Difference Expansion Based Reversible Data Hiding Using Two Embedding Directions

Yongjian Hu, Member, IEEE, Heung-Kyu Lee, Kaiying Chen, and Jianwei Li

Abstract—Current difference-expansion (DE) embedding techniques perform one layer embedding in a difference image. They do not turn to the next difference image for another layer embedding unless the current difference image has no expandable differences left. The obvious disadvantage of these techniques is that image quality may have been severely degraded even before the later layer embedding begins because the previous layer embedding has used up all expandable differences, including those with large magnitude. Based on integer Haar wavelet transform, we propose a new DE embedding algorithm, which utilizes the horizontal as well as vertical difference images for data hiding. We introduce a dynamical expandable difference search and selection mechanism. This mechanism gives even chances to small differences in two difference images and effectively avoids the situation that the largest differences in the first difference image are used up while there is almost no chance to embed in small differences of the second difference image. We also present an improved histogram-based difference selection and shifting scheme, which refines our algorithm and makes it resilient to different types of images. Compared with current algorithms, the proposed algorithm often has better embedding capacity versus image quality performance. The advantage of our algorithm is more obvious near the embedding rate of 0.5 bpp.

Index Terms—Data hiding, difference expansion embedding, integer Haar wavelet transform, reversible watermarking.

I. INTRODUCTION

REVERSIBLE data hiding can provide the extracted data as well as the losslessly recovered original image at the receiver end. This property makes it very useful in the areas where image quality is strictly required though the images need to be modified in the process. Each layer embedding progressively decreases the correlation inherent in a neighborhood [14]. Third, the algorithm can not keep its behavior smoothly because each layer embedding has its own embedding capacity limit. The sudden dip in the capacity versus distortion curve for the DE methods around 0.5 bpp is the effect of multiple-layer embedding [14].

In this work, we aim at solving the mentioned problems. Based on integer Haar wavelet transform, we propose an algorithm that selects expandable differences under the same selection threshold in two difference images and embeds the payload in two orthogonal embedding directions. Furthermore, no matter in which difference image, our algorithm can always give priority to the use of small differences. This scheme greatly improves image quality. Our algorithm does not have the original layer embedding capacity limit. The algorithm...
performance is smooth and varies gradually with the change of payloads.

The rest of this paper is organized as follows. Section II briefly reviews the DE-based algorithms in [6] and [14]. Section III describes our reversible data hiding algorithm in detail. Section IV gives the experimental results of our algorithm. We also compare it with other current algorithms. In Section V, we draw the conclusion.

II. REVIEW

This section briefly reviews the principle of DE embedding and the typical algorithms in [6] and [14]. Suppose a pair of image pixel values \((x, y)\), \(0 \leq x, y \leq 255\), their integer average \(l\) and difference \(h\) are computed as

\[
l := \left\lfloor \frac{x + y}{2} \right\rfloor, \quad h := x - y
\]

The inverse transform of (1) is

\[
x = l + \left\lfloor \frac{h + 1}{2} \right\rfloor, \quad y = l - \left\lfloor \frac{h}{2} \right\rfloor.
\]

The reversible transforms (1) and (2) are called integer Haar wavelet transform in [6]. Apparently, \(l\) and \(h\) are the low-frequency and high-frequency signal components, respectively. Tian kept \(l\) unchanged and embedded a binary bit \(i\) into \(h\) by the following DE rule.

\[
h' = 2h + i.
\]

Tian defined expandable differences by (4) and exchangeable differences by (5). The constraint (4) ensures the inverse transform between (1) and (2).

\[
[2h + i] \in [0, \min(2^n - 1 - l), 2l + 1]
\]

\[
[\frac{h}{2} + i] \in [0, \min(2^n - 1 - l), 2l + 1]
\]

where \(n = 8\) for an 8-bit image. Data embedding and extraction only take place in expandable and exchangeable differences. In essence, Tian’s algorithm relies on expandable differences to provide spare space for data hiding and exchangeable differences to guarantee blind data extraction. All difference values in the difference image are classified into three groups: expandable, exchangeable, and nonexchangeable. Four disjoint sets are further defined to describe these three groups. For a payload, a number of expandable differences are selected from the expandable sets. The corresponding location map of selected differences is then obtained. The compressed location map is saved along with the payload for blind data extraction. Due to multiple-layer embedding and inefficient compression of location map, Tian’s algorithm often has the problems of layer embedding capacity limit and low image quality.

During implementation, Tian’s sets-based difference selection scheme is indirect and complicated. In particular, it is inconvenient for capacity control. To facilitate the selection process and capacity control, Thodi et al. [14] introduced a more intuitive scheme. They used a difference histogram to describe the statistical distribution of expandable differences. The histogram is divided into two types of regions: the inner region (to be embedded) that consists of the bins of selected expandable differences and the outer regions (to be shifted) that contain the bins of unselected differences. By directly shifting the selection threshold along the difference axis, they could change the size of inner and outer regions, and thus, assign a proper number of expandable differences for the embedding of a payload. To ensure lossless recovery of the original image, these two regions are kept not intersected with each other, even after embedding. Thodi et al. used a histogram shifting operation to avoid region-overlapping. In order to avoid overflow/underflow of embedded/expanded pixel values, this histogram only consists of expandable differences. All possible overflow differences are expelled from the histogram in advance and their locations are recorded by an overflow location map. From now on, we use overflow to denote either overflow or underflow for the sake of simplicity. Such a map is saved along with the payload for the reversibility of the embedding process. The histogram-based difference selection and shifting scheme and the good compressibility of the overflow location map enable Thodi et al.’s method to be an efficient algorithm. There are five versions of the algorithm in [14], where DE-HS-OM (DE with histogram shifting and overflow map) and DE-HS-FB (DE with histogram shifting and flag bits) methods are based on integer Haar wavelet transform domain, and the rest three are based on image prediction errors. For large payloads, the prediction-error (PE)-based methods often have better performance; for small payloads, however, the integer Haar wavelet transform domain (DE-based) ones are more suitable. In general, these two types of methods have different advantages. More detailed information can be found in [14]. Usually, DE-HS-OM and DE-HS-FB methods have close performance. Like all other integer Haar wavelet transform domain methods, both of them suffer from the problems mentioned in Section I.

