unchanged, i.e., $s''_k(i,j) = s'_k(i,j)$. In this way, the median pixels still can be preserved after data embedding. The difference histogram shifting operations with $EL = 2$, 1 and 0 are shown in Fig. 7(c) and (d), Fig. 7(e) and (f), and Fig. 7(g) and (h), respectively. In this way, $s''_1(i,j), s''_2(i,j), \ldots, s''_{u\times v}(i,j)$ are obtained for each block.

Step 8: Image recomposition.

The last step is to recompose the subimages $s''_1, s''_2, \ldots, s''_{u\times v}$ and the marked image I' is obtained.

3.3. Data extraction and Image recovery

Suppose the received marked image I' in decoder suffers no alterations during transmission. As an inverse process of data embedding, the block diagram of data extraction and image recovery is shown in Fig. 9 with the details described below.

Step 1: Image partition and subsampling.

Partition I' into non-overlapping $u \times v$ blocks with each comprising the pixels $s''_1(i,j), s''_2(i,j), \ldots, s''_{u\times v}(i,j)$.

Step 2: Median image reconstruction.

Sort $s''_1(i,j), s''_2(i,j), \ldots, s''_{u\times v}(i,j)$ in ascending order as $p''_1(i,j) \leqslant p''_2(i,j) \leqslant \cdots \leqslant p''_{u\times v}(i,j)$. The median pixel $s_m(i,j)$ can be retrieved as

$$s_m(i,j) = p''_{\left\lfloor \frac{u\times v+1}{2} \right\rfloor}(i,j) \tag{27}$$

Step 3: Difference computation.

The differences between $s_m(i,j)$ and the other $(u \times v) - 1$ pixels are computed as

$$d''_k(i,j) = s''_k(i,j) - s_m(i,j) \tag{28}$$

where $1 \leqslant k \leqslant u \times v$, $k \neq m$.

Step 4: Data extraction.

First, set $EL = 0$ and scan all the differences. If we come across ±1, a secret bit "1" is extracted; otherwise "0" is extracted.

$$w = \begin{cases} 1 & \text{if } d''_k(i,j) \in \{-1, 1\} \\ 0 & \text{if } d''_k(i,j) = 0 \end{cases} \tag{29}$$

Second, repeat the scanning process with $EL = 1$ and $EL = 2$ respectively, and extract the secret bit as

$$w = \begin{cases} 1 & \text{if } d''_k(i,j) \in \{2 \times EL + 1, -2 \times EL - 1\} \\ 0 & \text{if } d''_k(i,j) \in \{2 \times EL, -2 \times EL\} \end{cases} \tag{30}$$

That is, if we come across ±3, "1" is extracted; if ±2, "0" is extracted. Then in the next round, if we come across ±5, "1" is extracted; if ±4, "0" is extracted.

Step 5: Difference histogram construction.

According to Eq. (28), there are totally n_d differences produced and a histogram as shown in Fig. 10(a) is constructed.

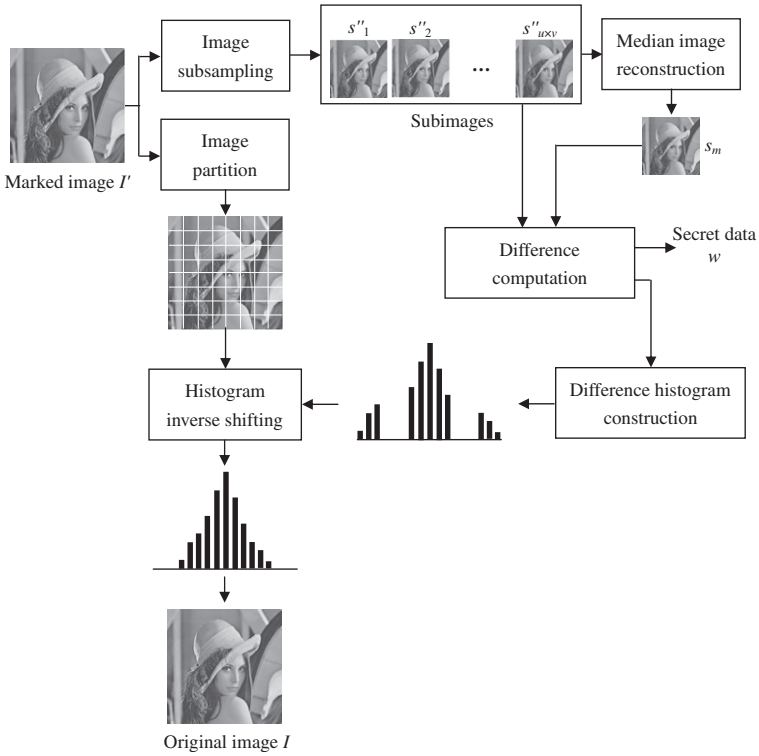


Fig. 9. Block diagram of data extraction and image recovery.

