Reversible Data Embedding Using a Difference Expansion

Jun Tian

***Abstract*—Reversible data embedding has drawn lots of interest recently. Being reversible, the original digital content can be completely restored. In this paper, we present a novel reversible data-embedding method for digital images. We explore the redundancy in digital images to achieve very high embedding capacity, and keep the distortion low.**

***Index Terms*—Difference expansion (DE), location map, reversible data embedding, reversible integer transform.**

I. Introduction

REVERSIBLE data embedding, which is also called lossless data embedding, embeds invisible data (which is called a payload) into a digital image in a reversible fashion. As a basic requirement, the quality degradation on the image after data embedding should be low. An intriguing feature of reversible data embedding is the reversibility, that is, one can remove the embedded data to restore the original image.

From the information hiding point of view, reversible data embedding hides some information in a digital image in such a way that an authorized party could decode the hidden information and also restore the image to its original, pristine state. The performance of a reversible data-embedding algorithm can be measured by the following.

1) *Payload capacity limit:* what is the maximal amount of information can be embedded?
2) *Visual quality:* how is the visual quality on the embedded image?
3) *Complexity:* what is the algorithm complexity?

The motivation of reversible data embedding is distortion-free data embedding [1]. Though imperceptible, embedding some data will inevitably change the original content. Even a very slight change in pixel values may not be desirable, especially in sensitive imagery, such as military data and medical data. In such a scenario, every bit of information is important. Any change will affect the intelligence of the image, and the access to the original, raw data is always required.

From the application point of view, reversible data embedding can be used as an information carrier. Since the difference between the embedded image and original image is almost imperceptible from human eyes, reversible data embedding could be thought as a covert communication channel. By embedding its message authentication code, reversible data embedding provides a true self authentication scheme, without the use of metadata.

In this paper, we present a high-capacity, high visual quality, reversible data-embedding method for digital images. Our method can be applied to digital audio and video as well. We calculate the differences of neighboring pixel values, and select some difference values for the difference expansion (DE). The original content restoration information, a message authentication code, and additional data (which could be any data, such as date/time information, auxiliary data, etc.) will all be embedded into the difference values. In this paper we will consider grayscale images only. For color images, there are several options. One can decorrelate the dependence among different color components by a reversible color conversion transform [2], and then reversibly embed the data in the decorrelated components. Or one can reversibly embed each color component individually.

Please note that reversible data embedding is a fragile technique. When the embedded image is manipulated and/or lossy compressed, the decoder will find out it is not authentic and thus there will be no original content restoration.

A. Existing Methods

The earliest reference to reversible data embedding we could find is the Barton patent [3], filed in 1994. In his invention, the bits to be overlayed will be compressed and added to the bitstring, which will be embedded into the data block. Honsinger, *et al.* [4], reconstruct the payload from an embedded image, then subtract the payload from the embedded image to losslessly recover the original image. Macq [5] proposes an extension to the patchwork algorithm to achieve reversible data embedding. Fridrich, *et al.* [1], develop a high capacity reversible data-embedding technique based on embedding message on bits in the status of group of pixels. They also describe two reversible data-embedding techniques for lossy image format JPEG. De Vleeschouwer, *et al.* [6], propose a reversible data-embedding algorithm by circular interpretation of bijective transformations. Kalker, *et al.* [7], provide some theoretical capacity limits of lossless data compression based reversible data embedding [1] and give a practical code construction. Celik, *et al.* [8], [9], present a high capacity, low distortion reversible data-embedding algorithm by compressing quantization residues. They employ the lossless image compression algorithm CALIC, with quantized values as side-information, to efficiently compress quantization residues to obtain high embedding capacity.

B. Our Contributions

A common approach of high capacity reversible data embedding is to select an embedding area (for example, the least significant bits of some pixels) in an image, and embed both the payload and the original values in this area (needed for exact recovery of the original image) into such area. As the amount of information needed to be embedded (payload and

Manuscript received December 22, 2002; revised April 20, 2003.

The author is with ICompress Technologies, Tualatin, OR 97062 USA (e-mail: juntian@ieee.org).

Digital Object Identifier 10.1109/TCSVT.2003.815962

1051-8215/03$17.00 © 2003 IEEE

original values in the embedding area) is larger than that of the embedding area, techniques in [1], [7]–[9] rely on lossless data compression on the original values in the embedding area, and the space saved from compression will be used for embedding the payload. In [10] we introduced a DE technique, which discovers extra storage space by exploring the redundancy in the image content. We employ the DE technique to reversibly embed a payload into digital images. Both the payload capacity limit and the visual quality of embedded images of the DE method are among the best in the literature, along with a low computational complexity.