Basically, every method using this type of difference images faces the same problem. The reason is multiple-layer embedding. For example, for a payload, the integer Haar wavelet transform domain method first searches for and uses small expandable differences in one difference image. With the increase of payload size, it selects more and more large differences to obtain enough embeddable locations. Note that small or large differences refer to differences with small or large magnitude in this paper. When the payload size is close to 0.5 bpp, the method almost uses up all expandable differences in the current difference image, including the largest possible differences. If the payload size continues to increase, this method can not satisfy the capacity requirement in the single layer embedding but depends on multiple-layer embedding. However, only after an exhausted search for expandable differences can not satisfy the payload, the method goes to the next layer embedding, which often takes place in the difference image obtained in orthogonal transform direction. Obviously, before beginning the next layer embedding, the image quality has been greatly degraded due to the use of large differences. One way to solve this problem is to assign the payload to the two difference images obtained in orthogonal transform directions, so that the differences under the same level of magnitude in the two difference images can be used. It could avoid the situation that there is few chance to use small differences in the second difference image while all large expandable
differences are used up in the first difference image. This is the initial motivation of this work.

III. ALGORITHM

For simplicity of description, we use $I$ to represent the image and use $HH$, $HL$, $VH$, and $VL$ as its subscripts to denote horizontal difference/high-frequency image, horizontal average/low-frequency image, vertical difference/high-frequency image, and vertical average/low-frequency image, respectively.

We further define the following notations.

$\mathcal{P}$ Payload.

$C$ Embedding capacity.

$A$ Auxiliary data package.

$Q$ Header file.

$M$ 2-D binary overflow location map.

$\mathcal{M}$ Bitstream of compressed 2-D binary overflow location map.

$T$ Selection threshold.

$Z$ Zero point.

$B$ Embedded bitstream.

$\eta(\cdot)$ Function used for calculating the length of a bitstream.

We use a single capital letter $H$ or $V$ as the subscript to denote the above symbols for the horizontal or vertical difference image.

The main phases of the proposed algorithm are exhibited in Fig. 1. We first perform integer Haar wavelet transform in row direction to decompose the image $I$ into $I_{HH}$ and $I_{HL}$. Then, the improved histogram-based difference selection and shifting scheme is carried out in $I_{HH}$ to embed the horizontal payload $\mathcal{P}_H$. Similarly, in column direction, we have $I_{VH}$ and $I_{VL}$. The improved histogram-based difference selection and shifting scheme is performed in $I_{VH}$ to embed the vertical payload $\mathcal{P}_V$.

The division of the payload $\mathcal{P}$ into $\mathcal{P}_H$ and $\mathcal{P}_V$ depends on the dynamical expandable difference search and selection mechanism, which is the focus of this work. It is worth mentioning that Fig. 1 only illustrates the situation that the embedding process begins from the horizontal difference image. If the embedding process begins from the vertical difference image, a similar diagram can be obtained. Below we first describe our histogram-based difference selection and shifting scheme, and then, describe the dynamical expandable difference search and selection mechanism. Afterwards, we discuss data hiding. In the last
subsection, we show how to perform data extraction and the recovery of the original image.

A. Scheme for Selecting Expandable Differences

Our histogram-based difference selection and shifting scheme is an improved version of the scheme in [14]. We briefly review that scheme before introducing our improved one.

1) Histogram-Based Difference Selection and Shifting Scheme: Thodi et al. [14] selected expandable differences/locations based on the difference histogram. Since the difference values of the difference image approximately follow the Laplace distribution, Thodi et al. designed an outward histogram shifting scheme to give priority to the bins with small differences in the process of difference selection. We use Fig. 2 to make a concise explanation.

In Fig. 2, the regions occupied by the bins with selected and unselected expandable differences are defined as the inner region and the outer regions, respectively. Assume the selection threshold $T \geq 0$. The selected bins in the inner region have difference values in the range $[-T-1, T]$. After DE embedding, the use of the embedding rule (3) makes the inner region expanded to become $[-2T-2, 2T+1]$. Apparently, the expanded parts $[-2T-2, -T-2] \cup [T+1, 2T+1]$ would overlap the original outer regions. The original difference values in the overlapped regions would be permanently lost if no action is taken to avoid such an overlap. Therefore, before embedding, Thodi et al. shifted the left outer region and the right one outwards along the difference axis by at least $T+1$, respectively.

2) Improved Histogram-Based Difference Selection and Shifting Scheme: Although the above scheme is simple and efficient, it still has some weakness. We improve it in two aspects. We first discuss how to use the histogram zero points, and then, we introduce a payload-dependent overflow location map.

Basically, histogram shifting implicitly corresponds to altering difference values. It brings about numerical distortion to the image. Therefore, we should decrease the histogram shifting distance as much as possible. One way to reduce the shifting distance is to use zero points. We use Figs. 2 and 3 to explain this idea. Suppose $T = 0$ in Fig. 2. The inner region is $[-1,0]$ and the corresponding outer regions are $[-28, -2]$ and $[1,33]$. After DE embedding, the inner region is expanded and becomes $[-2,1]$. According to the shifting strategy in [14], prior to embedding, the outer regions should be shifted outwards to become $[-29, -3]$ and $[2,34]$, respectively. However, the difference histogram derived from a natural image may be like what shown in Fig. 3. So the size of shifted intervals/segments as well as the shifting distance can be reduced by using zero points. A zero point is usually defined as $h(\neq 0)$ with $f(h) = 0$. In this work, however, we give one more constraint. A zero point should not be located in the inner region but in $[-255, -T-2]$ or $[T+1, 255]$. Theoretically, a difference value coming from an 8-bit image falls into the range $[-255, 255]$ though the real difference histogram is much narrower. However, only zero points within outer regions of the histogram contribute to our improved histogram shifting scheme. A useful zero point should not be greater than 126 or less than $-127$ because the largest inner region occurs at $T = 126$ under the embedding rule (3).

