General Framework to Histogram-Shifting-Based Reversible Data Hiding

Xiaolong Li, Bin Li, Bin Yang, and Tieyong Zeng

***Abstract*— Histogram shifting (HS) is a useful technique of reversible data hiding (RDH). With HS-based RDH, high capacity and low distortion can be achieved efficiently. In this paper, we revisit the HS technique and present a general framework to construct HS-based RDH. By the proposed framework, one can get a RDH algorithm by simply designing the so-called shifting and embedding functions. Moreover, by taking specific shifting and embedding functions, we show that several RDH algorithms reported in the literature are special cases of this general construction. In addition, two novel and efficient RDH algorithms are also introduced to further demonstrate the universality and applicability of our framework. It is expected that more efficient RDH algorithms can be devised according to the proposed framework by carefully designing the shifting and embedding functions.**

***Index Terms*— Embedding performance, histogram shifting (HS), reversible data hiding (RDH).**

I. Introduction

FOR most image data hiding methods, the host image is permanently distorted and it can not be restored from the marked content. But in some applications such as medical image sharing [1], multimedia archive management [2], and image trans-coding [3], any distortion due to data embedding is intolerable and the availability of the original image is in high demand. To this end, a solution called "reversible data hiding" (RDH) is proposed, in which the host image can be fully restored after data embedding [4]. RDH is a hybrid method which combines various techniques to ensure the reversibility. Its feasibility is mainly due to the lossless compressibility of natural images.

Many RDH methods have been proposed in recent years, e.g., the methods based on lossless compression [5]–[7], difference expansion (DE) [8]–[10], histogram shifting (HS) [11]–[13], and integer transform [14]–[17], etc. All these methods aim at increasing the embedding capacity (EC) as high as possible while keeping the distortion low.

Tian's DE algorithm [8] is an important work of RDH. In DE algorithm, the host image is divided into pixel pairs, and the difference value of two pixels in a pair is expanded to carry one data bit. This method can provide an embedding rate (ER) up to 0.5 bits per pixel (BPP) and it significantly outperforms previous compression-based works. In particular, Tian employed a location map to record all expandable locations, and afterwards, the technique of location map is widely adopted by most RDH algorithms. Later on, Tian's work has been improved in many aspects. Alattar [18] generalized DE to a triple or a quad of pixels which can increase the maximum ER from 0.5 BPP to 0.75 BPP. Kamstra and Heijmans [9] utilized low-pass image to predict expandable locations so that the location map can be remarkably compressed. Thodi and Rodriguez [10] achieved a significant improvement by incorporating DE with HS. In addition, instead of the difference value, Thodi and Rodriguez suggested utilizing the prediction-error for expansion embedding since this can better exploit local correlations within neighboring pixels. In [19], as an extension of Thodi and Rodriguez's work, Hu *et al.* proposed a method by constructing a payload dependent location map. Recently, Hu *et al.*'s method is further investigated by Li *et al.* [20] by considering adaptive embedding in which smooth pixels are embedded with more data bits. Another improvement of Hu *et al.*'s method is the work of Zhou and Au [21] by introducing a new capacity parameters determination strategy.

Ni *et al.*'s HS-based algorithm [11] is another important work of RDH, in which the peak of image histogram is utilized to embed data. In this method, each pixel value is modified at most by 1, and thus the visual quality of marked image is guaranteed. In [12], Lee *et al.*'s proposed a method by using the histogram of difference image. This method outperforms Ni *et al.*'s by improving both EC and visual quality. The spatial correlation of natural images is exploited in Lee *et al.*'s method and thus a more appropriate histogram is obtained. In other words, compared with the ordinary one-dimensional histogram, the difference-histogram is better for RDH since it is regular in shape and has a much higher peak point. In [13], Hong *et al.* proposed a new HS-based method by modifying the prediction-error histogram. This method can well exploit the image redundancy and thus achieve a more better performance compared with the previously introduced

Manuscript received July 19, 2012; revised January 22, 2013; accepted January 31, 2013. Date of publication February 8, 2013; date of current version March 29, 2013. The work of B. Li was supported in part by the National Natural Science Foundation of China under Grant 61103174 and the Fundamental Research General Program of Shenzhen City under Grant JCYJ20120613113535357. The work of T. Zeng was supported in part by Grant RGC 211710, Grant RGC 211911, the NSFC under Grant 11271049 and the FRGs of Hong Kong Baptist University. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Hitoshi Kiya.

X. Li and B. Yang are with the Institute of Computer Science and Technology, Peking University, Beijing 100871, China (e-mail: lixiaolong@pku.edu.cn; yang_bin@pku.edu.cn).

B. Li is with the College of Information Engineering, Shenzhen University, Shenzhen 518060, China (e-mail: libin@szu.edu.cn).

T. Zeng is with the Department of Mathematics, Hong Kong Baptist University, Kowloon, Hong Kong (e-mail: zeng@hkbu.edu.hk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIP.2013.2246179

DE-based methods such as [8]–[10]. Recently, Wu and Huang [22] proposed a novel HS-based method where the histogram bins used for expansion embedding are specifically selected such that the embedding distortion is minimized. The experimental results reported in [22] demonstrated that this method is better than some state-of-the-art works including the most recently proposed integer-transform-based method [16].

Besides aforementioned works [11]–[13], [22], many HS-based RDH algorithms have also been proposed so far [23]–[31]. In general, HS-based RDH is implemented by modifying host image's histogram of a certain dimension. It has two major advantages. On one hand, the maximum modification to pixel values can be controlled and thus the embedding distortion can be well limited. On the other hand, the location map used to record underflow/overflow locations is usually small in size especially for low ER case. As pointed out by Sachnev *et al.* [32], RDH algorithm with smaller, or in some cases, no location maps, are very desirable. Therefore, HS is a good choice among existing approaches of RDH.

The main goal of this work is to provide a general framework to HS-based RDH. First in Section II, by analyzing in details Ni *et al.*'s and Lee *et al.*'s works, we summarize that HS-based RDH algorithms have a common feature: for data embedding, certain pixel values (or pixel-value-arrays, for high-dimensional histogram) are shifted to create vacant spaces whereas some others are manipulated to carry hidden data by filling the created vacant spaces. In Section III, a general framework to construct HS-based RDH is presented. According to the proposed general framework, one can get a HS-based RDH algorithm by simply designing the so-called shifting and embedding functions. In Section IV, by taking specific shifting and embedding functions, we show that several RDH algorithms reported in the literature [1], [10], [12], [19], [33] are special cases of this general construction. In Section V, to further demonstrate the universality and applicability of the proposed framework, two novel and efficient HS-based RDH algorithms are devised and introduced in details. Finally, concluding remarks are given in the last section.

II. Related Works

We start our presentation by introducing a modified version of Ni *et al.*'s method [11]. The data embedding is implemented by shifting one-dimensional image histogram.

Consider here a gray-scale image I. For a given integer a, $1 \leq a \leq 253$, the hidden data is embedded into I in the following way to get the marked image $\widetilde{I}$:

$$\widetilde{I}_{i,j} = \begin{cases} I_{i,j} - 1, & \text{if } I_{i,j} < a \\ I_{i,j} - m, & \text{if } I_{i,j} = a \\ I_{i,j} + m, & \text{if } I_{i,j} = a+1 \\ I_{i,j} + 1, & \text{if } I_{i,j} > a+1 \end{cases} \quad (1)$$

where (i, j) is a pixel location and $m \in \{0, 1\}$ is a data bit to be embedded. Notice that in this procedure, each pixel value is modified at most by 1. Thus the PSNR of marked image versus original image is at least 48.13 dB. Consequently, the visual quality of marked image is guaranteed.

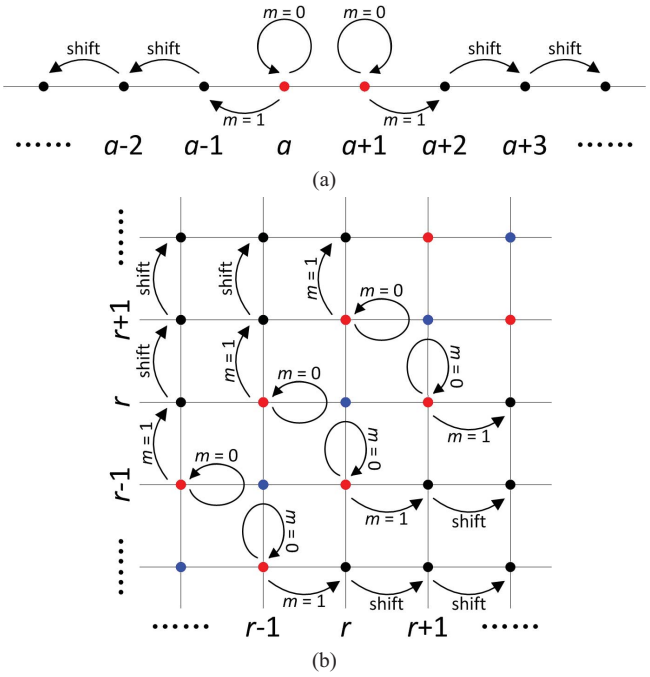

Fig. 1. Illustrations of the modified version of Ni *et al.* [11] method and Lee *et al.* [12] method. The black values are shifted, and each red value is manipulated to carry one data bit $m \in \{0, 1\}$.