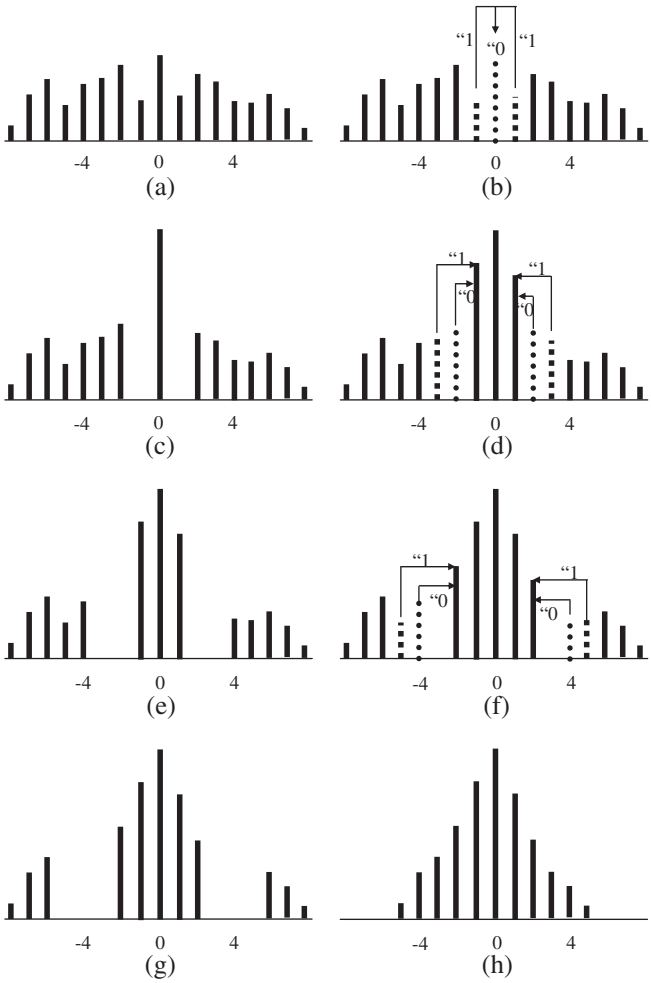


Fig. 10. Difference histogram shifting mechanism with $EL = 2$.

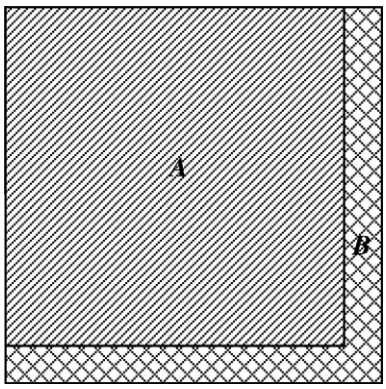


Fig. 11. Location map embedding strategy.

Step 6: Inverse histogram shifting.
This is the key operation for image recovery. When $EL = 0$, implement

$$s'_k(i,j) = \begin{cases} s''_k(i,j) + 1 & \text{if } d''_k(i,j) = -1 \\ s''_k(i,j) - 1 & \text{if } d''_k(i,j) = 1 \\ s''_k(i,j) & \text{if } d''_k(i,j) = 0 \end{cases} \tag{31}$$

Thus Fig. 10(b) is transformed into Fig. 10(c). When $EL > 0$, implement

$$s'_k(i,j) = \begin{cases} s''_k(i,j) - EL & \text{if } d''_k(i,j) = 2EL \\ s''_k(i,j) - EL - 1 & \text{if } d''_k(i,j) = 2EL + 1 \\ s''_k(i,j) + EL & \text{if } d''_k(i,j) = -2EL \\ s''_k(i,j) + EL + 1 & \text{if } d''_k(i,j) = -2EL - 1 \end{cases} \tag{32}$$

Repeat this operation for $EL = 1$ and $EL = 2$ respectively, as shown in Fig. 10(d)–(g). At last, refill $b_{\pm 3}$, $b_{\pm 4}$, $b_{\pm 5}$ as

$$s_k(i,j) = \begin{cases} s'_k(i,j) - EL & \text{if } s'_k(i,j) > s_m(i,j) + EL \\ s'_k(i,j) + EL & \text{if } s'_k(i,j) < s_m(i,j) - EL \\ s'_k(i,j) & \text{otherwise} \end{cases} \tag{33}$$

where $1 \leqslant k \leqslant u \times v$, $k \neq m$. Hence the histogram is recovered as shown in Fig. 10(h).

Step 7: Image recomposition.

As $s_1, s_2, \ldots, s_{u\times v}$ obtained, recompose all the blocks and the host image I is recovered.

4. Discussions

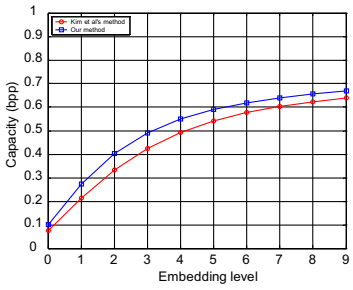
4.1. Overflow and underflow prevention

To an 8-bit gray level image, the pixel values overflow or underflow may occur during data embedding. The overflow refers to $I'(i,j) > 255$ whereas the underflow means $I'(i,j) < 0$. Obviously, the operations of peak points empty and histogram shifting may cause overflow or underflow. Suppose I_{max} and I_{min} denote the maximum and minimum of the pixel values in I, respectively. In the worst case, if $I_{max} + EL + 1 > 255$, the overflow occur, and if $I_{min} - EL - 1 < 0$, the underflow occur. Typically, the overflow and underflow prevention is coped well with two strategies. One is based on modulo operation and the other is based on location map. In our scheme, the modulo-based method is not applicable because the median pixels cannot be preserved after data embedded. In other words, the marked pixels may not evenly distribute in L and R, and therefore the reversibility is destroyed. Additionally, several image content-dependent thresholds are usually involved in modulo-based approaches. In this work, the location map method is used. If an overflow or underflow occurs, a "1" is recorded, otherwise

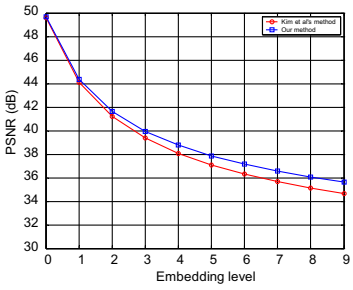
Table 6
Lower bounds of PSNR (dB) in our scheme and Kim et al.'s scheme [11].