II. A Simple Example of the Difference Expansion

Assume we have two values $x = 206$, $y = 201$, we would like to reversibly embed one bit $b = 1$. First we compute the integer average l and difference h of x and y,

$$l = \left\lfloor \frac{206+201}{2} \right\rfloor = \left\lfloor \frac{407}{2} \right\rfloor = 203,\ h = 206 - 201 = 5$$

where the symbol $\lfloor \cdot \rfloor$ is the floor function meaning “the greatest integer less than or equal to.” Next we represent the difference value h into its binary representation $h = 5 = 101_2$. Then we append b into the binary representation of h after the least significant bit (LSB), the new difference value h' will be $h' = 101b_2 = 1011_2 = 11$. Mathematically, this is equivalent to

$$h' = 2 \times h + b = 2 \times 5 + 1 = 11.$$

Finally we compute the new values, based on the new difference value h' and the original integer average value l,

$$x' = 203 + \left\lfloor \frac{11+1}{2} \right\rfloor = 209,\ y' = 203 - \left\lfloor \frac{11}{2} \right\rfloor = 198.$$

From the embedded pair (x', y'), we can extract the embedded bit b and restore the original pair $(x,\, y)$. Again we compute the integer average and difference

$$l' = \left\lfloor \frac{209+198}{2} \right\rfloor = 203,\ h' = 209 - 198 = 11.$$

Look into the binary representation of h', $h' = 11 = 1011_2$. Extract the LSB, which is 1 in this case, as the embedded bit b, which leaves the original value of the difference as $h = 101_2 = 5$. Mathematically, this is equivalent to

$$b = \mathrm{LSB}(h') = 1,\ h = \left\lfloor \frac{h'}{2} \right\rfloor = 5.$$

Now with the integer average value l' and restored difference value h, we can restore exactly the original pair (x, y).

In this example, we have embedded one bit b by increasing the valid bit length of the difference value h from 3 bits (for $h = 5$) to 4 bits (for $h' = 11$). This reversible data-embedding operation

$$h' = 2 \times h + b$$

is called the DE.

III. Reversible Data Embedding

A. Reversible Integer Transform

We start with a simple reversible integer transform, which has already been used in Section II. For an 8 bits grayscale-valued pair (x, y), $x,\ y \in \mathbf{Z}$, $0 \leq x,\ y \leq 255$, define their integer average l and difference h as

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

The inverse transform of (1) is

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

The reversible integer transforms (1) and (2) are also called integer Haar wavelet transform, or the S transform. The reversible integer transforms set up a one-to-one correspondence between (x, y) and (l, h).

From (2), to prevent the overflow and underflow problems, i.e., to restrict $x,\ y$ in the range of [0, 255], it is equivalent to have

$$0 \leq l + \left\lfloor \frac{h+1}{2} \right\rfloor \leq 255,\ \text{and}\ 0 \leq l - \left\lfloor \frac{h}{2} \right\rfloor \leq 255.$$

Since both l and h are integers, one can derive that the above inequalities are equivalent to

$$|h| \leq 2(255 - l),\ \text{and}\ |h| \leq 2l + 1. \tag{3}$$

It is easy to see that Condition (3) is equivalent to

$$\begin{cases} |h| \leq 2(255 - l), & \text{if } 128 \leq l \leq 255. \\ |h| \leq 2l + 1, & \text{if } 0 \leq l \leq 127. \end{cases}$$

B. Expandable and Changeable Difference Values

As we embed a bit b into the difference value h by the DE, the new, expanded difference value h' will be

$$h' = 2 \times h + b.$$

According to Condition (3), to prevent overflow and underflow, h' should satisfy

$$|h'| \leq \min\left(2(255 - l), 2l + 1\right).$$

We formulate it as the following.

Definition 1: We say a difference value h is *expandable* under the integer average value l if

$$|2 \times h + b| \leq \min\left(2(255 - l), 2l + 1\right)$$

for both $b = 0$ and 1.

As the DE does not change the integer average value l, for simplicity, we will say h is expandable, as an abbreviation of h is expandable under l. For an expandable difference value h, if we embed a bit by the DE, the expanded difference value h' still satisfies Condition (3). The new pair computed from l and h' via (2) is guaranteed to have grayscale values. Thus expandable difference values are the candidates for the DE.

As each integer can be represented by the sum of a multiple of 2 and its LSB, for the new, expanded difference value h'

$$h' = 2 \times \left\lfloor \frac{h'}{2} \right\rfloor + \mathrm{LSB}(h')$$

with $\mathrm{LSB}(h') = 0$ or 1. If we modify its LSB

$$g = 2 \times \left\lfloor \frac{h'}{2} \right\rfloor + b'$$

with $b' = 0$ or 1, then

$$\begin{aligned} |g| &= \left| 2 \times \left\lfloor \frac{h'}{2} \right\rfloor + b' \right| = \left| 2 \times \left\lfloor \frac{2 \times h + b}{2} \right\rfloor + b' \right| \\ &= |2 \times h + b'| \le \min\left(2(255 - l), 2l + 1\right). \end{aligned}$$

Thus the difference value h' could have its LSB modified, without causing an overflow or underflow. We refer to such difference values as changeable.