The decoder can extract the embedded data and restore the original image by simply reading marked pixel values.

1) If $\widetilde{I}_{i,j} < a - 1$, there is no hidden data in the pixel and its original value is $\widetilde{I}_{i,j} + 1$.
2) If $\widetilde{I}_{i,j} \in \{a - 1, a\}$, the pixel is used to carry hidden data and its original value is a. The embedded data bit is $m = a - \widetilde{I}_{i,j}$.
3) If $\widetilde{I}_{i,j} \in \{a + 1, a + 2\}$, the pixel is also used to carry hidden data and its original value is $a+1$. The embedded data bit is $m = \widetilde{I}_{i,j} - (a + 1)$.
4) If $\widetilde{I}_{i,j} > a+2$, similar to the first case, there is no hidden data and the original value is $\widetilde{I}_{i,j} - 1$.

Fig. 1(a) shows an illustration of this method, in which black values are shifted and red ones are used to embed data. This simple method can illustrate the general mechanism of HS-based RDH algorithms: for data embedding, certain pixel values are shifted to create vacant spaces whereas some others are manipulated to carry hidden data by filling those vacant spaces. Particularly, the pixels whose values can not be shifted or manipulated (i.e., the pixels which cause underflow/overflow) can be distinguished by employing a location map.

In this method, the integer-valued parameter a may be arbitrarily selected from the range of $[1, 253]$. However, there is a better choice. We assume for convenience that image pixels do not take boundary values 0 or 255. Thus there is no underflow/overflow and the EC of this method is $h(a) + h(a + 1)$, where h is the one-dimensional histogram defined by

$$h(t) = \sharp\{(i, j) : I_{i,j} = t\}. \quad (2)$$

Here, the symbol $\sharp$ represents the cardinal number of a set. Moreover, we point out that the expected value of embedding

distortion (in l^2-norm) can be formulated as follows:

$$\mathbb{E}(\|\widetilde{I}-I\|_{l^2}^2)=\sum_{I_{i,j}\in\{a,a+1\}}\frac{1}{2}+\sum_{I_{i,j}\notin\{a,a+1\}}1 =N-\frac{1}{2}\Big(h(a)+h(a+1)\Big) \tag{3}$$

where N is the total number of image pixels. Hence, to maximize EC and minimize embedding distortion, we should take

$$a=\arg\max_{1\le t\le 253}\Big(h(t)+h(t+1)\Big). \tag{4}$$

At the same time, the image-dependent integer a should be communicated to decoder. To this end, we may embed this parameter into host image as auxiliary information. The details will be given later.

Let us now introduce Lee *et al.*'s method [12] where the hidden data is embedded by shifting the difference-histogram. Since the difference-histogram can be generated by projecting the two-dimensional histogram along its main diagonal direction, it is equivalent to shifting the two-dimensional histogram. In this method, for two adjacent pixels $(i,2j-1)$ and $(i,2j)$, the marked pixel values are determined as follows.

$$(\widetilde{I}_{i,2j-1},\widetilde{I}_{i,2j})=\begin{cases}(p,q), & \text{if } (p,q)\in D_0\\ (p+m,q), & \text{if } (p,q)\in D_1\\ (p,q+m), & \text{if } (p,q)\in D_{-1}\\ (p+1,q), & \text{if } (p,q)\in D_t \text{ and } t>1\\ (p,q+1), & \text{if } (p,q)\in D_t \text{ and } t<-1\end{cases} \tag{5}$$

where $(p,q)=(I_{i,2j-1},I_{i,2j})$, $m\in\{0,1\}$ is a data bit to be embedded, and

$$D_t=\{(p,q)\in\mathbb{Z}^2: p-q=t\}. \tag{6}$$

Here, the pixel-value-pairs belonging to the set $\cup_{t\notin\{0,\pm1\}}D_t$ are shifted while the ones in $D_1\cup D_{-1}$ are used to embed data (see Fig. 1(b) for illustration).

Let h^d be the difference-histogram:

$$h^d(t)=\sharp\{(i,j): I_{i,2j-1}-I_{i,2j}=t\} \tag{7}$$

and we assume that image pixels do not take boundary value 255. Then the EC of Lee *et al.*'s method is $h^d(1)+h^d(-1)$. Notice that neighboring pixels in natural image are often correlated. Thus $h^d(1)+h^d(-1)$ is larger compared with the EC of Ni *et al.*'s method. For example, we can embed respectively about 5,400 (Ni *et al.*) and 24 000 (Lee *et al.*'s) bits for 512×512 sized gray-scale image Lena. On the other hand, in Lee *et al.*'s method, at most half of image pixels are modified by 1 in value, thus the resulting PSNR is no less than 51.14 dB. In this light, compared with Ni *et al.*'s method which utilizes one-dimensional histogram, a better performance is achieved by employing two-dimensional histogram.

The aforementioned two methods will be generalized by introducing a unified embedding mechanism in next section.

Finally, we remark that, since one can get a better performance by fully exploiting the spatial redundancy in natural images, it is more desirable to take higher-dimensional histogram to design RDH.

III. Proposed Method

A. Main Idea

In our method, we first divide the host image into non-overlapping blocks such that each block contains n pixels (e.g., $n=1$ for Ni *et al.*'s method, and $n=2$ for Lee *et al.*'s method). Then, an n-dimensional histogram is generated by counting the frequency of the pixel-value-array sized n of each divided block. Finally, data embedding is implemented by modifying the resulting n-dimensional histogram. Notice that the pixel-value-array is an element of $\mathbb{Z}^n$, we then need to divide $\mathbb{Z}^n$ into two disjointed sets, one set is used to carry hidden data by expansion embedding, and the other set is simply shifted to create vacant spaces to ensure the reversibility. We now present our new approach.

Let S and T be a partition of $\mathbb{Z}^n$: $S\cup T=\mathbb{Z}^n$ and $S\cap T=\emptyset$. Suppose that three functions $g: T\to\mathbb{Z}^n$, $f_0: S\to\mathbb{Z}^n$ and $f_1: S\to\mathbb{Z}^n$ satisfy the following conditions:

C1: The functions g, f_0 and f_1 are injective.

C2: The sets $g(T)$, $f_0(S)$ and $f_1(S)$ are disjointed with each other.

Here, g is called "shifting function" and will be used to shift pixel values, f_0 and f_1 are called "embedding functions" and will be used to embed data. More specifically, each block with value $\mathbf{x}\in T$ will be shifted to $g(\mathbf{x})$, and the block with value $\mathbf{x}\in S$ will be expanded to either $f_0(\mathbf{x})$ or $f_1(\mathbf{x})$ to carry one data bit. The shifting and embedding functions will give a HS-based RDH algorithm where the reversibility can be guaranteed by the conditions *C1* and *C2*.

The underflow/overflow is an inevitable problem of RDH, i.e., for gray-scale image, the shifted and expanded values should be restricted in the range of $[0,255]$. To deal with this, the above defined sets T and S need be further processed. Let

$$A_n=\{\mathbf{x}=(x_1,\ldots,x_n)\in\mathbb{Z}^n: 0\le x_i\le 255\} \tag{8}$$

be the set of all pixel-value-arrays of length n of gray-scale image. We define

$$T_s=A_n\cap g^{-1}(A_n) \tag{9}$$

$$S_e=A_n\cap f_0^{-1}(A_n)\cap f_1^{-1}(A_n) \tag{10}$$

$$T_{u,o}=A_n\cap T-T_s \tag{11}$$

$$S_{u,o}=A_n\cap S-S_e. \tag{12}$$

The sub-indices "s", "e" and "u,o" mean "shift", "embed" and "underflow/overflow", respectively. Obviously, the four sets T_s, S_e, $T_{u,o}$ and $S_{u,o}$ are disjointed with each other and constitute a partition of A_n, i.e.,

$$A_n=T_s\cup S_e\cup T_{u,o}\cup S_{u,o}. \tag{13}$$

Moreover, the sets $g(T_s)$, $f_0(S_e)$ and $f_1(S_e)$ are contained in A_n and the condition *C2* ensures that they are also disjointed.

By definitions (9)-(12), each block with value $\mathbf{x}\in T_s$ will be shifted, each block with value $\mathbf{x}\in S_e$ will be expanded to carry one data bit, and the block with value $\mathbf{x}\in T_{u,o}\cup S_{u,o}$ will remain unchanged since it cannot be shifted or expanded due to undeflow/overflow.