Parameters		Kim et al.'s scheme	Our scheme
$u \times v = 2 \times 2$	$EL = 0$	49.38	49.38
	$EL = 1$	43.36	43.36
	$EL = 2$	39.84	39.84
$u \times v = 3 \times 3$	$EL = 0$	48.64	48.64
	$EL = 1$	42.62	42.62
	$EL = 2$	39.10	39.10
$u \times v = 4 \times 4$	$EL = 0$	48.41	48.41
	$EL = 1$	42.39	42.39
	$EL = 2$	38.87	38.87
$u \times v = 5 \times 5$	$EL = 0$	48.31	48.31
	$EL = 1$	42.29	42.29
	$EL = 2$	38.77	38.77
$u \times v = 8 \times 8$	$EL = 0$	48.20	48.20
	$EL = 1$	42.18	42.18
	$EL = 2$	38.66	38.66

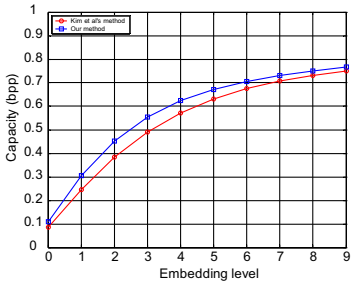
Fig. 12. Test images, Lena, Barbara, Airplane, Aerial, Truck (from left to right).



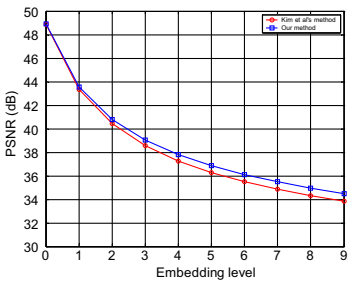
C (bpp), 2×2 block partition



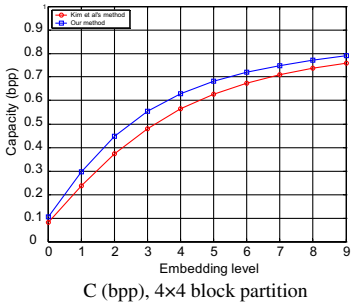
PSNR, 2×2 block partition



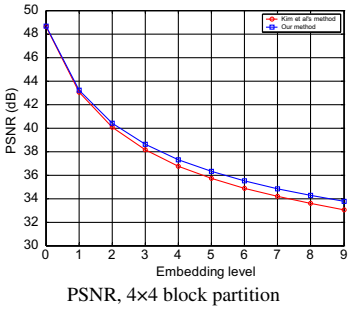
C (bpp), 3×3 block partition



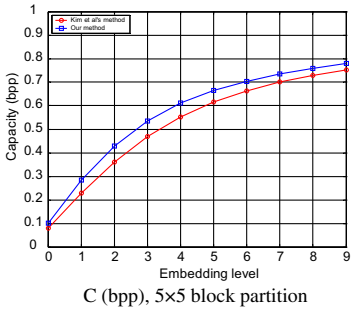
PSNR, 3×3 block partition



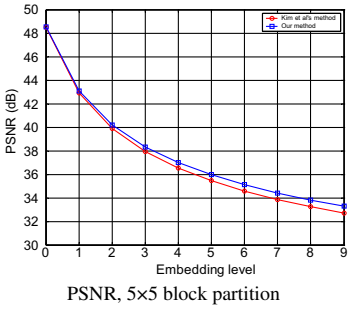
C (bpp), 4×4 block partition



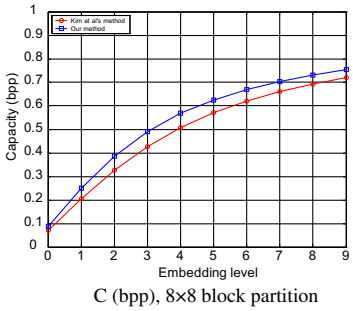
PSNR, 4×4 block partition



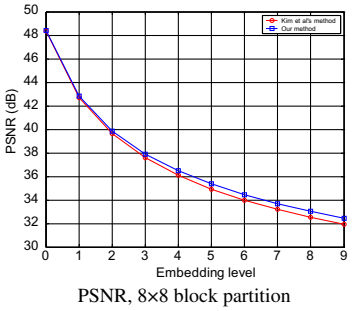
C (bpp), 5×5 block partition



PSNR, 5×5 block partition



C (bpp), 8×8 block partition



PSNR, 8×8 block partition

Fig. 13. Comparison of Kim et al.'s scheme [11] and our scheme on Lena image.

a "0" recorded. In this way, after all pixels processed, a $W \times H$ binary location map M is produced. A large quantity of experimental results shows that no overflow or underflow occurs if EL is set as a small integer. With the EL increased, a few of them occur, and thus an M with a large quantity of "0"s and a few of "1"s is produced. Then M is compressed by high efficiency lossless compression techniques. In our case, the arithmetic coding is used. The compressed map M_c is also hidden in I. More specifically, I is segmented into two parts, A and B, for hiding w and M_c respectively. An example is shown in Fig. 11. Given a 512×512 host image and 3×3 block partition, the pixels in the last two rows and columns, i.e., the part B, are used for hiding M_c. The remainder pixels constitute A. That is, some pixels in B are selected according to a secret key, and their least significant bits (LSB) are replaced by M_c, and finally, these LSB bits are hidden along with w in A. In the decoding phase, the same key is used to retrieve the selected pixels' LSB bits in B and thus M_c is reconstructed. After lossless decompression, M is further generated. Then we can extract w and the original LSB bits of the selected pixels in B. Finally, the host image can be recovered by removing w from A and replacing LSB bits of the selected pixels in B with the latter part of data extracted from A.