It can be observed in Fig. 3 that there are four zero points, i.e., $-23, -19, 23,$ and $31$, within the tails of the histogram and two zero points, i.e., $-29$ and $34$, as the histogram ends. These zero points initially divide the histogram into several segments. Precisely, the right outer region has three segments: $[1,22]$, $[24,30]$, and $[32,33]$, and the left outer region also has three segments: $[-28, -24]$, $[-22, -20]$, and $[-18, -2]$. Apparently, for some $T$, we do not need to shift the whole outer regions. We use an example to explain the situation. When $T = 0$, only $[1,22]$ is shifted to $[2,23]$, and $[-18, -2]$ shifted to $[-19, -3]$. It is unnecessary to shift the other four segments. When $T = 1$, the right segments $[1,22]$ and $[24,30]$ are shifted to $[3,24]$ and $[25,31]$, respectively. At the same time, the left segments $[-18, -2]$ and $[-22, -20]$ are shifted to $[-20, -4]$ and $[-23, -21]$, respectively. Still, two segments are left intact. When $T = 2$, $[1,22]$, $[24,30]$, $[32,33]$ become $[4,25]$, $[26,32]$, and $[33,34]$, respectively; and meanwhile, $[-18, -2]$, $[-22, -20]$, and $[-28, -24]$ become $[-21, -5]$, $[-24, -22]$, and $[-29, -25]$, respectively. Obviously, only after $T \geq 2$, that is, all of the zero points within the histogram are used up, the whole outer region would be shifted. Even so, four of the original segments divided by these zero points do not shift as much as the original cases (without using zero points) do. This improvement effectively reduces the alteration to the difference values in the outer regions.
We further investigate the use of zero points. Generally, the effect of using zero points depends on two factors: image characteristics and payload size. If an image has many zero points within its histogram, the outer regions are often divided into separate segments. Using zero points would greatly reduce the alteration to the histogram during histogram shifting. On the other hand, if the payload size is small, the inner region is small and the corresponding outer regions are large. There may be more zero points within the outer regions. Using zero points has a stronger effect. However, as the payload size increases, the inner region becomes larger. There are less zero points left in the outer regions. The effect of using zero points becomes weaker. We still use the above example to explain this situation. Suppose the payload is large and the inner region has to be $[-25,24]$ for satisfying this payload. The corresponding outer regions are $[-28,-26], [25,30]$ and $[32,33]$. There is only one zero point, i.e., 31, left in the outer regions. Apparently, the effect of using zero points becomes much weaker. Since $-23,-19,23$ are now located in the inner region, they are no longer zero points. We treat these points as common locations in the inner region and determine whether they are expandable or not using (4).

Now we discuss how to improve the overflow location map. As mentioned before, in order to ensure no overflow occurring during embedding and shifting, the difference histogram in [14] only consists of expandable differences. To match the histogram, a payload-independent overflow location map is constructed before using the histogram, which records all of possible overflow locations in the difference image. Although that payload-independent overflow location map has efficient compression for common images, it often can not maintain good compression for other types of images, e.g., images with rich textures. In this work, we present an overflow location map that can suit different types of images. This overflow location map is payload-dependent and varies with the change of inner and outer regions. It can achieve much higher compression ratio. Basically, our algorithm adopts a quite different difference histogram. It consists of all differences of the difference image, including expandable, exchangeable and unexchangeable differences, rather than only expandable differences in Thodi et al.’s histogram. The proposed overflow location map is used to match our new difference histogram. To correctly construct the overflow location map, we give the constraints on embedding and shifting operations below.

The construction of a 2-D binary overflow location map $M$ begins from initialization. $M$ is initialized with “0”. If a difference value causes overflow, the corresponding location is changed to “1”. $M$ describes the state of a difference in the inner region or outer regions.

- For a difference $h$ in the inner region, we judge it using (4). If $h$ does not satisfy (4), its location in $M$ is indicated with “1”; otherwise, we do not alter $M$.
- For a difference $h$ in the outer regions, we first calculate the shifted difference value using $h' = h + \xi$, where $\xi(|\xi| \leq T + 1)$ is the shifting distance corresponding to a segment divided by zero points. We then determine whether $h'$ causes overflow or underflow by (5). Note that we substitute $h'$ for $h$ when using (5). If (5) is not satisfied, its location in $M$ is indicated with “1”; otherwise, we do not alter $M$. A difference value that satisfies (5) is defined as a shiftable difference in this work. The reason of using (5) is that the shifted difference values may be performed with LSB replacement during data encoding. We will explain the LSB replacement operation in Section III-C.

So far, we have described the improved histogram-based difference selection and shifting scheme. To sum up, the histogram-based difference selection process is the process determining the selection threshold $T'$ for a given payload. The search process begins with $T' = 0$. We increase $T'$ by 1 each time until the right $T'$ is found. The expansion of the inner region occurs after the outward shifting of the outer regions. The zero points are used for decreasing the shifting distance of the outer regions. Since the matrix of our payload-dependent overflow location map is often sparser than that in [14], it has higher compression efficiency.

B. Dynamical Search for Expandable Differences in Horizontal and Vertical Difference Images

We carry out our difference selection scheme to search for expandable locations in both horizontal and vertical difference images. To illustrate our two-direction search mechanism, we further give Fig. 4 focusing on the parts of interest in Fig. 1. We use Figs. 1 and 4 to explain each step of our dynamical search mechanism.