Before further describing our method, we give an example for better understanding the definitions above. More examples will be given in Section IV. Take $n = 1$ and for an integer $a \in \{1,\ldots,253\}$, we define

$$\begin{cases} S = \{a, a+1\} \text{ and } T = \mathbb{Z} - S \\ g(x) = \begin{cases} x-1, \text{ if } x < a \\ x+1, \text{ if } x > a+1 \end{cases} \\ f_0(a) = a, \ \ f_0(a+1) = a+1 \\ f_1(a) = a-1, \ \ f_1(a+1) = a+2. \end{cases} \tag{14}$$

One can verify that (g, f_0, f_1) satisfy the conditions *C1* and *C2*. Particularly, the sets defined in (9)-(12) can be detailed as follows: $T_s = \{1,\ldots,a-1\} \cup \{a+2,\ldots,254\}$, $S_e = \{a, a+1\}$, $T_{u,o} = \{0, 255\}$, and $S_{u,o} = \emptyset$. We will see that this specific construction corresponds the modified version of Ni *et al.*'s method introduced in Section II.

We now briefly describe how to reversibly embed data by using the shifting and embedding functions (see Fig. 2 for illustration). Consider a pixel block of length n whose value is $\mathbf{x} \in A_n$.

1) If $\mathbf{x} \in T_s$, the block does not carry any hidden data and its value is simply shifted to $g(\mathbf{x}) \in A_n$. For instance, in the example (14), the pixel value in $T_s = \{1,\ldots,a-1\} \cup \{a+2,\ldots,254\}$ needs to be shifted and the shifted value belongs to the set $g(T_s) = \{0,\ldots,a-2\} \cup \{a+3,\ldots,255\}$.
2) If $\mathbf{x} \in S_e$, the new pixel value is taken as $f_m(\mathbf{x}) \in A_n$ where $m \in \{0, 1\}$ is the corresponding data bit to be embedded. In this situation, one data bit is embedded into the block. Since the sets $g(T_s)$, $f_0(S_e)$ and $f_1(S_e)$ are disjointed with each other, decoder can distinguish the blocks used for shifting from those for embedding data. As a result, by the condition *C1*, decoder can recover the original pixel value and extract the embedded data bit. For instance, in the example (14), the pixel value in $S_e = \{a, a+1\}$ is expanded to carry one data bit, and the expanded value belongs to the set $f_0(S_e) \cup f_1(S_e) = \{a, a+1\} \cup \{a-1, a+2\} = \{a-1, a, a+1, a+2\}$.
3) If $\mathbf{x} \in T_{u,o}$, we know that $g(\mathbf{x}) \notin A_n$. In this case, to prevent underflow/overflow, we do nothing with $\mathbf{x}$. Regarding the example (14), the pixel will not be changed if its value belongs to $T_{u,o} = \{0, 255\}$.
4) If $\mathbf{x} \in S_{u,o}$, we see either $f_0(\mathbf{x}) \notin A_n$ or $f_1(\mathbf{x}) \notin A_n$. In this case, the block will also remain unchanged. Yet, the set $T_{u,o} \cup S_{u,o}$ may overlap with $g(T_s) \cup f_0(S_e) \cup f_1(S_e)$. Then we use a location map to record the locations of blocks whose values belong to $T_{u,o} \cup S_{u,o}$. In the example (14), the pixel whose value is 0 or 255 needs to be recorded in the location map.

In summary, with shifting and embedding functions, we can obtain a HS-based RDH algorithm. The detailed embedding and extraction procedures are given below.

We take $TS_u = T_{u,o} \cup S_{u,o}$ for simplicity in the following context.

B. Data Embedding

The embedding procedure contains several steps. First, after dividing the host image into non-overlapping blocks, the blocks are further divided into three parts to get I_1, I_2 and I_3. Then, by using shifting and embedding functions, embed the hidden data into I_1 and I_3. Next, by using LSB replacement, embed the location map which records the underflow/overflow locations into I_1. Notice that, before replacing LSBs, the original LSBs of I_1 should be recorded into a LSB sequence. Finally, embed the LSB sequence into I_2 using shifting and embedding functions.

Here, the partition of three parts is to solve the underflow/overflow problem by embedding the location map into the host image. The part I_1 is double embedded to embed first the hidden data (using shifting and embedding functions) and then the location map (using LSB replacement).

The detailed data embedding procedure is described as bellow.

Step 1: Divide the host image into k non-overlapping blocks $\{X_1,\ldots,X_k\}$ such that each X_i contains n pixels. Assume the value of X_i is $\mathbf{x}_i \in A_n$.

Step 2: Define the location map LM: $LM(i) = 0$ if $\mathbf{x}_i \in T_s \cup S_e$, and $LM(i) = 1$ if $\mathbf{x}_i \in TS_u$. Clearly, LM is a binary sequence of length k. Denote $k' = \lceil \log_2(k+1) \rceil$, where $\lceil \cdot \rceil$ is the ceiling function. Take $l = \sharp\{i : LM(i) = 1\}$ which is the amount of underflow/overflow blocks. Then we define a binary sequence LM_c of length $l_c = (l+1)k'$ to record all underflow/overflow locations.

1) $(LM_c(1),\ldots,LM_c(k'))$ is the binary representation of l.
2) For each $j \in \{1,\ldots,l\}$, $(LM_c(jk'+1),\ldots,LM_c(jk'+k'))$ is the binary representation of i, where i is the j-th index such that $LM(i) = 1$.

Step 3: Divide the k blocks into three parts to get I_1, I_2 and I_3.

1) $I_1 = \{X_1,\ldots,X_{k_1}\}$ with $k_1 = \lceil \frac{l_c}{n} \rceil$.
2) $I_2 = \{X_{k_1+1},\ldots,X_{k_1+k_2}\}$ such that it contains exactly l_c embeddable blocks. Here, a block is called "embeddable" if its value belongs to S_e.
3) $I_3 = \{X_{k_1+k_2+1},\ldots,X_k\}$ is the set of rest blocks.

Step 4: Embed the hidden data into I_1 and I_3, i.e., for any $i \in \{1,\ldots,k_1, k_1+k_2+1,\ldots,k\}$.

1) If $\mathbf{x}_i \in T_s$, replace the value of X_i by $g(\mathbf{x}_i)$.
2) If $\mathbf{x}_i \in S_e$, replace the value of X_i by $f_m(\mathbf{x}_i)$, where $m \in \{0, 1\}$ is the data bit to be embedded.
3) If $\mathbf{x}_i \in TS_u$, we do nothing with X_i.

Step 5: Record LSBs of the first l_c pixels of I_1 to get a binary sequence S_{LSB} (recall that I_1 contains $nk_1 = n\lceil \frac{l_c}{n} \rceil \geq l_c$ pixels). Then replace these LSBs by the sequence LM_c defined in *Step 2*.

Step 6: Embed the sequence S_{LSB} into I_2 in the same way as *Step 4*. Since the length of S_{LSB} is l_c, S_{LSB} can be embedded exactly into the embeddable blocks of I_2. Finally, we get the marked image.

The core of this procedure is *Step 4* and the partition in *Step 3* is to deal with the underflow/overflow problem.

It should be mentioned that there is another commonly used way to encode the location map: in *Step 2*, one can get LM_c by losslessly compressing LM. However, in HS-based RDH algorithms, there are usually only a few blocks which may cause underflow/overflow. We then prefer to record those