Definition 2: We say a difference value h is *changeable* under the integer average value l if

$$\left| 2 \times \left\lfloor \frac{h}{2} \right\rfloor + b \right| \le \min\left(2(255 - l), 2l + 1\right),$$

for both $b = 0$ and 1.

Also for simplicity, we will say changeable, as an abbreviation of changeable under l. Note that $(2 \times \lfloor h/2 \rfloor + b)$ and h are either equal or different by LSB. By definition, it can be easily proved that

1) For a changeable difference value h, if we modify its LSB, the difference value after modification is still changeable.
2) An expandable difference value h is changeable.
3) After the DE, the expanded difference value h' is changeable.
4) If $h = 0$ or -1, the conditions on expandable and changeable are equivalent.

C. Data-Embedding Algorithm

In a digital image, one can select some expandable difference values of pixels, and embed one bit into each of them. To extract the embedded data and restore the original values, the decoder needs to know which difference values have been selected for the DE. To facilitate it, we need to embed such location information, such that the decoder could access and employ it for decoding. For this purpose, we will create and embed a location map, which contains the location information of all selected expandable difference values.

Furthermore, the decoder needs to know where (from which difference values) to collect and decode the location map. After the DE, the expanded difference value h' might not be expandable. On the decoder side, to check whether h' is expandable does not tell whether the original h has been selected for the DE during embedding. As we know, the expanded difference value h' is changeable, so the decoder could examine each changeable difference value. As with many image processing techniques, the encoder serves for the decoder, in the DE method, the encoder will take all changeable difference values as the embedding area, so that the decoder will use the same data to decode. During data embedding, we will modify all changeable difference values, by either adding a new LSB (via the DE) or modifying its LSB. To guarantee an exact recovery of the original image, we will also embed the original values of those modified LSBs. In brief, the date-embedding DE algorithm consists of six steps: calculating the difference values, partitioning difference values into four sets, creating a location map, collecting original LSB values, data embedding by replacement, and finally an inverse integer transform. We discuss each step below.

First, the original image is grouped into pairs of pixel values. A pair consists of two neighboring pixel values or two with a small difference value. The pairing could be done horizontally by pairing the pixels on the same row and consecutive columns $(i, 2j - 1)$ and $(i, 2j)$, or vertically, or by a key-based specific pattern. The pairing could be through all pixels of the image or just a portion of it. We apply the integer transform (1) to each pair. Then we design a scanning order for all the difference values h, and order them as a one dimensional list $\{h_1, h_2, \cdots, h_n\}$.

Next we create four disjoint sets of difference values, EZ, EN, CN, and NC:

1) EZ: contains all expandable $h = 0$ and expandable $h = -1$.
2) EN: contains all expandable $h \notin \mathrm{EZ}$.
3) CN: contains all changeable $h \notin (\mathrm{EZ} \cup \mathrm{EN})$.
4) NC: contains all nonchangeable h.

Each difference value will fall into one and only one of the above four sets. As an expandable difference value is changeable, the whole set of changeable difference values is $\mathrm{EZ} \cup \mathrm{EN} \cup \mathrm{CN}$.

Third, we create a location map of selected expandable difference values. For every difference value h in EZ, it will be selected for the DE. For EN, depending on the payload size, some difference values will be selected for the DE. The selection of difference values in EN will be discussed in the next subsection. For convenience, we denote the subsets of selected and not selected difference values in EN as EN1 and EN2, respectively. We create a one-bit bitmap as the location map, with its size equal to the numbers of pairs of pixel values (in Step 1). For example, if we use the horizontal pairing through all pixels, the location map will have the same height as the image, and half the width. For an h in $\mathrm{EZ} \cup \mathrm{EN1}$, we assign a value 1 in the location map; for an h in $\mathrm{EN2} \cup \mathrm{CN} \cup \mathrm{NC}$, we assign a value 0. Thus a value 1 will indicate a selected expandable difference value. The location map will be losslessly compressed by a JBIG2 compression or run-length coding. The compressed bitstream is denoted as $\mathcal{L}$. An end of message symbol is at the end of $\mathcal{L}$.

Fourth, we collect original LSBs of difference values in EN2 and CN. For each h in $\mathrm{EN2} \cup \mathrm{CN}$, $\mathrm{LSB}(h)$ will be collected into a bitstream $\mathcal{C}$. However for those $h = 1$ or $h = -2$ in $\mathrm{EN2} \cup \mathrm{CN}$, their LSBs will be not collected, as such values (1 and 0, respectively) could be determined by the location map (which will be clear in the decoding section).