4.2. Lower bounds of PSNR

The mean square error (MSE) between I and I' is defined as

$$\mathrm{MSE} = \frac{1}{W \times H} \sum_{i=1}^{W} \sum_{j=1}^{H} (I(i,j) - I'(i,j))^2 \tag{34}$$

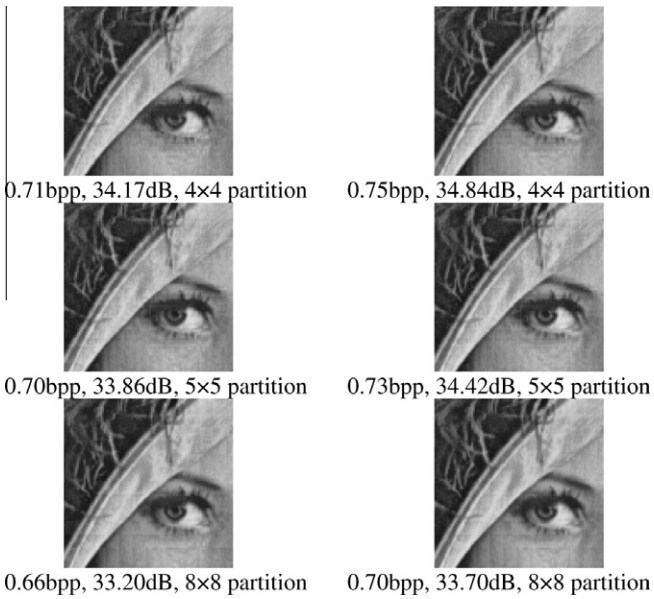


Fig. 14. Marked Lena images with $EL = 7$ obtained by Kim et al.'s scheme [11] (left column) and our scheme (right column).

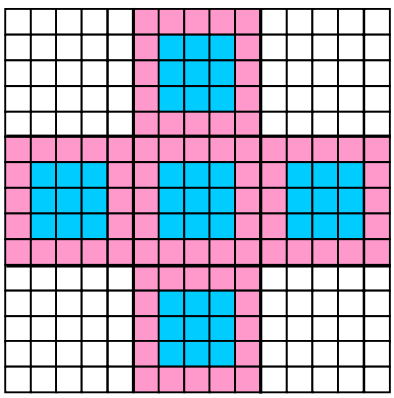


Fig. 15. Block artifact distortion analysis.

Table 7
Comparison of our scheme and Kim et al.'s scheme with 2×2 block partition.

Test image	Indicator	$EL = 0$		$EL = 1$		$EL = 2$	
		Kim et al.	Our	Kim et al.	Our	Kim et al.	Our
Lena	C (bits)	19,755	26,465	56,298	71,769	87,106	105,421
	PSNR	49.60	49.68	44.10	44.35	41.22	41.65
Barbara	C (bits)	13,914	18,281	40,298	50718	63,597	76,420
	PSNR	49.54	49.59	43.88	44.03	40.79	41.05
Airplane	C (bits)	28,991	36,221	73,201	86,036	101,935	115,262
	PSNR	49.71	49.80	44.40	44.64	41.66	42.04
Aerial	C (bits)	6473	8909	19,389	25,886	31,788	41,342
	PSNR	49.45	49.48	43.60	43.68	40.27	40.43
Truck	C (bits)	12,885	16,423	23,759	30,577	41,389	51,574
	PSNR	49.53	49.57	43.71	43.82	40.43	40.60
Average improvement	ΔC_a	4856.20		10408.20		12840.80	
	$\Delta PSNR_a$	0.06		0.17		0.28	

Table 8
Comparison of our scheme and Kim et al.'s scheme with 3×3 block partition.

Test image	Indicator	$EL = 0$		$EL = 1$		$EL = 2$	
		Kim et al.	Our	Kim et al.	Our	Kim et al.	Our
Lena	C (bits)	22,581	28,860	64,564	79,787	100,401	118,372
	PSNR	48.86	48.92	43.34	43.55	40.44	40.79
Barbara	C (bits)	15,865	19,788	45,964	56,054	72,461	85,033
	PSNR	48.79	48.83	43.12	43.25	40.02	40.23
Airplane	C (bits)	32,436	39,853	83,011	96,501	116,858	130,636
	PSNR	48.96	49.03	43.62	43.83	40.84	41.18
Aerial	C (bits)	7342	9156	21,767	26,789	35,843	43,650
	PSNR	48.71	48.73	42.85	42.90	39.52	39.62
Truck	C (bits)	14,418	18,184	26,848	33,674	46,803	57,340
	PSNR	48.78	48.82	42.96	43.05	39.66	39.81
Average improvement	ΔC_a	4639.80		10130.20		12533.00	
	$\Delta PSNR_a$	0.05		0.14		0.25	

Table 9
Comparison of our scheme and Kim et al.'s scheme with 4×4 block partition.