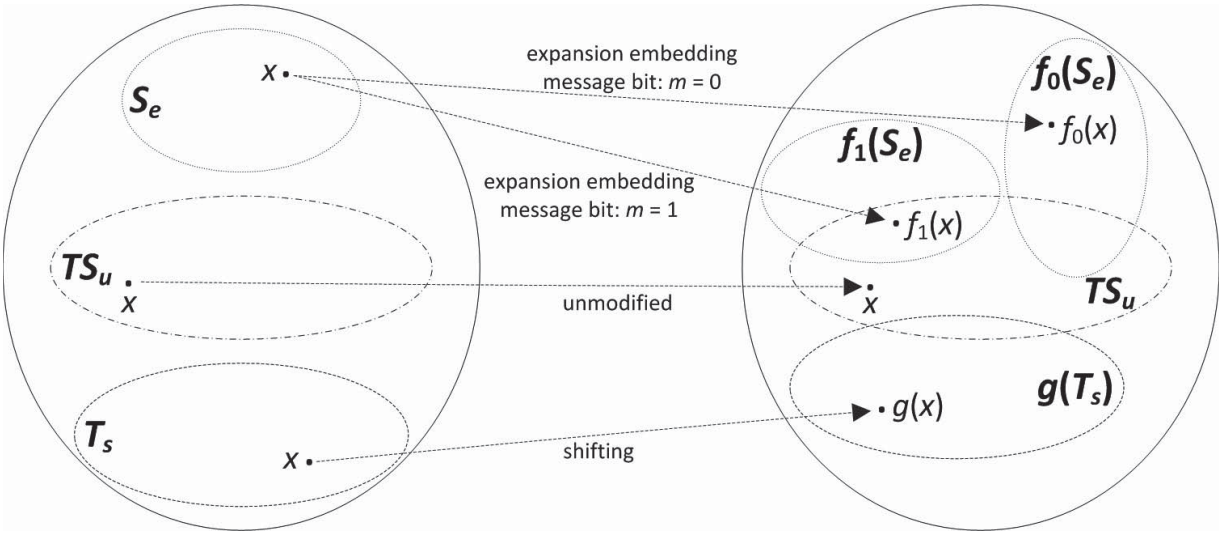

Fig. 2. Illustration of the data embedding procedure by using shifting and embedding functions. After data embedding, the sets $g(T_s)$, $f_0(S_e)$ and $f_1(S_e)$ are disjointed with each other, but the two sets $TS_u = T_{u,o} \cup S_{u,o}$ and $g(T_s) \cup f_0(S_e) \cup f_1(S_e)$ may overlapped, and a location map will be used to record the locations of pixel blocks whose values belong to TS_u.

problematic locations rather than compressing the location map because the latter solution is time-consuming.

C. Data Extraction

The data extraction procedure also contains several steps. First, the same as the data embedding, divide the marked image blocks into three parts to get I_1, I_2 and I_3. Then, determine the location map by reading LSBs of I_1. Next, according to the location map and by using shifting and embedding functions, determine the LSB sequence (defined in *Step 5* of data embedding) by extracting data from I_2, and then replace the LSBs of I_1 by the extracted LSB sequence. Finally, extract the embedded data from I_1 and I_3. Notice that, using shifting and embedding functions, the image restoration can be realized simultaneously with the data extraction.

The detailed data extraction procedure is described as bellow.

Step 1: The same as *Step 1* of data embedding, divide the marked image into k non-overlapping blocks $\{Y_1, \ldots, Y_k\}$. Assume the value of Y_i is $\mathbf{y}_i \in A_n$.

Step 2: Firstly, determine the amount of problematic locations, l, by reading LSBs of the first $k' = \lceil \log_2(k+1) \rceil$ pixels. Secondly, read LSBs of the first $l_c = (l+1)k'$ pixels to get the sequence LM_c. Then we can get the location map LM. Finally, with $k_1 = \lceil \frac{l_c}{n} \rceil$, LM, and by identifying embeddable blocks, we can obtain the same partition as defined in *Step 3* of data embedding.

Step 3: Extract data from I_2 and recover original pixel values of I_2, i.e., for any $i \in \{k_1 + 1, \ldots, k_1 + k_2\}$.

1) If $LM(i) = 0$ and $\mathbf{y}_i \in g(T_s)$, the original pixel value is $g^{-1}(\mathbf{y}_i)$ and there is no embedded data.
2) If $LM(i) = 0$ and $\mathbf{y}_i \in f_m(S_e)$ holds for a certain $m \in \{0, 1\}$, the original pixel value is $f_m^{-1}(\mathbf{y}_i)$ and the embedded data bit is m.
3) If $LM(i) = 1$, the original pixel value is $\mathbf{y}_i$ itself and there is no embedded data.

The sequence S_{LSB} defined in *Step 5* of data embedding is extracted in this step.

Step 4: Replace LSBs of the first l_c pixels of I_1 by S_{LSB}.

Step 5: Extract the embedded hidden data and recover original pixel values in $I_1 \cup I_3$ in the same way as *Step 3*. Finally, the hidden data is extracted and the original image is restored.

D. Embedding Capacity

According to the data embedding procedure, the EC of our method can be formulated as:

$$\sum_{\mathbf{x} \in S_e} h_n(\mathbf{x}) - \lceil \log_2(k+1) \rceil \sum_{\mathbf{x} \in TS_u} h_n(\mathbf{x}) - \lceil \log_2(k+1) \rceil \quad (15)$$

where h_n is the n-dimensional histogram: $h_n(\mathbf{x}) = \sharp\{i : \mathbf{x}_i = \mathbf{x}\}$.

Notice that T_s, S_e, TS_u constitute a partition of A_n. Then,

$$\sharp A_n = \sharp T_s + \sharp S_e + \sharp TS_u. \quad (16)$$

Moreover, recall that $g(T_s)$, $f_0(S_e)$ and $f_1(S_e)$ are disjointed with each other and they are contained in A_n. Hence,

$$\sharp A_n \geq \sharp g(T_s) + \sharp f_0(S_e) + \sharp f_1(S_e). \quad (17)$$

By the condition *C1*, we have $\sharp T_s = \sharp g(T_s)$ and $\sharp S_e = \sharp f_0(S_e) = \sharp f_1(S_e)$. Consequently, by comparing (16) with (17), we see that

$$\sharp T_s + \sharp S_e + \sharp TS_u \geq \sharp T_s + \sharp S_e + \sharp S_e. \quad (18)$$

Hence, $\sharp TS_u \geq \sharp S_e$ holds. Thus, reviewing (15), since $\lceil \log_2(k+1) \rceil$ is a positive constant larger than 1, to successfully embed data and increase EC, we should carefully design (S, T) and (g, f_0, f_1) such that $h_n(\mathbf{x})$ is large for $\mathbf{x} \in S_e$ while small for $\mathbf{x} \in TS_u$. That is to say, we should use the most frequent bins in the histogram to embed data and meanwhile avoid underflow/overflow.

E. Extension of the Proposed Framework

In this subsection, as a complement of the proposed embedding mechanism, we discuss how to embed auxiliary information and how to embed a required amount of data bits.

As is known, a RDH algorithm usually depends on some parameters (e.g., the integer a defined in (4)) and these parameters should be communicated to decoder. To this end, we may slightly modify the embedding and extraction procedures. Firstly, we divide host image into two parts to get I_0 and I': I_0 contains a fixed number of pixels and I' is the rest pixels. Secondly, express the parameters in binary form to get a binary sequence and replace the LSBs of I_0 by this sequence. Here, the original LSBs of I_0 should be recorded and then embedded into host image as a part of hidden data. For example, in the modified version of Ni *et al.*'s method, we may reserve 8 pixels (i.e., $\sharp I_0 = 8$) to embed the value of a. Finally, consider I' as an "image" and embed data into it. For decoder, it only needs to read LSBs of I_0 to get the parameters, and then extract the hidden data from I'. In particular, I_0 can be restored by overwriting its LSBs by a certain part of extracted hidden data.

We now discuss how to embed a required amount of data bits which is helpful for practical applications. Actually, in data embedding, we may stop *Step 4* once all data bits are embedded and then move directly to *Step 5*, i.e., we stop shifting and embedding for $i > i_0$, where $i_0 \in \{1,\ldots,k_1, k_1 + k_2 + 1,\ldots,k\}$ is the smallest index such that the blocks $\{X_1,\ldots,X_{k_1}, X_{k_1+k_2+1},\ldots,X_{i_0}\}$ are sufficient to carry all data bits. Here, the rest pixels of $I_1 \cup I_3$ remain unchanged. In this situation, the value of EC should be embedded into host image as auxiliary information. Accordingly, the decoder may read the value of EC and stop *Step 5* of data extraction after all data bits are extracted. The rest pixels of $I_1 \cup I_3$ can be restored as themselves.

IV. Examples of Shifting and Embedding Functions

Examples of the proposed general construction are presented in this section by introducing definitions of (S, T) and (g, f_0, f_1). In each example, one can prove that the shifting and embedding functions satisfy conditions *C1* and *C2*. The corresponding RDH algorithms of these examples are methods presented in the literature, and the only difference between existing methods and our framework is the treatment of location map.

A. Lee et al.'s Method [12]

Take $n = 2$ and we define

1) $S = \{(x, y) \in \mathbb{Z}^2 : |x - y| = 1\}$ and $T = \mathbb{Z}^2 - S$.
2) For $(x, y) \in T$,

$$g(x,y) = \begin{cases} (x,y), & \text{if } |x-y| < 1, \\ (x+1,y), & \text{if } x-y > 1, \\ (x,y+1), & \text{if } x-y < -1. \end{cases} \tag{19}$$

3) For $(x, y) \in S$ and $m \in \{0, 1\}$,

$$f_m(x,y) = \begin{cases} (x+m,y), & \text{if } x-y = 1, \\ (x,y+m), & \text{if } x-y = -1. \end{cases} \tag{20}$$

This algorithm is exactly Lee *et al.*'s method [12].