Fifth, we embed the location map $\mathcal{L}$, the original LSBs $\mathcal{C}$, and a payload $\mathcal{P}$. The payload $\mathcal{P}$ includes an authentication hash of the original image (for example, a message authentication

TABLE I
EMBEDDING ON DIFFERENCE VALUES

Category	Original Set	Original Value	Location Map Value	New Value	New Set
Changeable	EZ or EN1	h	1	$2 \times h + b$	CH
	EN2 or CN	h	0	$2 \times \left\lfloor \frac{h}{2} \right\rfloor + b$	
Non-changeable	NC	h	0	h	NC

code). The payload size (bit length) is limited by the payload capacity limit and will be discussed in the next subsection. We combine $\mathcal{L}$, $\mathcal{C}$, and $\mathcal{P}$ together into one binary bitstream $\mathcal{B}$

$$\mathcal{B} = \mathcal{L} \cup \mathcal{C} \cup \mathcal{P} = b_1 b_2 \cdots b_m$$

where $b_i \in \{0, 1\}$, $1 \leq i \leq m$, and m is the bit length of $\mathcal{B}$. Here, we append $\mathcal{C}$ to the end of $\mathcal{L}$, and append $\mathcal{P}$ to the end of $\mathcal{C}$. The data embedding (by replacement) will be

```
1) Set $i = 1$ and $j = 0$.
2) While $(i \leq m)$
        $j = j + 1$.
        If $h_j \in (\mathrm{EZ} \cup \mathrm{EN1})$
            $h_j = 2 \times h_j + b_i$.
            $i = i + 1$.
        Elseif $h_j \in (\mathrm{EN2} \cup \mathrm{CN})$
            $h_j = 2 \times \lfloor h_j/2 \rfloor + b_i$.
            $i = i + 1$.
3) End
```

The above scheme is also illustrated in Table I, as Column 5 shows the new value of h after embedding, depending on which set (Column 2) it originally belongs to.

Finally after all bits in $\mathcal{B}$ are embedded, we apply the inverse integer transform (2) to obtain the embedded image.

D. Discussions

1) Capacity: The bitstream $\mathcal{B}$ has a bit length of $(|\mathcal{L}| + |\mathcal{C}| + |\mathcal{P}|)$, where $|\cdot|$ is the cardinality (bit length or numbers of elements) of a set. From the date-embedding algorithm, the total embedding capacity will be $(|\mathrm{EZ}| + |\mathrm{EN1}| + |\mathrm{EN2}| + |\mathrm{CN}|)$. Accordingly, to have $\mathcal{B}$ successfully embedded, we must have

$$|\mathcal{L}| + |\mathcal{C}| + |\mathcal{P}| \leq |\mathrm{EZ}| + |\mathrm{EN1}| + |\mathrm{EN2}| + |\mathrm{CN}|. \quad (4)$$

Assume the total number of 1 and -2 in $\mathrm{EN2} \cup \mathrm{CN}$ is N, then

$$|\mathrm{EN2}| + |\mathrm{CN}| = |\mathcal{C}| + N.$$

Substitute into (4), we derive

$$|\mathcal{P}| \leq |\mathrm{EZ}| + |\mathrm{EN1}| + N - |\mathcal{L}|. \quad (5)$$

Thus the payload size is upper bounded by the sum of the number of selected expandable difference values and the number of not selected or not expandable $h = 1$ and $h = -2$, minus the bit length of the location map. From (5), most of the payload capacity is from $\mathrm{EZ} \cup \mathrm{EN1}$ (selected expandable difference values); except for those $h = 1$ and $h = -2$, the payload capacity from $\mathrm{EN2} \cup \mathrm{CN}$ (not selected but expandable or changeable but not expandable difference values) is zero. As $(|\mathrm{EZ}| + |\mathrm{EN1}| + N)$ could not exceed 0.5 bpp (bit per pixel), $\mathcal{P}$ is less than 0.5 bpp.

2) Expandable Difference Value Selection: To achieve the payload capacity limit, all expandable difference values will be selected for the DE, i.e., $\mathrm{EN1} = \mathrm{EN}$, $\mathrm{EN2} = \emptyset$. For a payload whose size is less than the payload capacity limit, the selection is constrained by Condition (5). We present two simple selection methods here.