- Step 1: For an image $I$, we perform integer Haar wavelet transform in row direction and obtain the horizontal difference image $I_{H\ H}$.
- Step 2: We then calculate the histogram of $I_{H\ H}$.
- Step 3: We perform the improved histogram-based difference selection and shifting scheme. Suppose the payload as $P$. The initial difference selection threshold is 0, but it increases by 1 in each new search round. Suppose the current selection threshold and the horizontal embedding capacity are $T_H$ and $C_H$, respectively. In essence, $C_H$ is the number of selected expandable differences under $T_H$. So the required capacity $C_{H'} = \eta(Q_H) + \eta(M_H) + \eta(P) - C_H$. We will explain why we embed $Q_H$ and $M_H$ in the next subsection. We first judge whether $C_{H'} > \eta(Q_H) + \eta(M_H)$. This is the basic requirement. The set of selected expandable differences supplies spare space for the payload only after it has enough locations for the auxiliary data that consists of $Q_H$ and $M_H$. If this condition is not satisfied, we directly increase $T_H$ by 1 and repeat Step 3 again. Otherwise, we go to the next judgement. If $C_{H'} > 0$, the horizontal capacity under $T_H$ is not large enough for $P$. Only partial payload, i.e., $P_H = C_{H'} - \eta(Q_H) - \eta(M_H)$, will be embedded in this direction. We need the additional embedding capacity and have to search in the vertical difference image. If $C_{H'} \leq 0$, the horizontal capacity under $T_H$ is large enough for $P$. We do not need any more locations. In either of these two cases, we obtain the horizontally embedded difference image $I_{HH}$ by performing data hiding, which we will further discuss in Section III-C. However, for the former case, $I_{HH}$ is the preparation for Steps 4–8, whereas for the latter case, $I_{HH}$ is the preparation for the end of the search process. The latter case happens when a payload is small. We use Fig. 5 to further explain the operations in Step 3.
- Step 4: We obtain the horizontally embedded image $I'$ by synthesizing $I_{HH}$ and the average image $I_{HL}$ in Fig. 1.
- Step 5: We perform integer Haar wavelet transform on $I'$ in column direction to obtain the vertical difference image $I_{VH}^*$.
- Step 6: We then calculate the difference histogram of $I_{VH}^*$. We use another inner search loop to test whether the vertical embedding capacity under $T_H$, called $C_V$, satisfies $C_V \leq C_H$. The inner search loop begins with $T_V = 0$. In each pass, $T_V$ increases by 1. The search process continues until $C_V > 0$. Note that, no matter how we increase $T_V$, the vertical selection threshold should never be greater than the horizontal threshold, i.e., $T_V \leq T_H$. If the final $C_V$ is greater than 0, the search round under current $T_H$ fails. We have to increase $T_H$ by 1 and repeat the steps 3–7 again. Once $C_V \leq 0$, it implies that the vertical difference image can provide enough embedding capacity for the rest of the payload. We then perform data hiding and obtain the vertically embedded difference image $I_{VH}'$. We use Fig. 6 to further explain the operations in Step 7.
- Step 8: By performing inverse integer Haar wavelet transform, we synthesize the vertically embedded difference image $I_{VH}'$ and the average image $I_{VL}$ in Fig. 1 to obtain the output image $I''$.

**C. Data Hiding**

As seen in Fig. 1, data hiding is carried out during the dynamical expandable difference search process. In practice, prior to each search in the vertical difference image, we have to perform data hiding in the horizontal difference image $I_{HH}$ to get $I'$, from which we derive $I_{VH}$. Below we discuss the data hiding operations. We first show how to construct the header file.

1) **Construction of Header File:** For blind data extraction and the lossless recovery of the original image, we use the header file $Q$ to record all of embedding parameters for the reversibility of the embedding process. Since data hiding takes place in both horizontal and vertical directions, we need two header files. We use the horizontal header file $Q_H$ to explain the construction process. The vertical header file $Q_V$ is constructed in a similar way.

$$Q_H = \mathcal{F}(16 \text{ bits}) + T_H(8 \text{ bits}) + Z_H((T_H + 1) \times 8 \text{ bits}) + Z_{HH}((T_H + 1) \times 8 \text{ bits}) + \eta(M_H)(24 \text{ bits}) + \eta(P_H)(24 \text{ bits})$$

where $\mathcal{F}$ is a hexadecimal number that denotes the layer embedding information. $F = 0000$ and $F = FFFF$ refer to horizontal embedding and vertical embedding in a single layer embedding.
mode, respectively. $\mathcal{F} = 00\text{FF}$ and $\mathcal{F} = \text{FF}00$ refer to horizontal embedding and vertical embedding in a two-layer embedding mode, respectively. $Z_{\text{HP}}$ and $Z_{\text{HN}}$ represent the arrays of zero points in the positive difference axis and the negative difference axis, respectively. In fact, for the selection threshold $T_H$, the shifting of right outer region passes through $T_H + 1$ positive zero points, and the shifting of left outer region passes through $T_H + 1$ negative zero points. So we need these zero points in the process of data extraction. $\eta_\mathcal{M}(M_H)$ and $\eta_\mathcal{P}(P_H)$ refer to the length information of the compressed overflow location map and the embedding payload, respectively.

2) Embedding for Blind Data Extraction: The data hiding process is identical in both horizontal and vertical difference images, so we give a description without indicating the specific embedding direction. As stated in the last subsection, the auxiliary data $\mathcal{A} = Q \cup M$, where $\cup$ denotes the concatenation operation and $M$ is the compressed $M$ obtained by using JBIG-kit in [15]. $\mathcal{A}$ is a necessity for blind data extraction. Consequently, the bitstream to be embedded is constructed as $\mathcal{B} = \mathcal{A} \cup \mathcal{P} = Q \cup M \cup \mathcal{P}$. The number of selected expandable differences should be equal to $\eta_\mathcal{B}(B)$, so that the bitstream bits are embedded into the selected expandable difference values bit by bit using (3). However, at the decoder, we could not extract the hidden data if we had taken such a manner of embedding. In fact, without the overflow location map and the embedding parameters, we could not know the locations of the selected differences. It means that even we have determined the bitstream as well as the expandable locations for hiding, we have to make such a hiding process easy for blind data extraction.

In [6], Tian implemented blind data extraction using the LSBs of changeable differences. Before embedding, the original LSBs of both unselected expandable differences and exchangeable differences are saved into a set. Then, the combined bitstream of the compressed location map, the LSBs set and the payload are embedded in sequence into the LSBs of expandable and exchangeable differences. During blind data extraction, according to the properties of exchangeable differences and expandable differences, the combined bitstream can be extracted from the LSBs of the exchangeable differences, so that the original image can be recovered losslessly. The reader is referred to [6] for more detailed information.

We borrow the idea from [6]. Data hiding begins from the upper left corner of the difference image in a manner of raster scan. We save original LSBs of both shiftable differences and exchangeable differences in a set $I_{\text{LS}}$. Unlike [6], we only save the LSBs of the first $\eta_\mathcal{A}$ difference image pixels. In other words, we change the amount of difference image pixels as less as possible in order to keep embedding distortion to the lowest degree. We append the bitstream $I_{\text{LS}}$ to $\mathcal{B}$. Apparently, $\eta_\mathcal{B}(\mathcal{B})$ bits are embedded by DE, and $\eta_\mathcal{I}_{\text{LS}}$ bits are embedded by LSB replacement. We implement data hiding in the following steps.