B. Weng et al.'s Method [33]

Take $n = 2$ and for an integer $t > 0$, we define

1) $S = \{(x, y) \in \mathbb{Z}^2 : |x - y| < t\}$ and $T = \mathbb{Z}^2 - S$.
2) For $(x, y) \in T$,

$$g(x,y) = \begin{cases} (x + \lfloor \frac{t+1}{2} \rfloor, y - \lfloor \frac{t+1}{2} \rfloor), & \text{if } x-y \geq t, \\ (x - \lfloor \frac{t+1}{2} \rfloor, y + \lfloor \frac{t+1}{2} \rfloor), & \text{if } x-y \leq -t. \end{cases} \tag{21}$$

3) For $(x, y) \in S$ and $m \in \{0, 1\}$,

$$f_m(x,y) = \left(x + \left\lfloor \frac{x-y}{2} \right\rfloor + m, y - \left\lfloor \frac{x-y}{2} \right\rfloor - m\right). \tag{22}$$

This algorithm is the integer-transform-based method of Weng *et al.* [33]. Here, $\lfloor \cdot \rfloor$ is the floor function.

C. D2 Algorithm of Thodi and Rodriguez [10]

Take $n = 2$ and for an integer $t > 0$, we define

1) $S = \{(x, y) \in \mathbb{Z}^2 : -t \leq x - y < t\}$ and $T = \mathbb{Z}^2 - S$.
2) For $(x, y) \in T$,

$$g(x,y) = \begin{cases} (x + \lfloor \frac{t+1}{2} \rfloor, y - \lfloor \frac{t}{2} \rfloor), & \text{if } x-y \geq t \text{ and } x-y \text{ is even}, \\ (x + \lfloor \frac{t}{2} \rfloor, y - \lfloor \frac{t+1}{2} \rfloor), & \text{if } x-y \geq t \text{ and } x-y \text{ is odd}, \\ (x - \lfloor \frac{t}{2} \rfloor, y + \lfloor \frac{t+1}{2} \rfloor), & \text{if } x-y < -t \text{ and } x-y \text{ is even}, \\ (x - \lfloor \frac{t+1}{2} \rfloor, y + \lfloor \frac{t}{2} \rfloor), & \text{if } x-y < -t \text{ and } x-y \text{ is odd}. \end{cases} \tag{23}$$

3) For $(x, y) \in S$ and $m \in \{0, 1\}$,

$$f_m(x,y) = \left(2x - \left\lceil \frac{x+y}{2} \right\rceil + m, 2y - \left\lceil \frac{x+y}{2} \right\rceil\right). \tag{24}$$

For $(x, y) \in \mathbb{Z}^2$, let $l(x, y) = \lfloor \frac{x+y}{2} \rfloor$ be the integer average of x and y, and $h(x, y) = x - y$ be the difference value. Denote the shifted or embedded pixel-value-pair (i.e., $g(x, y)$ or $f_m(x, y)$) as (x', y'). We can verify that the following relations hold

$$l(x', y') = l(x, y) \tag{25}$$

$$h(x', y') = \begin{cases} 2h(x,y) + m, & \text{if } -t \leq x-y < t \\ h(x,y) + t, & \text{if } x-y \geq t \\ h(x,y) - t, & \text{if } x-y < -t. \end{cases} \tag{26}$$

With those equations, we see that this example is the D2 algorithm of Thodi and Rodriguez [10]. In this light, the DE algorithm can be included in the proposed framework.

D. Coatrieux et al.'s Method [1]

We use four-dimensional histogram in this example. The corresponding algorithm is the method of Coatrieux *et al.* [1]. Let $\mathbf{v} = (1, -1, -1, 1)$ and we define

1) $S = \{\mathbf{x} = (x_1, x_2, x_3, x_4) \in \mathbb{Z}^4 : |d_{\mathbf{x}}| < 1\}$ and $T = \mathbb{Z}^4 - S$, where $d_{\mathbf{x}} = (x_2 + x_3 - 2x_1)/4$.
2) For $\mathbf{x} \in T$,

$$g(\mathbf{x}) = \begin{cases} \mathbf{x} - \mathbf{v}, & \text{if } d_{\mathbf{x}} \geq 1, \\ \mathbf{x} + \mathbf{v}, & \text{if } d_{\mathbf{x}} \leq -1. \end{cases} \tag{27}$$

3) For $\mathbf{x} \in S$ and $m \in \{0, 1\}$,

$$f_m(\mathbf{x}) = \mathbf{x} + (2m - 1)\mathbf{v}. \tag{28}$$

E. Hu et al.'s Method [19]

Take $n = 4$ and for a 2×2 block $\mathbf{x} = (x_1, x_2, x_3, x_4)$ (see Fig. 3(a)), consider the context adaptive predictor MED (median edge detector [34]):

$$\widehat{x}_1 = \begin{cases} \min(x_2, x_3), & \text{if } x_4 \geq \max(x_2, x_3) \\ \max(x_2, x_3), & \text{if } x_4 \leq \min(x_2, x_3) \\ x_2 + x_3 - x_4, & \text{otherwise.} \end{cases} \tag{29}$$

In this way, $\widehat{x}_1$ is a prediction of x_1. Suppose $e_1 = x_1 - \widehat{x}_1$ is the prediction-error.

For an integer $t > 0$, take $t_l = \lfloor \frac{t}{2} \rfloor$ and $t_r = \lceil \frac{t}{2} \rceil$. We then define

1) $S = \{\mathbf{x} \in \mathbb{Z}^4 : -t_l \leq e_1 < t_r\}$ and $T = \mathbb{Z}^4 - S$.
2) For $\mathbf{x} \in T$,

$$g(\mathbf{x}) = \begin{cases} (x_1 + t_r, x_2, x_3, x_4), & \text{if } e_1 \geq t_r, \\ (x_1 - t_l, x_2, x_3, x_4), & \text{if } e_1 < -t_l. \end{cases} \tag{30}$$

3) For $\mathbf{x} \in S$ and $m \in \{0, 1\}$,

$$f_m(\mathbf{x}) = (x_1 + e_1 + m, x_2, x_3, x_4). \tag{31}$$

This design yields essentially Hu *et al.*'s prediction-error expansion (PEE) based method [19].

It should be noticed that there is a little difference between Hu *et al.*'s method and our construction in this example. In their method, image pixels are ordered and the hidden data is embedded pixel by pixel. But in our framework, the host image is first divided into non-overlapping blocks and then we embed data block by block. More specifically, in our method, for each block $\mathbf{x} = (x_1, x_2, x_3, x_4)$, only x_1 is altered and at most one data bit will be embedded into it, while other pixels x_2, x_3, x_4 remain unchanged. However, we may use multiple-pass embedding so that each pixel in the host image can be embedded. We illustrate the idea as follows.

Suppose the required EC is c and it is a multiple of 4. As shown in Fig. 4(a), image pixels are labeled as "A", "B", "C" or "D". For 1st embedding layer, we divide host image I into 2×2 blocks (see Fig. 4(b)) and embed $c/4$ bits into I to get $\widetilde{I}^1$. Notice that, to minimize embedding distortion, the parameter t is taken as the smallest one such that it is capable to embed the required payload, and the EC formulation in (15) can be utilized to determine this parameter. Here, only the pixels labeled "A" may be altered. For 2nd embedding layer, $\widetilde{I}^1$ is divided into 2×2 blocks (see Fig. 4(c), noticing that the partition here is different from the 1st embedding layer), and we embed $c/4$ bits into $\widetilde{I}^1$ to get $\widetilde{I}^2$. In this case, only the pixels labeled "B" may be altered. Similarly, we do the 3rd and 4th layer of embedding, to get $\widetilde{I}^3$ and $\widetilde{I}^4$. Finally, c bits are embedded into I and the marked image is $\widetilde{I}^4$.

V. Two Novel HS-Based RDH Algorithms

A. Algorithm I

In this subsection, we present a novel HS-based RDH algorithm by exploiting nine-dimensional histogram. In this way, we see that by incorporating existing PEE [19] and pixel selection [20] techniques into the proposed framework, one can achieve an excellent performance.

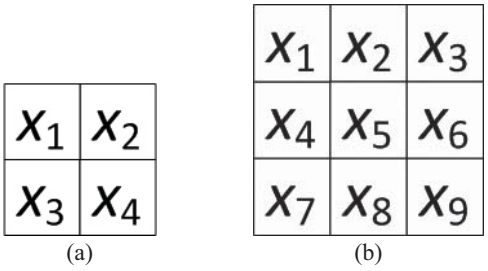

Fig. 3. Pixel block.