For a grayscale-valued pair (x, y), assume after the DE the new grayscale-valued pair is (x', y'). As the integer average value l is unchanged,

$$x - x' \approx y' - y,$$

where the approximation (instead of equality) is due to the rounding of the floor function $\lfloor \cdot \rfloor$. We have

$$\begin{aligned}(x - x')^2 + (y - y')^2 &\approx 2 \times (y - y')^2 \\ &= 2 \times \left(\left\lfloor \frac{h}{2} \right\rfloor - \left\lfloor \frac{h'}{2} \right\rfloor \right)^2 \approx \frac{h^2}{2}.\end{aligned}$$

Thus the Euclidean distance between (x, y) and (x', y') is proportional to the difference value h. To minimize the mean square error between the original image and embedded image, we should select h with small magnitudes for the DE. We choose a threshold T, and partition EN into EN1 and EN2 by

$$\mathrm{EN1} = \{h \in \mathrm{EN} : |h| \leq T\}, \ \mathrm{EN2} = \{h \in \mathrm{EN} : |h| > T\}.$$

For a payload $\mathcal{P}$, we start with a small threshold T, then increase T gradually until Condition (5) is met. Then one can fine tune the selection by flipping some expandable difference values $h = \pm T$ (or $|h|$ close to T) from EN1 to EN2, such that $(|\mathrm{EZ}| + |\mathrm{EN1}| + N - |\mathcal{L}|)$ is equal to the payload size. One could preprocess an image and create a threshold versus capacity table, by calculating $(|\mathrm{EZ}| + |\mathrm{EN1}| + N - |\mathcal{L}|)$. When embedding a payload, one could check this table and pick an appropriate threshold.

Alternately, we can define a hiding ability of an expandable difference value, as follows.

Definition 3: For the difference value h and integer average value l, if $k \geq 1$ is the largest integer such that

$$|k \times h + b| \leq \min\left(2(255 - l), 2l + 1\right),$$

for all $0 \leq b \leq k - 1$, then we say the *hiding ability* of h is $\log_2 k$.

For a difference value h with hiding ability $\log_2 k$, we could replace h with a new difference value $k \times h + b$, where $b \in \{0, \cdots, k - 1\}$, without causing an overflow or underflow. This means we could reversibly embed $\log_2 k$ bit. For an expandable difference value, as k will be at least 2, its hiding ability will be at least $\log_2 2 = 1$. Although in this paper we are not going to embed more than one bit into a difference value, the hiding ability could be used as a guide on selecting expandable difference values for the DE. In general, selecting an expandable difference value with large hiding ability will degrade less on the visual quality. A large hiding ability implies that the average of two pixel values is close to mid tone, while their difference is close to zero. Again we can choose a threshold T, and partition EN into EN1 and EN2 by

$$\mathrm{EN1} = \{h \in \mathrm{EN} : \mathrm{HidingAbility}(h) \geq T\}$$
$$\mathrm{EN2} = \{h \in \mathrm{EN} : \mathrm{HidingAbilility}(h) < T\}.$$

Please note that with a different threshold T in the above two selection methods, the location map $\mathcal{L}$ also changes, so does its bit length $|\mathcal{L}|$. Thus a third method to partition EN could be based on the compressibility of the location map. We may select expandable difference values such that the location map is more compressible. The third method is currently under study and will be reported in a forthcoming paper.

3) Multiple-Layer Embedding: For the date-embedding DE algorithm, we can apply it to an image more than once for multiple-layer embedding. For an already embedded image, we can embed it again with another payload. Even for one payload, we can divide the payload into several pieces and use multiple-layer embedding to embed them. As we have a choice of pairing of pixel values in Step 1 during embedding, we can use a different pairing for each layer. As each layer has a payload capacity limit less than 0.5 bpp, a multiple-layer embedding will have a payload capacity limit less than $M/2$ bpp, where M is the number of layers. In practice, the payload capacity limit of each layer will decrease gradually, as the redundancy in pixel values becomes less and less in a multiple-layer embedding.

To assist the decoder to determine whether or not it is a multiple-layer embedding, we embed a 16 bits header information before the location map $\mathcal{L}$. The bitstream $\mathcal{B}$ now becomes

$$\mathcal{B} = \mathcal{H} \cup \mathcal{L} \cup \mathcal{C} \cup \mathcal{P}$$

where $\mathcal{H}$ is a 16 bits header. For the original image (first layer), $\mathcal{H}$ is set to 0. The pairing pattern of the original image will be the $\mathcal{H}$ at the second layer embedding. The pairing pattern of the second layer embedding will be the $\mathcal{H}$ at the third layer embedding, and so on.

4) True Fidelity at Half Resolution: The DE method does not change integer averages of pixel values. During embedding, if we divide the image into 2 × 2 blocks and use a pairing (horizontal, vertical, or diagonal) inside each block, then the mean value of each 2 × 2 block on the embedded image will be the same as the original image, except for some rounding errors from integer averages. Thus except for rounding errors, at half resolution (by reducing both the height and width by half), the embedded image will be identical to the original image. When the embedded image is previewed at half or lower resolution, the visual difference from the original image at the same resolution will be imperceptible. We will give some examples in Section V.

5) Security: For security, the bitstream $\mathcal{B}$ can be encrypted by the Advanced Encryption Standard (AES) algorithm prior to embedding. The security discussion is beyond the scope of this paper and we will not discuss it in detail here.