- Step 1: We scan the difference image in a raster scan manner and perform data hiding from the upper left corner.
- Step 2: If the difference image pixel $h_i$ is in the inner region and expandable, we hide one bit of the to-be-embedded bitstream into $h_i$ using (3).
- Step 3: If $h_i$ is in the outer regions and shiftable, we shift it based on its segment using our shifting scheme to obtain $h'_i$. We then save the LSB of $h'_i$ into $I_{\text{LS}}$ in order. After that, we hide one bit of the to-be-embedded bitstream into $h'_i$ using LSB replacement.
- Step 4: If $h_i$ is neither expandable nor shiftable, but it is exchangeable, we first save the LSB of $h_i$ into $I_{\text{LS}}$. After that, we hide one bit into $h_i$ using LSB replacement.
- Step 5: If $h_i$ may lead to overflow, we skip it and go to the next pixel.
- Step 6: We repeat Steps 2-5 until we save $\eta_\mathcal{A}$ LSBs into $I_{\text{LS}}$. After that, we only perform Step 2 and hide bits into expandable differences. As soon as all bits of bitstream $\mathcal{B}$ are hidden, we hide the bits of $I_{\text{LS}}$ immediately. The hiding operation proceeds until we hide all bits of $\mathcal{B}$ and $I_{\text{LS}}$.

**TABLE I**

<table>
<thead>
<tr>
<th>Difference classification</th>
<th>Before data hiding</th>
<th>After data hiding</th>
<th>Value in $M$</th>
<th>Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expandable</td>
<td>Inner region</td>
<td>$[-2T - 2, 2T + 1]$</td>
<td>0</td>
<td>$h = [\eta/2]$</td>
</tr>
<tr>
<td>Shiftable</td>
<td>Outer regions</td>
<td>$[-Z_{\text{max}} - 2T - 3, 2T + 2, Z_{\text{pmax}}]$</td>
<td>0</td>
<td>$2[\eta/2] + i$, then shift back</td>
</tr>
<tr>
<td>Exchangeable</td>
<td>Inner/outer regions</td>
<td>$[-Z_{\text{max}}, Z_{\text{pmax}}]$</td>
<td>1</td>
<td>$h = 2[\eta/2] + i$</td>
</tr>
<tr>
<td>Unexchangeable</td>
<td>Inner/outer regions</td>
<td>$[-Z_{\text{max}}, Z_{\text{pmax}}]$</td>
<td>1</td>
<td>$h$</td>
</tr>
</tbody>
</table>

**Fig. 7** Test images of $512 \times 512 \times 8$ bits. From left to right: Lena, F-16, Baboon, and Stream and bridge.
D. Data Extraction and Lossless Recovery

Data extraction and the lossless recovery of the original image are implemented in two separate processes. The latter process needs the result of the former. We first discuss data extraction. According to [6], a changeable location remains changeable even after its LSB is replaced, whereas an expandable location may not be expandable after DE embedding, but it remains changeable.

- Step 1: For a test difference image, we begin data extraction from the upper left corner of the image in a manner of raster scan, just as the data hiding process does.
- Step 2: We extract the LSBs of difference image pixels to resume the embedding parameters. According to the structure of \( Q \), the LSBs of the first 16 difference image pixels correspond to \( F \). The following 8 LSBs correspond to the selection threshold \( T \). The next two \( (T+1) \times 8 \) LSBs sequences correspond to the arrays of positive and negative zero points in turn. After that, we get the length information of the overflow location map \( \eta(M) \) and the length information of the embedded payload from the following two 24 LSBs sequences, respectively. So we have obtained the complete information bits of the header file \( Q \).
- Step 3: According to the order of data hiding, the following \( \eta(M) \) LSBs constitute the bitstream of compressed overflow location map \( M \). So we decompress \( M \) using inverse JBIG compression to get back the original overflow location map \( M \). Thereafter, with the obtained \( Q \) and \( M \), we can extract the rest of hidden bits from the difference image.
- Step 4: Due to \( \eta(L_{\text{rev}}) = \eta(A) \), we are able to recalculate the position where we stop performing LSB replacement in the process of data hiding. We keep the operation of extracting LSBs from the difference image pixels until we reach that position. After that position, we only extract the LSBs from expanded differences. After the first \( \eta(P) \) LSBs, the extracted bits belong to two parts: the first \( \eta(P) \) bits are the payload bits hidden in this difference image; the rest bits are the elements of \( L_{\text{rev}} \).

After data extraction is finished, we can begin the image recovery process. From the upper left corner, we scan the difference image again and resume each difference image pixel. The recovery operations depend on \( Q \), \( M \), and \( L_{\text{rev}} \). We list the rules for recovery in the Table I. The recovery process proceeds until all the difference image pixels are scanned.
IV. EXPERIMENT AND DISCUSSION

We have tested the proposed algorithm on different types of images. In this work, we give the experimental results of four test images that are shown in Fig. 7. We focus on comparing our algorithm with Thodi et al.’s DE-HS-OM algorithm in [14], which is one of the latest DE-based algorithms in literature. We also simply compare our algorithm with Tian’s original DE algorithm in [6], and Kamstra et al.’s Extended DE algorithm in [9]. Extended DE algorithm is an improved version of Tian’s DE algorithm. Since Extended DE algorithm uses the characteristics of the low-pass image to predict expandable locations, it often has good performance under small payloads. More detailed information can be found in [9]. For simplicity, we call our algorithm as DE-TED in Fig. 8 and Table II, where Fig. 8 gives the embedding capacity versus image quality curves of four algorithms and Table II gives the detailed results of our algorithm. For better comparison, Table II also gives the detailed results of DE-HS-OM algorithm. The payloads 14 000 bits, 27 000 bits, ... 210 000 bits, 237 000 bits approximately correspond to 0.05 bpp, 0.1 bpp, 0.2 bpp, ... 0.8 bpp, 0.9 bpp, in sequence.