Test image	Indicator	$EL = 0$		$EL = 1$		$EL = 2$	
		Kim et al.	Our	Kim et al.	Our	Kim et al.	Our
Lena	C (bits)	21,629	27,723	62,319	77,861	97,691	117,160
	PSNR	48.61	48.66	43.04	43.22	40.08	40.41
Barbara	C (bits)	15,106	18,849	44,621	54,558	71,007	83,928
	PSNR	48.55	48.58	42.84	42.95	39.70	39.90
Airplane	C (bits)	31,301	38,976	80,965	95,580	114,520	130,948
	PSNR	48.70	48.77	43.29	43.50	40.44	40.78
Aerial	C (bits)	6784	8677	20,489	25,767	33,698	42,124
	PSNR	48.47	48.49	42.59	42.65	39.24	39.34
Truck	C (bits)	13,866	16,900	25,727	32,023	44,858	54,774
	PSNR	48.53	48.56	42.69	42.77	39.37	39.50
Average improvement	ΔC_a	4487.80		10333.60		13432.00	
	$\Delta PSNR_a$	0.04		0.13		0.22	

In our case, there are at most $(u \times v) - 1$ pixels are modified in a $u \times v$ block, and the worst case is that they are added or subtracted by $EL + 1$. Therefore the MSE is computed as

$$\mathrm{MSE} = (EL+1)^2 \times \left(\frac{u \times v - 1}{u \times v}\right) \tag{35}$$

The quality of I' is evaluated using the peak signal to noise ratio (PSNR) defined as

$$\mathrm{PSNR} = 10 \times \log_{10} \frac{255^2}{\mathrm{MSE}} (\mathrm{dB}) \tag{36}$$

According to Eqs. (35) and (36), the expected PSNR lower bounds of I' are given in Table 6, which are exactly the same as those obtained by Kim et al. [11].

5. Experimental results

In the conventional histogram shifting based methods [19], some side information such as the peak and zero points must be transmitted to the decoder for data extraction and image recovery. In contrast, the side information in this work includes the block size and the embedding level, while the peak and zero points are not required. For example, for 4×4 block partition and $EL = 4$, only six side bits should be transmitted.

The validity of the proposed scheme is demonstrated by a series of experiments. As shown in Fig. 12, five 512×512 gray level images, Lena, Barbara, Airplane, Aerial and Truck are selected as test images. The secret data w is a binary sequence created by pseudo random number generator. Besides PSNR, the hiding capacity is another essential factor for performance evaluation. Here it is represented by C (bits) or C (bpp), where the former means the total number of secret bits embedded, and the latter means bits per pixel (bpp) provided. As the side information is extremely small with respect to the capacity, the side bits are not counted in C (bits) and C (bpp) in the following experimental results.

Table 10
Comparison of our scheme and Kim et al.'s scheme with 5×5 block partition.

Test image	Indicator	$EL = 0$		$EL = 1$		$EL = 2$	
		Kim et al.	Our	Kim et al.	Our	Kim et al.	Our
Lena	C (bits)	20,739	26,603	60,193	74,185	94,624	112,425
	PSNR	48.50	48.54	42.90	43.07	39.90	40.20
Barbara	C (bits)	14,638	17,980	42,900	51,385	68,228	79,633
	PSNR	48.44	48.47	42.72	42.81	39.55	39.71
Airplane	C (bits)	29,774	36,401	77,592	91,863	111,409	127,757
	PSNR	48.57	48.64	43.13	43.32	40.24	40.56
Aerial	C (bits)	6481	7924	19,246	23,543	31,960	38,893
	PSNR	48.36	48.38	42.47	42.52	39.11	39.19
Truck	C (bits)	13,048	16,247	24,416	30,676	42,884	52,275
	PSNR	48.42	48.45	42.57	42.64	39.24	39.36
Average improvement	ΔC_a	4095.00		9461.00		12375.60	
	$\Delta \mathrm{PSNR}_a$	0.04		0.11		0.20	

Table 11
Comparison of our scheme and Kim et al.'s scheme with 8×8 block partition.

Test image	Indicator	$EL = 0$		$EL = 1$		$EL = 2$	
		Kim et al.	Our	Kim et al.	Our	Kim et al.	Our
Lena	C (bits)	18,595	23,225	54,023	65,811	85,416	101,299
	PSNR	48.36	48.40	42.71	42.84	39.63	39.86
Barbara	C (bits)	12,981	15,511	38,063	44,843	60,637	70,702
	PSNR	48.31	48.33	42.54	42.61	39.32	39.45
Airplane	C (bits)	26,225	32,580	69,715	83,532	101,516	118,000
	PSNR	48.43	48.48	42.89	43.06	39.92	40.20
Aerial	C (bits)	5645	6869	16,774	20,599	27,565	33,965
	PSNR	48.25	48.26	42.34	42.37	38.94	39.01
Truck	C (bits)	11,372	13,983	21,281	26,831	37,416	46,313
	PSNR	48.30	48.32	42.41	42.48	39.05	39.15
Average improvement	ΔC_a	3470.00		8352.00		11545.80	
	$\Delta \mathrm{PSNR}_a$	0.03		0.09		0.16	