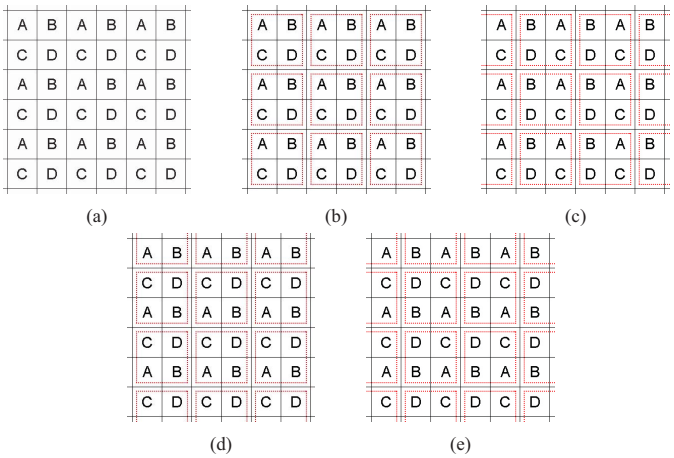

Fig. 4. (a) Labels of image pixels. For the first, second, third, and fourth embedding layers, the pixels are divided into 2×2 blocks as shown in (b)–(e), respectively. Each block is bounded by a red rectangle.

For a 3×3 block $\mathbf{x} = (x_1, \ldots, x_9)$ (see Fig. 3(b)), we take the following linear predictor with nonuniform weight

$$\widehat{x}_5 = \frac{1}{16}(x_1 + x_3 + x_7 + x_9) + \frac{3}{16}(x_2 + x_4 + x_6 + x_8) \tag{32}$$

to predict x_5. The prediction-error is denoted as $e_5 = x_5 - \widehat{x}_5$. Here, unlike MED which uses only half-enclosing casual pixels to make estimation, full-enclosing pixels are exploited in the above predictor and a better prediction result can be expected.

As we have discussed in our previous work [20], utilizing smooth pixels for reversible data embedding whereas ignoring the noisy ones will significantly reduce the embedding distortion in RDH. Then we take the following function

$$\begin{aligned} C(\mathbf{x}) = &\max\{x_1, \ldots, x_4, x_6, \ldots, x_9\} \\ &- \min\{x_1, \ldots, x_4, x_6, \ldots, x_9\} \end{aligned} \tag{33}$$

to measure the local complexity of pixel x_5 and we will use an integer-valued parameter s to select smooth pixels.

For an integer $t > 0$, take $t_l = \lfloor \frac{t}{2} \rfloor$ and $t_r = \lceil \frac{t}{2} \rceil$. We then define

1) $S = \{\mathbf{x} \in \mathbb{Z}^9 : -t_l \leq e_5 < t_r, C(\mathbf{x}) < s\}$ and $T = \mathbb{Z}^9 - S$.
2) For $\mathbf{x} \in T$,

$$g(\mathbf{x}) = \begin{cases} (x_1, \ldots, x_4, x_5 + t_r, x_6, \ldots, x_9), & \text{if } e_5 \geq t_r \text{ and } C(\mathbf{x}) < s, \\ (x_1, \ldots, x_4, x_5 - t_l, x_6, \ldots, \mathbf{x}_9), & \text{if } e_5 < -t_l \text{ and } C(\mathbf{x}) < s, \\ \mathbf{x}, & \text{if } C(\mathbf{x}) \geq s. \end{cases} \tag{34}$$

TABLE I

COMPARISON OF PSNR (IN DB) BETWEEN THE PROPOSED ALGORITHM DESCRIBED IN SECTION V-A AND FOUR METHODS OF HU *et al.* [19], LUO *et al.* [29], LI *et al.* [20], AND HONG [35], FOR AN ER OF 0.5 BPP

Image	Hu *et al.* [19]	Luo *et al.* [29]	Li *et al.* [20]	Hong [35]	Proposed algorithm I
Lena	40.73	41.33	42.37	41.25	43.24
Baboon	30.65	29.47	31.42	31.10	32.21
Airplane (F-16)	44.21	43.95	45.97	44.84	47.21
Peppers	37.01	37.60	38.79	37.58	39.92
Sailboat on lake	35.33	35.67	36.63	35.66	37.77
Fishing boat	36.45	36.05	37.82	36.79	38.61
Average	**37.40**	**37.35**	**38.83**	**37.87**	**39.83**

TABLE II

COMPARISON OF PSNR (IN DB) BETWEEN THE PROPOSED ALGORITHM DESCRIBED IN SECTION V-B AND FOUR METHODS OF HU *et al.* [19], LUO *et al.* [29], LI *et al.* [20], AND HONG [35], FOR AN EC OF 10 000 BITS

Image	Hu *et al.* [19]	Luo *et al.* [29]	Li *et al.* [20]	Hong [35]	Proposed algorithm II
Lena	56.46	57.31	58.20	58.78	59.37
Baboon	50.29	51.06	54.03	53.26	54.41
Airplane (F-16)	57.24	57.97	61.26	62.08	62.65
Peppers	53.36	55.29	56.12	56.07	56.89
Sailboat on lake	53.62	55.32	58.22	57.82	58.27
Fishing boat	53.09	54.04	55.52	56.64	57.16
Average	**54.01**	**55.17**	**57.23**	**57.44**	**58.13**

TABLE III

COMPARISON OF PSNR (IN DB) BETWEEN THE PROPOSED ALGORITHM DESCRIBED IN SECTION V-B AND FOUR METHODS OF HU *et al.* [19], LUO *et al.* [29], LI *et al.* [20], AND HONG [35], FOR AN EC OF 20 000 BITS. THE RESULT FOR *Baboon* IS NOT PRESENTED HERE SINCE OUR METHOD CANNOT EMBED 20 000 BITS INTO THIS IMAGE

Image	Hu *et al.* [19]	Luo *et al.* [29]	Li *et al.* [20]	Hong [35]	Proposed algorithm II
Lena	52.86	53.83	54.82	54.92	55.93
Airplane (F-16)	54.63	55.43	56.84	58.58	59.26
Peppers	50.64	52.19	52.55	52.16	53.31
Sailboat on lake	50.69	52.17	53.25	52.03	53.19
Fishing boat	50.25	51.19	52.43	52.26	53.05
Average	**51.81**	**52.96**	**53.98**	**53.99**	**54.95**

3) For $\mathbf{x} \in S$ and $m \in \{0, 1\}$,

$$f_m(\mathbf{x}) = (x_1, \ldots, x_4, x_5 + \lfloor e_5 \rfloor + m, x_6, \ldots, x_9). \tag{35}$$

Similar to the example in Section IV-E, to minimize the embedding distortion, the parameter s is first fixed as its maximum 256 and the parameter t is taken as the smallest integer such that it is capable to embed the required payload. When t is determined, to take advantage of smooth pixels as much as possible, we vary the parameter s and then take it as the smallest integer such that it is capable to embed the required payload. Finally, we use multiple-pass embedding as described in Section IV-E such that each pixel in the host image can be embedded.

Fig. 5 shows the performance evaluation of this algorithm by comparing it with four recent algorithms of Hu *et al.* [19], Luo *et al.* [29], Li *et al.* [20], and Hong [35], for six standard 512×512 sized gray-scale images: Lena, Baboon, Airplane, Peppers, Sailboat on lake, and Fishing boat. All these images are downloaded from the USC-SIPI database[1]. According to the figure, one can observe that by utilizing our framework and existing techniques, this novel algorithm achieves a better performance compared with those state-of-the-art works. Except for Airplane at ER = 0.9 BPP, it can provide a larger PSNR whatever the test image or ER is. Moreover, referring to Table I, it can be seen that this novel algorithm improves the state-of-the-art works by at least 1 dB in average for an ER of 0.5 BPP.

B. Algorithm II

In this subsection, by exploiting nine-dimensional histogram also, we present another novel HS-based RDH algorithm based on a new prediction strategy. Specifically, referring to Fig. 3(b), we will use x_2 and x_4 to predict x_1 in the following way

$$\widehat{x_1} = \begin{cases} p, & \text{if } x_1 \geq p \\ q - 1, & \text{if } x_1 < q \end{cases} \tag{36}$$