For common images like Lena, when the embedding rate is less than 0.15 bpp, our algorithm and DE-HS-OM algorithm produce close PSNR values. The main reason is that our algorithm only uses one embedding direction under such small payloads, so does DE-HS-OM algorithm, as shown in Table II. From 0.2 bpp to 0.4 bpp, however, there are some difference between our algorithm and DE-HS-OM algorithm in image quality, and this difference tends to be larger with the increase of payload size. Table II exhibits that our algorithm takes two embedding directions at these embedding rates, but DE-HS-OM algorithm still embeds in one layer; moreover, under the same payload, our algorithm uses smaller selection thresholds than DE-HS-OM algorithm does. It implies that our algorithm uses smaller differences for embedding. Therefore, our algorithm produces higher PSNR values. Since the difference between our selection thresholds and DE-HS-OM algorithm’s counterparts is not large under these payloads, the difference between our algorithm’s PSNR values and DE-HS-OM algorithm’s counterparts is not large. The biggest difference between our algorithm and DE-HS-OM algorithm in PSNR values occurs at 0.5 bpp, as shown in Fig. 8(a). The PSNR value produced by the former is about 5 dB higher than that produced by the latter. This phenomenon can be easily explained by Table II. At 0.5 bpp, our algorithm has $T_v = 3$ for vertical embedding direction and $T_H = 2$ for horizontal embedding direction, whereas DE-HS-OM algorithm has $T = 126$ for the first layer embedding and $T = 0$ for the second layer embedding. The use of large differences in its first layer embedding certainly damages the performance of DE-HS-OM algorithm. From 0.5 bpp to 0.9 bpp, our algorithm still performs better than DE-HS-OM algorithm. But with the increase of payload size, our algorithm uses larger selection thresholds, and thus, our advantage over DE-HS-OM algorithm becomes weaker. When the payload size approaches the maximum capacity, the difference between our algorithm and DE-HS-OM algorithm is very small. With regard to DE and Extended DE algorithms, the latter performs better than the former when the embedding rate is less than 0.4 bpp, but the former performs a little better than the latter from 0.5 bpp. Our algorithm outperforms both of them, especially, around 0.5 bpp. It is worth mentioning that the good performance of our algorithm for Lena is mainly due to the dynamical expandable difference search and selection mechanism. Since Lena has a sharp and narrow difference histogram with few zero points in its tails, the use of zero points and the payload-dependent overflow location map has a weak contribution to the algorithm performance.

For flat images like $F = 16$, the situation is similar to that for Lena, but the advantage of our algorithm is more obvious. This is because a flat image is more sensitive to the alteration of image values. It can be seen in Fig. 8(b) that the embedding capacity versus image quality curves of DE-HS-OM, DE, and Extended DE algorithms have the sharpest dip around 0.5 bpp among the four test images. Our embedding capacity versus image quality curve is very smooth. At 0.5 bpp, the PSNR value of our algorithm is about 11 dB higher than any of the other three. Our algorithm respectively uses $T_v = 2$ and $T_H = 2$ for vertical and horizontal embedding directions, and produces a PSNR value of 43 dB; meanwhile, DE-HS-OM algorithm respectively uses 126 and 0 for its first- and second-layer embedding, and produces a PSNR value of 32 dB. DE and Extended DE algorithms also face layer embedding capacity limitation, so they produce similar PSNR values as DE-HS-OM algorithm does. Note that, although the smoothness of the low-pass image of $F = 16$ enables Extended DE algorithm to have a better performance than our algorithm under small payloads, its performance worsens quickly as the embedding rate is greater than 0.3 bpp.

For texture images like Baboon, our algorithm performs better than either DE-HS-OM or DE algorithm at all embedding rates. DE algorithm has the worst embedding capacity versus image quality performance among the four test images. The biggest difference between our algorithm and these two algorithms in PSNR values still occurs at 0.5 bpp. The good performance of our algorithm mainly comes from two sources: the dynamical expandable difference search and selection mechanism and the high compressibility of our payload-dependent overflow location map. According to Table II, from 0.05 bpp to 0.8 bpp, our algorithm gradually increases the selection thresholds, for example, $T_v = 0, 1, 3, 5, 8, 11, 17, 24, 37$; meanwhile, DE-HS-OM algorithm increases the selection thresholds very quickly and uses the selection thresholds 2, 4, 9, 19, 41, 126, 126, 126 in sequence for its first layer embedding. It is clear that DE-HS-OM algorithm uses much larger differences than our algorithm. The use of small differences certainly benefits our image quality. In addition, our payload-dependent overflow location map also shows its advantages for Baboon whose difference histogram is flat and wide. It can be seen that the size of our compressed location map is small and varies with payloads; moreover, the increase in size is small even when the payload becomes very large. Specifically, when the payload is 14 000 bits, the size of our compressed overflow location map
### TABLE II

**DETAILED RESULTS.** We compare our algorithm with Thodi et al.’s DE-HS-OM algorithm. \( P \), \( T \) and \( M \) refer to the payload, the selection threshold and the compressed overflow location map, respectively. The unit of \( P \) and \( M \) is bits. \( V \) and \( H \) refer to the vertical and horizontal embedding directions, respectively. "\(-\)" refers to unavailable. The vertical direction is used as the first embedding direction.