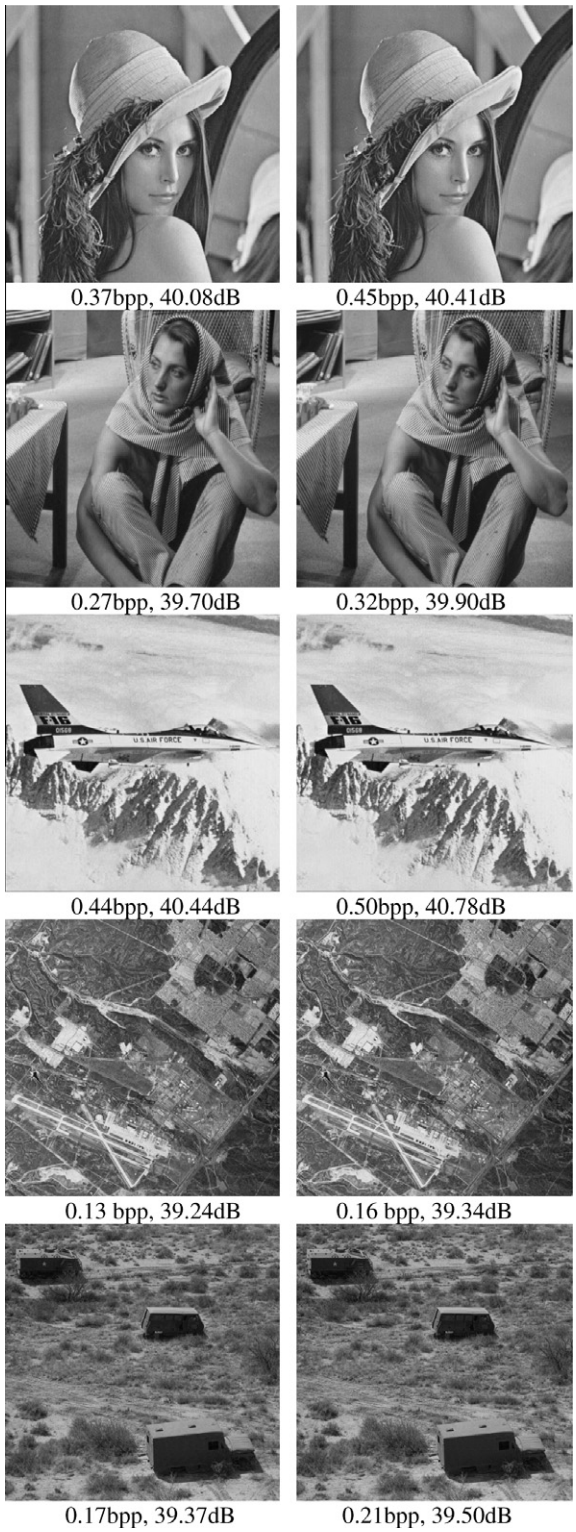


Fig. 16. Marked Lena, Barbara, Airplane, Aerial, Truck (from top to bottom) obtained with Kim et al.'s scheme [11] (left column) and our scheme (right column) with 4×4 block partition, $EL = 2$.

Experimental results on Lena obtained by Kim et al.'s scheme and our scheme are compared in Fig. 13 with various *EL* and block sizes tested. It is easy to find that both capacities and PSNRs of our scheme are superior to those obtained by Kim et al.

Table 12
Comparison of our scheme and several other schemes with 3×3 block partition and $EL = 0$.

Scheme	Lena		Airplane		Baboon	
	C (bpp)	PSNR	C (bpp)	PSNR	C (bpp)	PSNR
Ni et al. [19]	0.02	48.2	0.06	48.2	0.02	48.2
Hwang et al. [10]	0.02	48.2	0.06	48.3	0.02	48.2
Kuo et al. [12]	0.02	48.2	0.06	48.3	0.02	48.2
Kim et al. [11]	0.09	48.9	0.12	49.0	0.02	48.7
Our scheme	0.11	48.9	0.15	49.0	0.03	48.7

[11]. Specifically, the distinct improvement of capacity can be achieved for $EL = 2, 3, 4$. Besides, a larger EL indicates more evident enhancement of PSNR.

In order to investigate the effect of block size, the vertical axes of all capacity and PSNR figures in Fig. 13 are normalized to [0, 1.0] bpp and [30,50] dB, respectively. For example, for $EL = 9$, the largest capacity is achieved at approximately 0.8 bpp with the block size 4×4, while it drops when a smaller or larger block size used. The reasons lie in the following two aspects. On one hand, the capacity is closely related to the total number of differences n_d. A smaller block size implies a smaller n_d and lower peak points. Further, a small capacity is supplied. On the other hand, a larger size block partition is capable of generating more differences. However, they may not be always suitable for data embedding. This is because the differences between $s_m(i,j)$ and its neighbors far from it may be relatively large, i.e., they are not highly correlated any longer. Therefore these pixels can be only used for peak points empty. In a word, the largest capacity is achieved with a good tradeoff between the number of provided differences and those of really used for hiding secret bits.

It is apparent that capacity and image quality always contradict each other in data hiding techniques. According to Fig. 13, the highest PSNR values are achieved for 2×2 block partition. To Lena, approximately 50 dB can be achieved at for $EL = 0$ and 36 dB for $EL = 9$, respectively. An interesting phenomenon arises in consequence with 8×8 block partition for both its capacity and PSNR results are inferior in comparison with those of 4×4 block partition. This is just because although more differences produced, many of them are not used for data embedding but only for peak points empty. In addition, as the differences are relatively large, thus peak points empty leads to relatively serious distortions.

Fig. 14 highlights parts of the marked Lena images for $EL = 7$. The left and right columns are produced by Kim et al. [11] and our scheme respectively. In each row, the block sizes are the same, i.e., 4×4, 5×5 and 8×8. Both capacities and PSNR values of our scheme are higher than those of [11]. Nevertheless, the block artifact manifesting in the way of intensity changes are noticeable. This can be explained using the example of 5×5 block partition as shown in Fig. 15 where a 15×15 block is examined. The central 5×5 block and its four neighbors are investigated. In most cases, the median pixels usually lie in the center (marked as blue) of blocks. The differences between the fringe pixels (marked as red) and the medians are relatively large. For a large EL, histogram modification leads to serious distortions on these fringe pixels. Consequently, the block artifact is distinguishable in the adjacent blocks boundaries. From Fig. 14, these block artifact distortions are alleviated to some extent in our scheme compared with those obtained by Kim et al. [11]. This is because the centermost pixel of each block is fixed as the reference pixel in [11] and thus the differences are larger than that in our scheme in statistics.