IV. Decoding and Authentication

Referring to Table I, we can retrieve the embedded bitstream $\mathcal{B}$, by collecting LSBs of all changeable difference values. From $\mathcal{B}$, we can decode the location map $\mathcal{L}$ and the original LSBs $\mathcal{C}$. The location map gives the location information of all expanded difference values. For expanded difference values, an (integer) division by 2 will give back their original values; for other changeable difference values, we restore their original LSBs from the bitstream $\mathcal{C}$. After all changeable difference values have restored their original values, we can restore the original image exactly.

The decoding and authentication process consists of five steps. First we calculate the difference values. For an image, we do the pairing using the same pattern as in the embedding, and apply the integer transform (1) to each pair. We use the same scanning order to order all difference values as a one dimensional list $\{h_1, h_2, \cdots, h_n\}$.

Next we create two disjoint sets of difference values, CH, and NC:

1) CH: contains all changeable h.
2) NC: contains all nonchangeable h.

Note that we do not examine expandability at the decoder. For an embedded image, the set CH (which is after embedding) will be identical to EZ∪EN∪CN (which is before embedding). This invariant feature (which is an invariant set containing location information of changeable difference values) is the key for exact recovery.

Third we collect LSBs of all difference values in CH, and form a binary bitstream $\mathcal{B} = b_1 b_2 \cdots b_m$.

Fourth, we decode the location map from $\mathcal{B}$ by a JBIG2 decoder. As the JBIG2 bitstream has an end of message symbol at its end, the decoder knows exactly the location in $\mathcal{B}$, where it is the last bit from the location map bitstream $\mathcal{L}$. Assume the first s bits in $\mathcal{B}$ are the location map bitstream $\mathcal{L}$ (including the end of message symbol). We restore the original values of differences as follows.

```
1) Set $\mathrm{i} = \mathrm{s} + 1$.
2) For $\mathrm{j} = 1 : \mathrm{n}$
  If $h_j \in \mathrm{CH}$
    If the location map value at $h_j$ is 1
      $h_j = \lfloor h_j/2 \rfloor$.
    Else
      If $(0 \leq h_j \leq 1)$
        $h_j = 1$.
      Elseif $(-2 \leq h_j \leq -1)$
        $h_j = -2$.
      Else
        $h_j = 2 \times \lfloor h_j/2 \rfloor + b_i$.
        $i = i + 1$.
3) End
```

If the location map value is 1, the difference value has been expanded during embedding. Conversely, for a nonchangeable

TABLE II
EMBEDDED PAYLOAD SIZE VS PSNR OF EMBEDDED "LENA" IMAGE

Payload Size (bits)	39566	63676	84066	101089	120619	141493	175984	222042	260018	377869	516794
Bit Rate (bpp)	0.1509	0.2429	0.3207	0.3856	0.4601	0.5398	0.6713	0.8470	0.9919	1.4415	1.9714
PSNR (dB)	44.20	42.86	41.55	40.06	37.66	36.15	34.80	32.54	29.43	23.99	16.47

difference value, its location map value must be 0, otherwise the image has been tampered. For a changeable difference value h, if its location map value is 0, then its original value will be either equal to h or different from h by LSB. If $h \in \mathrm{CH}$, $0 \leq h \leq 1$, and its location map value is 0, then the original value of h must be 1. The reason is that the original value could be only either 0 or 1, as it is differed from h at most by LSB. If the original value of h was 0, then it would be an expandable zero (as changeable zero is expandable), and its location map value would be 1, which contradicts the fact that the location map value is 0. Similarly if $h \in \mathrm{CH}$, $-2 \leq h \leq -1$, and its location map value is 0, then the original value of h must be -2. For other changeable difference values with location map value 0, we restore their original LSBs from the embedded bitstream $\mathcal{C}$. After all difference values have been restored to their original values, the remaining bits in $\mathcal{B}$ is the embedded payload $\mathcal{P}$.

The fifth and last step is content authentication and original content restoration. We apply the inverse integer transform (2) to reconstruct a restored image. To authenticate the content, we compare the authentication hash in $\mathcal{P}$ with the hash of the restored image. If they match exactly, then the image content is authentic, and the restored image will be exactly the same as the original image. (Most likely a tampered image will not go through to this step because some decoding error could happen in Step 4, as a nonchangeable difference value might have a location map value 1 or a syntax error in the JBIG2 bitstream.)

For a multiple-layer embedding, the first 16 bits in $\mathcal{B}$ is the pairing pattern $\mathcal{H}$. After the first 16 bits are extracted, we decode the location map, reconstruct a restored image, and authenticate the content. If the content is authentic, we use $\mathcal{H}$ as the pairing pattern to decode the restored image again. The decoding process will continue until $\mathcal{H} = 0$ or when tampering has been discovered. If $\mathcal{H} = 0$, and no tampering has been discovered, then the final restored image will be exactly the same as the original image, pixel by pixel, bit by bit.