[1]http://sipi.usc.edu/database/database.php?volume=misc

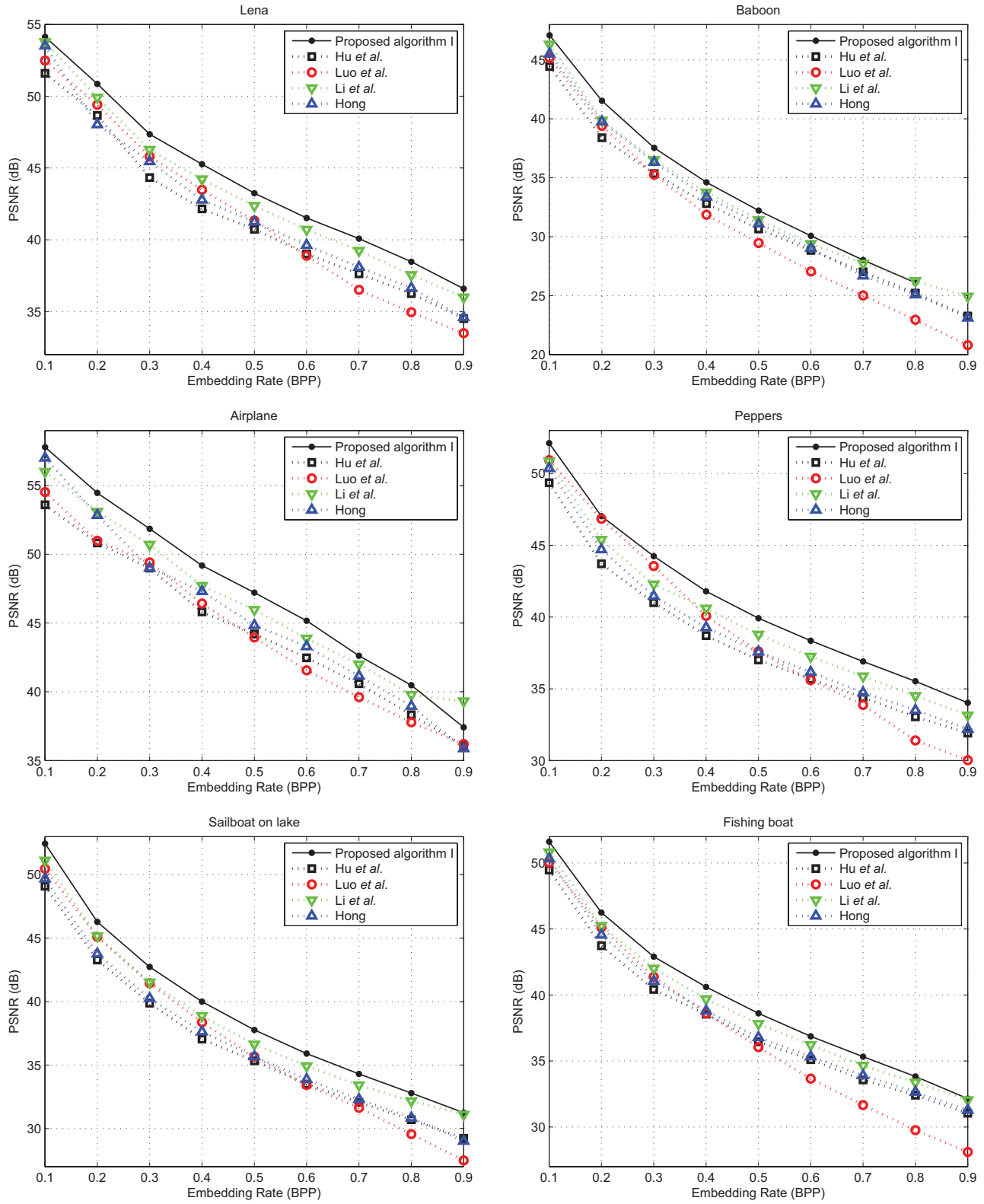

Fig. 5. Performance comparison between the proposed algorithm introduced in Section V-A and algorithms of Hu *et al.* [19], Luo *et al.* [29], Li *et al.* [20], and Hong [35].

where $p = \max\{x_2, x_4\}$, $q = \min\{x_2, x_4\}$, and $p \geq q$ holds. Here, only the pixels satisfying $x_1 \geq p$ or $x_1 < q$ will be predicted and used for data embedding, whereas the ones satisfying $q \leq x_1 < p$ are ignored. Notice that, unlike the usual predictors such as MED and (32), the pixel x_1 itself is involved in prediction. In this situation, to ensure the reversibility, the condition on x_1, $x_1 \geq p$ (as well as $x_1 < q$), should be hold after data embedding. We then take the following shifting and embedding functions

1) $S = S_1 \cup S_2$ and $T = \mathbb{Z}^9 - S$, where $S_1 = \{\mathbf{x} \in \mathbb{Z}^9 : x_1 - p = 0, C(\mathbf{x}) < s\}$ and $S_2 = \{\mathbf{x} \in \mathbb{Z}^9 : x_1 - q = -1, C(\mathbf{x}) < s\}$ are two disjointed sets, and $C(\mathbf{x})$ is defined as

$$C(\mathbf{x}) = \max\{x_2, \ldots, x_9\} - \min\{x_2, \ldots, x_9\} \quad (37)$$

to measure the local complexity. Here, s is used to select smooth pixels, and this parameter is taken as the smallest positive integer such that it is capable to embed the required payload.

2) For $\mathbf{x} \in T$,

$$g(\mathbf{x}) = \begin{cases} (x_1 + 1, x_2, \ldots, x_9), & \text{if } x_1 > p \text{ and } C(\mathbf{x}) < s, \\ (x_1 - 1, x_2, \ldots, x_9), & \text{if } x_1 < q - 1 \text{ and } C(\mathbf{x}) < s, \\ \mathbf{x}, & \text{if } q \le x_1 < p \text{ or } C(\mathbf{x}) \ge s. \end{cases} \quad (38)$$

3) For $\mathbf{x} \in S$ and $m \in \{0, 1\}$,

$$f_m(\mathbf{x}) = \begin{cases} (x_1 + m, x_2, \ldots, x_9), & \text{if } \mathbf{x} \in S_1, \\ (x_1 - m, x_2, \ldots, x_9), & \text{if } \mathbf{x} \in S_2. \end{cases} \quad (39)$$

Clearly, by this design, if $C(\mathbf{x}) < s$, only x_1 in the pixel block is modified according to the the following cases

1) it is expanded to x_1 or $x_1 + 1$ to carry one data bit when $x_1 = p$,
2) it is shifted to $x_1 + 1$ when $x_1 > p$,
3) it is expanded to x_1 or $x_1 - 1$ to carry one data bit when $x_1 = q - 1$,
4) it is shifted to $x_1 - 1$ when $x_1 < q - 1$.

Thus the condition $x_1 \ge p$ (as well as $x_1 < q$) is also satisfied after data embedding. Moreover, we use multiple-pass embedding as described in Section IV-E such that each pixel in the host image can be embedded. Since this algorithm modifies each pixel value at most by 1, it provides only low EC.

This algorithm is evaluated by comparing it with four recent algorithms of Hu *et al.* [19], Luo *et al.* [29], Li *et al.* [20], and Hong [35]. Referring to Tables II-III, one can observe that this new algorithm outperforms the state-of-the-art works.

With the two novel and effective examples introduced in this section, the universality of our framework is validated. It demonstrates that our framework is capable of constructing powerful RDH algorithms. It is expected that excellent performance can be achieved according to our framework by using high-dimensional histogram and carefully designed shifting and embedding functions.

VI. Conclusion

In this paper, by revisiting existing algorithms, a general framework to construct HS-based RDH is proposed. According to our framework, to obtain a RDH algorithm, one just needs to define the shifting and embedding functions. This work will facilitate the design of RDH. Furthermore, by incorporating our framework with the PEE and pixel selection techniques, two novel RDH algorithms are also introduced. These algorithms can achieve a better performance compared with the state-of-the-art works. So the proposed framework has a potential to provide excellent RDH algorithms. However, thought the proposed framework may design different RDH algorithms, it has also limitations. Some HS-based algorithms such as the one based on adaptive embedding [20] and the location-map-free methods [24], [36] cannot be derived by the proposed framework.

In future, to push forward the capacity-distortion behavior of RDH, more meaningful shifting and embedding functions are expected. Moreover, since the typical two-dimensional histogram based methods [10], [12], [33] are special cases of the proposed framework, a direct question is, based on two-dimensional histogram (i.e., by taking $n = 2$ in our framework), how to determine the optimized shifting and embedding functions such that the embedding distortion is minimized for a given EC. This is also an interesting direction for future work.

References

[1] G. Coatrieux, C. L. Guillou, J. M. Cauvin, and C. Roux, "Reversible watermarking for knowledge digest embedding and reliability control in medical images," *IEEE Trans. Inf. Technol. Biomed.*, vol. 13, no. 2, pp. 158–165, Mar. 2009.

[2] S. Lee, C. D. Yoo, and T. Kalker, "Reversible image watermarking based on integer-to-integer wavelet transform," *IEEE Trans. Inf. Forens. Security*, vol. 2, no. 3, pp. 321–330, Sep. 2007.

[3] R. Li, O. C. Au, C. K. M. Yuk, S. Yip, and T. Chan, "Enhanced image trans-coding using reversible data hiding," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2007, pp. 1273–1276.

[4] R. Caldelli, F. Filippini, and R. Becarelli, "Reversible watermarking techniques: An overview and a classification," *Eur. Assoc. Signal Process. J. Inf. Security*, vol. 2010, no. 2, pp. 1–19, 2010.