Besides Lena, we also perform our scheme on the other images to test the average performance, and the results are compared with those obtained by Kim et al. [11]. As shown in Tables 7–11, 2×2, 3×3, 4×4, 5×5, 8×8 block partitions are used. Note in 3×3 and 5×5 block partitions, the last two rows and columns of host images are not included for data embedding since the subimage size is set as $\lfloor \frac{W}{u} \rfloor \times \lfloor \frac{H}{v} \rfloor$. The correspondence between EL and capacity are $C_{EL=2} > C_{EL=1} > C_{EL=0}$, $\mathrm{PSNR}_{EL=2} < \mathrm{PSNR}_{EL=1} < \mathrm{PSNR}_{EL=0}$. All the PSNRs are always larger than the lower bounds listed in Table 6. For example, for 2×2 block partition, the lowest PSNRs of five test images in our scheme are 40.43 dB, 43.68 dB, 49.48 dB for $EL = 2, 1, 0$, respectively. All of them are larger than the lower bounds 39.84 dB, 43.36 dB and 49.38 dB. ΔC_a and $\Delta\mathrm{PSNR}_a$ represent the average improvement in capacity and PSNR for all the test images, respectively. It is easy to find that the enhancement in capacity is achieved at most with $\Delta C_a = 13432.00$ bits for 4×4 block partition and $EL = 2$; while the enhancement in PSNR is achieved at most with $\Delta\mathrm{PSNR}_a = 0.28$ dB for 2×2 block partition and $EL = 2$.

Fig. 16 shows the marked images Lena, Barbara, Airplane, Aerial, Truck obtained by our scheme and [11] with $EL = 2$ and 4×4 block partition. Although more data are embedded in our scheme, the introduced distortions are reduced for the PSNRs are enhanced.

Our scheme is also compared with several other histogram shifting based methods, where 3×3 block partition and $EL = 0$ are used for three 512×512 images. As shown in Table 12, our scheme evidently improves the capacities and PSNRs compared with the results obtained by [19,10,12,11].

It is necessary to point out that the performance of our scheme is not evidently superior than those transform domain methods. However, the complexity of our scheme is smaller than those transformation based algorithms [29,31] for no forward and inverse transformations are involved. Besides, the image distortion in our scheme is easier to be estimated and controlled compared with transform domain algorithms.

6. Conclusions

A reversible data hiding scheme for gray level images is proposed in this paper. It is based on a multi-level histogram shifting mechanism. The histogram is constructed by block differences with the reference of their integer medians. In order to preserve the block medians, the image blocks are divided into four categories with different embedding strategies used. In decoder, these medians are retrieved first, and accordingly the secret data is extracted and the host image is accurately reconstructed easily. A larger embedding level corresponds to a larger capacity with the marked images PSNR ranging from 32 to 50 dB. By experimental comparison of several available methods we observe that our scheme leads to a better performance in terms of both capacity and marked image's quality.

Acknowledgement

The authors thank the editor and the anonymous reviewers for their constructive comments and valuable suggestions for readability improvement. This work is partially supported by the National Scientific Fund of China (No. 61003255, No. 61071128).