V. EXPERIMENTAL RESULTS

All experiments were performed by embedding and decoding an actual bitstream. We use the MATLAB function rand() to generate pseudo random numbers, then round them to nearest integers to generate a payload.

We apply the DE method to various standard grayscale test images, including the University of Waterloo BragZone collection and the JPEG 2000 compression test images. Here we include the results for the 512 × 512, 8 bpp grayscale "Lena." The embedded payload size, its corresponding bit rate, and the peak signal to noise ratio (PSNR) of embedded images are listed in Table II. The embedded "Lena" images with payloads of 39 566 bits, 141 493 bits, and 516 794 bits are shown in Figs. 1, 2, and 3, respectively. We do not include any results with a PSNR higher than 44 dB, as the quality degradation at 44 dB is almost imperceptible, and the original image can be completely restored. At a very low 16 dB in Fig. 3, the image content is still preserved.

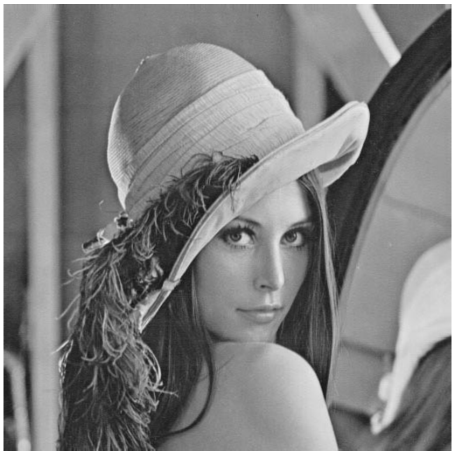

Fig. 1. Reversibly embedded "Lena," with a 39 566 bits (0.15 bpp) payload.

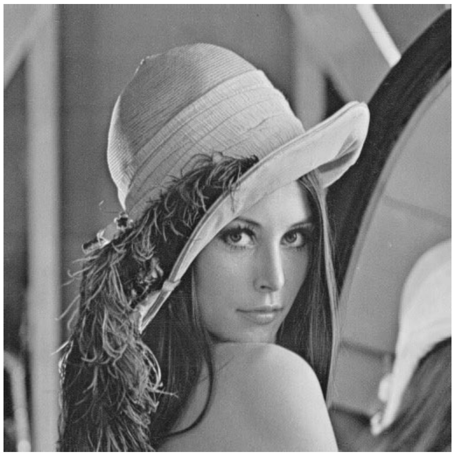

Fig. 2. Reversibly embedded "Lena," with a 141 493 bits (0.54 bpp) payload.

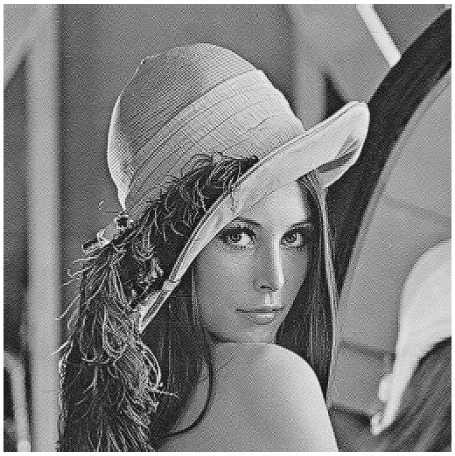

Fig. 3. Reversibly embedded "Lena," with a 516 794 bits (1.97 bpp) payload.