<table>
<thead>
<tr>
<th>Payload</th>
<th>14000</th>
<th>27000</th>
<th>53500</th>
<th>80000</th>
<th>105000</th>
<th>132000</th>
<th>159000</th>
<th>185000</th>
<th>210000</th>
<th>257000</th>
</tr>
</thead>
<tbody>
<tr>
<td>( P )</td>
<td>( V )</td>
<td>14000</td>
<td>27000</td>
<td>53500</td>
<td>80000</td>
<td>105000</td>
<td>132000</td>
<td>159000</td>
<td>185000</td>
<td>210000</td>
</tr>
<tr>
<td>( H )</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( T )</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( M )</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Payload</th>
<th>14000</th>
<th>27000</th>
<th>53500</th>
<th>80000</th>
<th>105000</th>
<th>132000</th>
<th>159000</th>
<th>185000</th>
<th>210000</th>
<th>257000</th>
</tr>
</thead>
<tbody>
<tr>
<td>( P )</td>
<td>( V )</td>
<td>14000</td>
<td>27000</td>
<td>53500</td>
<td>80000</td>
<td>105000</td>
<td>132000</td>
<td>159000</td>
<td>185000</td>
<td>210000</td>
</tr>
<tr>
<td>( H )</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( T )</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( M )</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Payload</th>
<th>14000</th>
<th>27000</th>
<th>53500</th>
<th>80000</th>
<th>105000</th>
<th>132000</th>
<th>159000</th>
<th>185000</th>
<th>210000</th>
<th>257000</th>
</tr>
</thead>
<tbody>
<tr>
<td>( P )</td>
<td>( V )</td>
<td>14000</td>
<td>27000</td>
<td>53500</td>
<td>80000</td>
<td>105000</td>
<td>132000</td>
<td>159000</td>
<td>185000</td>
<td>210000</td>
</tr>
<tr>
<td>( H )</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( T )</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( M )</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Payload</th>
<th>14000</th>
<th>27000</th>
<th>53500</th>
<th>80000</th>
<th>105000</th>
<th>132000</th>
<th>159000</th>
<th>185000</th>
<th>210000</th>
<th>257000</th>
</tr>
</thead>
<tbody>
<tr>
<td>( P )</td>
<td>( V )</td>
<td>14000</td>
<td>27000</td>
<td>53500</td>
<td>80000</td>
<td>105000</td>
<td>132000</td>
<td>159000</td>
<td>185000</td>
<td>210000</td>
</tr>
<tr>
<td>( H )</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( T )</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( M )</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

In the vertical difference image, a location map of 6960 bits is an important factor that affects image quality. Even when the payload is 210000 bits, saving a compressed overflow location map of 25264 bits is a burden. Basically, such a low efficient compression of the second overflow location map is due to the improper embedding in the first difference image. On the other hand, our payload-dependent overflow location map is compression-efficient. It can save a lot of spare space for this type of images. As for Extended DE
algorithm, its performance is better than either DE-HS-OM or DE algorithm’s performance, but still below our algorithm’s performance from 0.05 bpp.

For images with some textures and uneven brightness like Stream and bridge, the advantage of our algorithm is more clear. Fig. 8(d) shows that, when the embedding rate is less than 0.1 bpp, the PSNR value of our algorithm is at least 10 dB higher than that of DE-HS-OM algorithm or Extended DE algorithm, and 20 dB higher than that of DE algorithm. In addition, between 0.1 bpp–0.5 bpp, the PSNR value of our algorithm is at least 5 dB higher than any of the other three. We investigate our algorithm performance from three aspects. In Table II, under the same payload, our algorithm uses much smaller selection thresholds than DE-HS-OM algorithm does; besides, the size of our compressed overflow location map is much smaller than its counterpart of DE-HS-OM algorithm. These two advantages demonstrate that both the dynamical expandable difference search and selection mechanism and our payload-dependent overflow location map contribute to our algorithm performance. However, another major contribution comes from using zero points. Unlike Lena or F—16 that has a sharp and narrow difference histogram, or Baboon that has a flat but continuous difference histogram, Stream and bridge has a difference histogram that is not sharp but ragged and with many zero points close to the origin. Specifically, the closest zero points are 1 on the positive difference axis and –1 on the negative difference axis. Those zero points divide the outer regions into many small segments, which facilitate the use of zero points. For a small payload, our algorithm only shifts a small portion of differences, so that the distortion is very small. That is why our algorithm performs even better than Extended DE algorithm. Usually, Extended DE algorithm has good performance at small embedding rates, as seen in Fig. 8(a)–(c). The effect of using zero points becomes weak as the payload increases.

From the results of these four test images, we conclude that our algorithm not only yields higher image quality and overcomes the abrupt change in the algorithm performance near the layer embedding rate limit, but also has better resilience to different types of images. Table II clearly indicates that our algorithm uses the dynamical expandable difference search and selection mechanism to keep $T_V$ and $T_H$ not far from each other, and the magnitude difference between them is at most 2. In contrast, the DE-HS-OM algorithm goes to the second difference image unless the first difference image can not bear any more payload. Thus, the largest magnitude difference between its two selection thresholds is 126. Table II also shows that our algorithm does not evenly assigns the payload to the two embedding directions, but make efforts on evenly selecting expandable differences under the same threshold in two embedding directions. We have to point out that the results in Fig. 8 and Table II are obtained by using the vertical direction as the first embedding direction. Although there is no much difference in experimental results for our algorithm between using the vertical direction as the first embedding direction and using the horizontal direction as the first embedding direction, the difference does exist for the other three algorithms. In other words, quite different results may appear in Fig. 8 and Table II for these three algorithms if using the horizontal direction as the first embedding direction. We briefly discuss the problem using Table III. To obtain PSNR1, we use the horizontal direction as the first embedding direction. To obtain PSNR2, we use the vertical direction as the first embedding direction. The difference between PSNR1 and PSNR2 is clearly revealed by Table III. Under the same embedding rate, the largest PSNR difference is about 2 dB. This phenomenon can be theoretically explained. In our algorithm, we always select differences under the same threshold in two difference images, so the embedding in the first direction would not severely affect the correlation among image pixels. However, the other three algorithms use up all possible expandable differences in the first difference image before using expandable differences in the second difference image. Since statistical correlation among pixels in the vertical difference image is often different from that in the horizontal difference image, the choice of first embedding direction often leads to different results. This disadvantage is certainly not good for real-world applications.