[5] J. Fridrich, M. Goljan, and R. Du, "Lossless data embedding-new paradigm in digital watermarking," *Eur. Assoc. Signal Process. J. Appl. Signal Process.*, vol. 2002, no. 2, pp. 185–196, Feb. 2002.

[6] T. Kalker and F. M. J. Willems, "Capacity bounds and constructions for reversible data hiding," *Security Watermarking Multimedia Contents V*, vol. 5020, pp. 604–611, Jun. 2003.

[7] M. U. Celik, G. Sharma, A. M. Tekalp, and E. Saber, "Lossless generalized-LSB data embedding," *IEEE Trans. Image Process.*, vol. 14, no. 2, pp. 253–266, Feb. 2005.

[8] J. Tian, "Reversible data embedding using a difference expansion," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 8, pp. 890–896, Aug. 2003.

[9] L. Kamstra and H. J. A. M. Heijmans, "Reversible data embedding into images using wavelet techniques and sorting," *IEEE Trans. Image Process.*, vol. 14, no. 12, pp. 2082–2090, Dec. 2005.

[10] D. M. Thodi and J. J. Rodriguez, "Expansion embedding techniques for reversible watermarking," *IEEE Trans. Image Process.*, vol. 16, no. 3, pp. 721–730, Mar. 2007.

[11] Z. Ni, Y. Q. Shi, N. Ansari, and W. Su, "Reversible data hiding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 16, no. 3, pp. 354–362, Mar. 2006.

[12] S. K. Lee, Y. H. Suh, and Y. S. Ho, "Reversible image authentication based on watermarking," in *Proc. IEEE Int. Conf. Multimedia Expo*, Jul. 2006, pp. 1321–1324.

[13] W. Hong, T. S. Chen, and C. W. Shiu, "Reversible data hiding for high quality images using modification of prediction errors," *J. Syst. Softw.*, vol. 82, no. 11, pp. 1833–1842, Nov. 2009.

[14] D. Coltuc and J. M. Chassery, "Very fast watermarking by reversible contrast mapping," *IEEE Signal Process. Lett.*, vol. 14, no. 4, pp. 255–258, Apr. 2007.

[15] X. Wang, X. Li, B. Yang, and Z. Guo, "Efficient generalized integer transform for reversible watermarking," *IEEE Signal Process. Lett.*, vol. 17, no. 6, pp. 567–570, Jun. 2010.

[16] F. Peng, X. Li, and B. Yang, "Adaptive reversible data hiding scheme based on integer transform," *Signal Process.*, vol. 92, no. 1, pp. 54–62, Jan. 2012.

[17] D. Coltuc, "Low distortion transform for reversible watermarking," *IEEE Trans. Image Process.*, vol. 21, no. 1, pp. 412–417, Jan. 2012.

[18] A. M. Alattar, "Reversible watermark using the difference expansion of a generalized integer transform," *IEEE Trans. Image Process.*, vol. 13, no. 8, pp. 1147–1156, Aug. 2004.

[19] Y. Hu, H. K. Lee, and J. Li, "DE-based reversible data hiding with improved overflow location map," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 19, no. 2, pp. 250–260, Feb. 2009.
[20] X. Li, B. Yang, and T. Zeng, "Efficient reversible watermarking based on adaptive prediction-error expansion and pixel selection," *IEEE Trans. Image Process.*, vol. 20, no. 12, pp. 3524–3533, Dec. 2011.
[21] J. Zhou and O. C. Au, "Determining the capacity parameters in PEE-based reversible image watermarking," *IEEE Signal Process. Lett.*, vol. 19, no. 5, pp. 287–290, May 2012.
[22] H.-T. Wu and J. Huang, "Reversible image watermarking on prediction errors by efficient histogram modification," *Signal Process.*, vol. 92, no. 12, pp. 3000–3009, Dec. 2012.
[23] G. Xuan, Y. Q. Shi, P. Chai, X. Cui, Z. Ni, and X. Tong, "Optimum histogram pair based image lossless data embedding," in *Proc. Int. Workshop Digit. Watermarking Ser. Springer Lect. Notes Comput. Sci.*, 2007, pp. 264–278.
[24] M. Fujiyoshi, S. Sato, H. L. Jin, and H. Kiya, "A location-map free reversible data hiding method using block-based single parameter," in *Proc. IEEE Int. Conf. Image Process.*, vol. 3, Oct. 2007, pp. 257–260.
[25] M. Fallahpour, "Reversible image data hiding based on gradient adjusted prediction," *Inst. Electron. Inf. Commun. Eng. Electron. Exp.*, vol. 5, no. 20, pp. 870–876, Oct. 2008.
[26] K. S. Kim, M. J. Lee, H. Y. Lee, and H. K. Lee, "Reversible data hiding exploiting spatial correlation between sub-sampled images," *Pattern Recognit.*, vol. 42, no. 11, pp. 3083–3096, Nov. 2009.
[27] P. Tsai, Y. C. Hu, and H. L. Yeh, "Reversible image hiding scheme using predictive coding and histogram shifting," *Signal Process.*, vol. 89, no. 6, pp. 1129–1143, Jun. 2009.
[28] W. L. Tai, C. M. Yeh, and C. C. Chang, "Reversible data hiding based on histogram modification of pixel differences," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 19, no. 6, pp. 906–910, Jun. 2009.
[29] L. Luo, Z. Chen, M. Chen, X. Zeng, and Z. Xiong, "Reversible image watermarking using interpolation technique," *IEEE Trans. Inf. Forens. Security*, vol. 5, no. 1, pp. 187–193, Mar. 2010.
[30] W. Hong, T. S. Chen, Y. P. Chang, and C. W. Shiu, "A high capacity reversible data hiding scheme using orthogonal projection and prediction error modification," *Signal Process.*, vol. 90, no. 11, pp. 2911–2922, Nov. 2010.
[31] X. Gao, L. An, Y. Yuan, D. Tao, and X. Li, "Lossless data embedding using generalized statistical quantity histogram," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 21, no. 8, pp. 1061–1070, Aug. 2011.
[32] V. Sachnev, H. J. Kim, J. Nam, S. Suresh, and Y. Q. Shi, "Reversible watermarking algorithm using sorting and prediction," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 19, no. 7, pp. 989–999, Jul. 2009.
[33] S. Weng, Y. Zhao, J. S. Pan, and R. Ni, "Reversible watermarking based on invariability and adjustment on pixel pairs," *IEEE Signal Process. Lett.*, vol. 15, no. 11, pp. 721–724, Nov. 2008.
[34] M. J. Weinberger, G. Seroussi, and G. Sapiro, "The LOCO-I lossless image compression algorithm: Principles and standardization into JPEG-LS," *IEEE Trans. Image Process.*, vol. 9, no. 8, pp. 1309–1324, Aug. 2000.
[35] W. Hong, "Adaptive reversible data hiding method based on error energy control and histogram shifting," *Opt. Commun.*, vol. 285, no. 2, pp. 101–108, 2012.
[36] M. Fujiyoshi, T. Tsuneyoshi, and H. Kiya, "A parameter memorization-free lossless data hiding method with flexible payload size," *Inst. Electron. Inf. Commun. Eng. Electron. Exp.*, vol. 7, no. 23, pp. 1702–1708, Nov. 2010.

Xiaolong Li received the B.S. degree from Peking University, Beijing, China, the M.S. degree from the Ecole Polytechnique, Palaiseau, France, and the Ph.D. degree in mathematics from ENS de Cachan, Cachan, France, in 1999, 2002, and 2006, respectively.

He is currently a researcher with the Institute of Computer Science and Technology, Peking University, where he was a Post-Doctoral Fellow from 2007 to 2009. His current research interests include image processing and information hiding.

Bin Li received the B.E. and Ph.D. degrees from Sun Yat-sen University, Guangzhou, China, in 2004 and 2009, respectively.

He is currently with Shenzhen University, Shenzhen, China, where he joined in 2009. From 2007 to 2008, he was a Visiting Scholar with the New Jersey Institute of Technology, Newark, NJ, USA, where he was involved in research. His current research interests include image processing and information security.

Bin Yang received the B.S. and M.S. degrees in computer science from Peking University, Beijing, China, in 1991 and 1994, respectively.

He is currently a Professor with the Institute of Computer Science and Technology, Peking University. His current research interests include image processing and information hiding.

Tieyong Zeng received the B.S. degree from Peking University, Beijing, China, the M.S. degree from the Ecole Polytechnique, Palaiseau, France, and the Ph.D. degree from the University of Paris XIII, Paris, France, in 2000, 2004, and 2007, respectively.

He is currently an Assistant Professor with Hong Kong Baptist University, Hong Kong. He was a Post-Doctoral Researcher with the Centre de Mathematiques et de Leurs Applications, École Normale Supérieure de Cachan, Cachan, France. His current research interests include image processing, statistical learning, scientific computing, and information hiding.