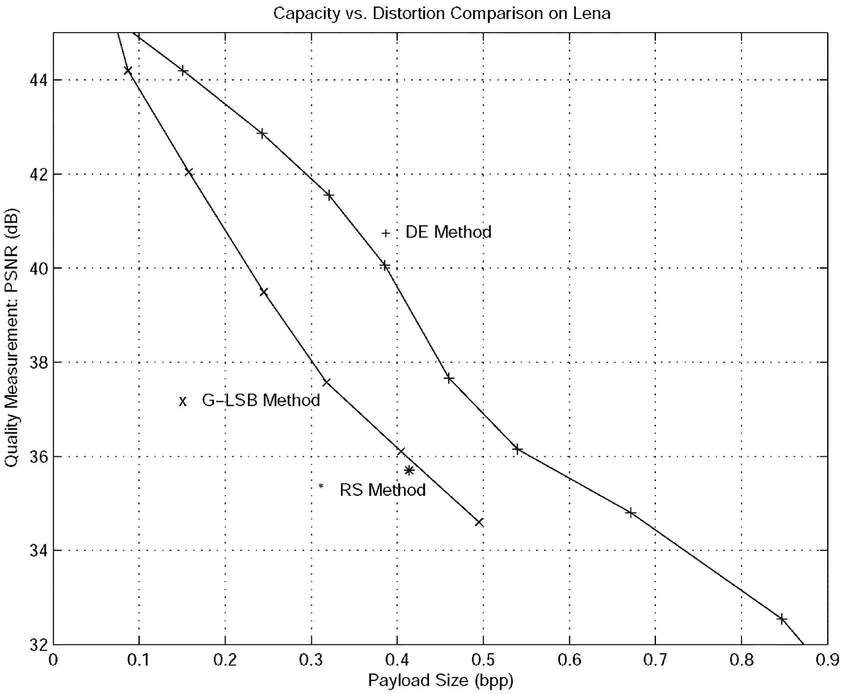

Fig. 4. Capacity versus distortion comparison of “Lena.”

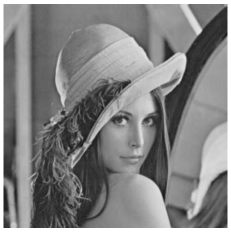

Fig. 5. Half resolution (256 × 256) of reversibly embedded “Lena” in Fig. 3.

As the DE increases the magnitudes of difference values, the embedding has the effect similar to mild sharpening in the mid tone regions.

We compare the results with [1], [8], and [9], namely the RS lossless date-embedding method in [1] and the lossless G-LSB date-embedding method in [8] and [9]. For the G-LSB method, the authors presented several implementations, and the selective embedding extension [9] gives the best capacity versus distortion results, thus we will use it for comparison. As the lack of a universal image quality measurement tool, we use PSNR as the distortion measurement. The capacity versus distortion comparison among the DE method, the G-LSB method, and the RS method on “Lena” is shown in Fig. 4. The top curve is the DE method. Except for images with large very smooth regions, the payload capacity limit of the G-LSB method does not exceed 1 bpp, while the DE method could easily embed more than 1 bpp. The payload capacity limit of the RS method is lower than both the G-LSB and the DE method. By embedding a payload of the same size (bit length), the embedded “Lena” image by the DE method is about 2 to 3 dB higher than both the G-LSB and the RS method.

Another feature of the DE method is the true fidelity at half resolution. The embedded image in Fig. 3 is from a multiple-layer embedding using horizontal and vertical pairing alternatively. We apply a 2 × 2 mean filter to generate a half resolution, 256 × 256 image, which is shown in Fig. 5. Its visual quality is imperceptible from the original “Lena” image at half resolution.

VI. Conclusions

In this paper, we have presented a simple and efficient reversible date-embedding method for digital images. We explored the redundancy in the digital content to achieve reversibility. Both the payload capacity limit and the visual quality of embedded images are among the best in the literature.

References

[1] J. Fridrich, M. Goljan, and R. Du, “Lossless data embedding—new paradigm in digital watermarking,” *EURASIP J. Appl. Signal Processing*, vol. 2002, no. 2, pp. 185–196, Feb. 2002.

[2] J. Tian, “Wavelet-based reversible watermarking for authentication,” in *Security and Watermarking of Multimedia Contents IV—Proc. SPIE*, E. J. Delp III and P. W. Wong, Eds., Jan. 2002, vol. 4675, pp. 679–690.

[3] J. M. Barton, “Method and Apparatus for Embedding Authentication Information Within Digital Data,” U.S. Patent 5 646 997, 1997.

[4] C. W. Honsinger, P. W. Jones, M. Rabbani, and J. C. Stoffel, “Lossless recovery of an original image containing embedded data,” U.S. Patent 6 278 791, 2001.

[5] B. Macq, “Lossless multiresolution transform for image authenticating watermarking,” in *Proc. EUSIPCO*, Sept. 2000, pp. 533–536.

[6] C. De Vleeschouwer, J. F. Delaigle, and B. Macq, “Circular interpretation of bijective transformations in lossless watermarking for media asset management,” *IEEE Tran. Multimedia*, vol. 5, pp. 97–105, Mar. 2003.

[7] T. Kalker and F. M. J. Willems, “Capacity bounds and constructions for reversible data hiding,” in *Proc. 14th Int. Conf. Digital Signal Processing*, vol. 1, July 2002, pp. 71–76.

[8] M. U. Celik, G. Sharma, A. M. Tekalp, and E. Saber, “Reversible data hiding,” in *Proc. Int. Conf. Image Processing*, vol. II, Sept. 2002, pp. 157–160.

[9] ——, “Lossless generalized-LSB data embedding,” *IEEE Trans. Image Processing*, submitted for publication.

[10] J. Tian, “Reversible watermarking by difference expansion,” in *Proc. Workshop on Multimedia and Security*, J. Dittmann, J. Fridrich, and P. Wohlmacher, Eds., Dec. 2002, pp. 19–22.