Prediction-error-based embedding is also a popular reversible data hiding technique in literature. Here we grossly compare our algorithm with P2 (PE expansion with histogram shifting and overflow map) method, which is the best PE version of Thodi et al.’s algorithm in [14]. We directly use the data provided by Thodi et al. For Lena, from 0.1 bpp to 0.9 bpp, P2 approximately produces PSNR values 47 dB, 46 dB, 44 dB, 42 dB, 40 dB, 38 dB, 37 dB, 36 dB, and 34 dB in sequence; and meanwhile, our algorithm produces PSNR values 52 dB, 49 dB, 46 dB, 43 dB, 41 dB, 40 dB, 38 dB, 36 dB, and 34 dB in sequence. It can be observed that, the PSNR values of our algorithm are much higher than those of P2 when the embedding rate is less than 0.3 bpp. From 0.4 bpp, our algorithm is still better than or as good as P2. According to the data in [14], the similar conclusion can be obtained for both Baboon and F—16. Actually, just as stated in [14], DE methods are better than PE-based methods at low embedding rates. Thodi et al. also stated that PE-based methods perform better at high embedding rates. However, due to our improvement, the proposed

<table>
<thead>
<tr>
<th>Embedding rate (bpp)</th>
<th>0.1029</th>
<th>0.2040</th>
<th>0.3051</th>
<th>0.4005</th>
<th>0.5035</th>
<th>0.6065</th>
<th>0.7057</th>
<th>0.8010</th>
<th>0.9040</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSNR1 (dB)</td>
<td>44.96</td>
<td>42.95</td>
<td>40.89</td>
<td>38.12</td>
<td>33.02</td>
<td>32.82</td>
<td>32.56</td>
<td>32.18</td>
<td>31.42</td>
</tr>
<tr>
<td>PSNR2 (dB)</td>
<td>46.03</td>
<td>44.46</td>
<td>42.78</td>
<td>40.31</td>
<td>35.18</td>
<td>34.76</td>
<td>34.24</td>
<td>33.52</td>
<td>32.24</td>
</tr>
</tbody>
</table>

**TABLE III: IMAGE QUALITY DIFFERENCE CAUSED BY EMBEDDING DIRECTIONS. THE RESULTS ARE OBTAINED BY PERFORMING TIAN’S DE ALGORITHM ON LENA. PSNR1 AND PSNR2 DENOTE THE PSNR VALUES OBTAINED FROM HORIZONTAL-VERTICAL EMBEDDING AND VERTICAL-HORIZONTAL EMBEDDING, RESPECTIVELY.**
DE-based method is still better than or as good as PE-based methods at medium and high embedding rates.

We briefly discuss the computational complexity of our algorithm before ending this section. Our simulation is carried out on a Pentium-4 3.44 GHz PC with 1 GB RAM. Our algorithm is implemented in Visual C 6.0. For Lena, from 0.1 bpp to 0.9 bpp, the embedding process takes about 0.250 s, 0.500 s, 0.594 s, 0.750 s, 1.000 s, 1.312 s, 1.718 s, 2.984 s, and 6.860 s, respectively. The corresponding detection process takes about 0.141 s, 0.188 s, 0.219 s, 0.235 s, 0.239 s, 0.242 s, 0.245 s, 0.247 s, and 0.250 s, respectively. It can be seen that the computational complexity of our algorithm is acceptable for general applications. When the embedding rate is close to the maximum capacity, the consuming time will be longer. However, the image quality obtained at such high embedding rates is often not used for commercial purposes.

V. CONCLUSION

DE embedding algorithms available in literature often use up all expandable differences in the current difference image before performing data embedding in the next difference image. Such a scheme may cause obvious distortion in the output image. Moreover, the algorithm cannot perform smoothly near the layer embedding capacity limit. In this paper, we have proposed a new embedding algorithm to overcome these problems. We introduce a dynamical expandable difference search and selection mechanism to balance the use of differences in two embedding directions. This mechanism effectively avoids severe embedding distortion resulting from the use of large differences in the previous difference image. We also exploit zero points and a payload-dependent overflow location map to improve the histogram-based difference selection and shifting scheme. Using zero points can decrease the histogram shifting distance and using the payload-dependent overflow location map can increase compression efficiency of the overflow location map.

The proposed algorithm does not have the original layer embedding capacity limit. It can perform smoothly at different embedding rates. The experimental results have proven that our algorithm often outperforms Thodi et al.’s newly published algorithm, Tian’s original DE algorithm, and Kamstra et al.’s Extended DE algorithm for different types of images. The advantage of our algorithm is more prominent around the embedding rate of 0.5 bpp.

ACKNOWLEDGMENT

The authors would like to thank the Associate Editor, Prof. I. Ahmad, and the anonymous reviewers for their valuable comments that have helped us improve the quality and clarity of the paper.

REFERENCES


Yongjian Hu (M’05) received the Ph.D. degree from South China University of Technology, Guangzhou, China, in 2002. He is a Professor of the College of Automation Science and Engineering, South China University of Technology. During 2000 to 2004, he visited the Department of Computer Science, City University of Hong Kong four times as a Research Assistant, Senior Research Associate, and Research Fellow, respectively. From 2005 to 2006, he worked as a Research Professor in the School of Information and Communication Engineering, Sung Kyun Kwan University, Korea. Since 2006, he has been visiting the Department of Computer Science, Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea, as a Visiting Professor. His research interests include information security, digital forensics, and image processing.

Heung-Kyu Lee received the B.S. degree in Electronics Engineering from the Seoul National University, Seoul, Korea, in 1978, and M.S. and Ph. D. degrees in computer science from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, in 1981, and 1984, respectively. From 1986, he is a Professor of the Department of Computer Science, KAIST. He is an author/coauthor of more than 100 international journal and conference papers. He is a reviewer for many international journals, including Journal of Electronic Imaging, IEEE Transactions on Circuits and Systems for Video Technology, etc. He was also a program chairman of many international conferences, including the International Workshop on Digital Watermarking (IWDW) in 2004, IEEE International conference on Real-Time Systems and Applications, etc. Since 2006, he has been a general director of Center of Fusion Technology for Security (CFTS). His major interests are information hiding and multimedia forensics.
Kaiying Chen received the B.E. degree from Xi’an University of Science and Technology, Xi’an, China, in 2005. She is currently pursuing the M.S. degree in pattern recognition and intelligent systems in the College of Automation Science and Engineering, South China University of Technology, Guangzhou. Her research interests include digital image processing, data hiding and pattern recognition.

Jianwei Li received the B.E. degree from Xi’an University of Technology, Xi’an, China, in 2005. He is currently pursuing his M.S. degree in pattern recognition and intelligent systems in the College of Automation Science and Engineering, South China University of Technology, Guangzhou. His research interests include digital image processing, data hiding and pattern recognition.