References

[1] A.M. Alattar, Reversible watermark using the difference expansion of a generalized integer transform, IEEE Transactions on Image Processing 13 (8) (2004) 1147–1156.
[2] C.C. Chang, C.C. Lin, C.S. Tseng, W.L. Tai, Reversible hiding in DCT-based compressed images, Information Sciences 177 (13) (2007) 2768–2786.
[3] C.C. Chang, C.Y. Lin, Reversible steganographic method using SMVQ approach based on declustering, Information Sciences 177 (8) (2007) 1796–1805.
[4] C.C. Chang, C.Y. Lin, Y.H. Fan, Lossless data hiding for color images based on block truncation coding, Pattern Recognition 41 (2008) 2347–2357.
[5] C.C. Chang, P.Y. Lin, J.S. Yeh, Preserving robustness and removability for digital watermarks using subsampling and difference correlation, Information Sciences 179 (13) (2009) 2283–2293.
[6] C.C. Chang, T.C. Lu, A difference expansion oriented data hiding scheme for restoring the original host images, The Journal of Systems and Software 79 (12) (2006) 1754–1766.
[7] X. Gao, L. An, X. Li, D. Tao, Reversibility improved lossless data hiding, Signal Processing 89 (10) (2009) 2053–2065.
[8] J.Y. Hsiao, K.F. Chan, J.M. Chang, Block-based reversible data embedding, Signal Processing 89 (4) (2009) 556–569.
[9] H.C. Huang, W.C. Fang, I.T. Tsai, Reversible data hiding using histogram-based difference expansion, in: IEEE International Symposium on Circuits and Systems, 2009, pp. 1661–1664.
[10] J. Hwang, J.W. Kim, J.U. Choi, A reversible watermarking based on histogram shifting, in: International Workshop on Digital Watermarking, Lecture Notes in Computer Science, vol. 4283, 2006, pp. 348–361.
[11] K.S. Kim, M.J. Lee, H.Y. Lee, H.K. Lee, Reversible data hiding exploiting spatial correlation between sub-sampled images, Pattern Recognition 42 (11) (2009) 3083–3096.
[12] W.C. Kuo, D.J. Jiang, Y.C. Huang, Reversible data hiding based on histogram, in: International Conference on Intelligent Computing, Lecture Notes in Artificial Intelligence, vol. 4682, 2007, pp. 1152–1161.
[13] C.C. Lee, H.C. Wu, C.S. Tsai, Y.P. Chu, Adaptive lossless steganographic scheme with centralized difference expansion, Pattern Recognition 41 (6) (2008) 2097–2106.
[14] C.C. Lin, N.L. Hsueh, A lossless data hiding scheme based on three-pixel block differences, Pattern Recognition 41 (4) (2008) 1415–1425.
[15] C.C. Lin, W.L. Tai, C.C. Chang, Multilevel reversible data hiding based on histogram modification of difference images, Pattern Recognition 41 (12) (2008) 3582–3591.
[16] D.C. Lou, M.C. Hu, J.L. Liu, Multiple layer data hiding scheme for medical images, Computer Standards and Interfaces 31 (2) (2009) 329–335.
[17] Z.M. Lu, J.X. Wang, B.B. Liu, An improved lossless data hiding scheme based on image VQ-index residual value coding, Journal of Systems and Software 82 (6) (2009) 1016–1024.
[18] T.C. Lu, C.C. Chang, Lossless nibbled data embedding scheme based on difference expansion, Image and Vision Computing 26 (2008) 632–638.
[19] Z. Ni, Y.Q. Shi, N. Ansari, W. Su, Reversible data hiding, IEEE Transactions on Circuits and Systems for Video Technology 16 (3) (2006) 354–362.
[20] W.L. Tai, C.M. Yeh, C.C. Chang, Reversible data hiding based on histogram modification of pixel differences, IEEE Transactions on Circuits and Systems for Video Technology 19 (6) (2009) 906–910.
[21] J. Tian, Reversible data embedding using a difference expansion, IEEE Transactions on Circuits and Systems for Video Technology 13 (8) (2003) 890–896.
[22] P. Tsai, Y.C. Hu, H.L. Yeh, Reversible image hiding scheme using predictive coding and histogram shifting, Signal Processing 89 (6) (2009) 1129–1143.
[23] H.W. Tseng, C.C. Chang, An extended difference expansion algorithm for reversible watermarking, Image and Vision Computing 26 (8) (2008) 1148–1153.
[24] H.W. Tseng, C.P. Hsieh, Prediction-based reversible data hiding, Information Sciences 179 (14) (2009) 2460–2469.
[25] J.X. Wang, Z.M. Lu, A path optional lossless data hiding scheme based on VQ index table joint neighboring coding, Information Sciences 179 (19) (2009) 3332–3348.
[26] S. Weng, Y. Zhao, J.S. Pan, R. Ni, Reversible data hiding using the companding technique and improved DE method, Circuits Systems Signal Process 27 (2) (2008) 229–245.
[27] S. Weng, Y. Zhao, J.S. Pan, Lossless data hiding based on companding technique and difference expansion of triplets, IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences E90-A (8) (2007) 1717–1718.
[28] S. Weng, Y. Zhao, J.S. Pan, A novel reversible data hiding scheme, International Journal of Innovative Computing, Information and Control 4 (2) (2008) 351–358.
[29] G. Xuan, Y.Q. Shi, Q. Yao, Z. Ni, C. Yang, J. Gao, P. Chai, Lossless data hiding using histogram shifting method based on integer wavelets, in: International Workshop on Digital Watermarking, Lecture Notes in Computer Science, vol. 4283, 2006, pp. 323–332.
[30] B. Yang, M. Schmucker, W. Funk, C. Brush, S. Sun, Integer DCT-based reversible watermarking for images using companding technique, in: Proceedings of the SPIE, Security, Steganography and Watermarking of Multimedia Contents, vol. 5306, 2004, pp. 405–415.
[31] B. Yang, M. Schmucker, X. Niu, C. Busch, S.H. Sun, Integer-DCT-based reversible image watermarking by adaptive coefficient modification, in: Proceedings of the SPIE, Security, Steganography, and Watermarking of Multimedia Contents, vol. 5681, 2005, pp. 218–229.
[32] B. Yang, Z.M. Lu, S.H. Sun, Reversible watermarking in the VQ-compressed domain, in: Proceedings of the Fifth International Conference on Visualization, Imaging, and Image Processing, 2005, pp. 289–303.
[33] Z.H. Wang, T.D. Kieu, C.C. Chang, M.C. Li, A novel information concealing method based on exploiting modification direction, Journal of Information Hiding and Multimedia Signal Processing 1 (1) (2010) 1–9.

[34] K. Yamamoto, M. Iwakiri, Real-time audio watermarking based on characteristics of PCM in digital instrument, Journal of Information Hiding and Multimedia Signal Processing 1 (2) (2010) 59–71.
[35] H.C. Huang, Y.H. Chen, Genetic fingerprinting for copyright protection of multicast media, Soft Computing 13 (4) (2009) 383–391.
[36] B. Yang, M. Schmucker, C. Busch, X. Niu, S, Sun, Approaching optimal value expansion for reversible watermarking, in: Proceedings of the 7th Workshop on Multimedia and Security (ACM MM&SEC), 2005, pp. 95–102.
[37] C.C. Chang, P.Y. Pai, C.M. Yeh, Y.K. Chan, A high payload frequency-based reversible image hiding method, Information Sciences 180 (11) (2010) 2286–2298.
[38] C.C. Chang, T.D. Kieu, A reversible data hiding scheme using complementary embedding strategy, Information Sciences 180 (16) (2010) 3045–3058.
[39] L. Li, X. Yuan, Z. Lu, J.S. Pan, Rotation invariant watermark embedding based on scale-adapted characteristic regions, Information Sciences 180 (15) (2010) 2875–2888.
[40] H.J. Shiu, K.L. Ng, J.F. Fang, R.C.T. Lee, C.H. Huang, Data hiding methods based upon DNA sequences, Information Sciences 180 (11) (2010) 2196–2208.
[41] Y. Lee, H. Kim, Y. Park, A new data hiding scheme for binary image authentication with small image distortion, Information Sciences 179 (22) (2009) 3866–3884.