HIGH CAPACITY REVERSIBLE DATA EMBEDDING AND CONTENT AUTHENTICATION

Jun Tian

Digimarc Corporation
19801 SW 72nd Avenue, Tualatin, OR 97062, USA
juntian@ieee.org

ABSTRACT

In this paper we present a high capacity reversible data embedding algorithm. It serves for the purposes of both self authentication and reversible data embedding. As being reversible, the original digital content (before data embedding) can be completely restored after authentication. We employ two techniques, difference expansion and generalized least significant bit embedding, to achieve very high embedding capacity, while keep the distortion (the quality degradation on the digital content after data embedding) low. A noticeable difference between our method and others is that we do not need to compress original values of the embedding area. We explore the redundancy in the digital content to achieve reversibility.

1. INTRODUCTION

Reversible data embedding [1, 2, 3, 4, 5, 6, 7, 8], which is also called reversible watermarking, has drawn lots of interest recently. It embeds invisible data (which is called a payload) into a digital content in a reversible fashion. As a basic requirement, the quality degradation on the digital content after data embedding should be low. A intriguing feature of reversible data embedding is the reversibility, that is, when the digital content has been authenticated, one can remove the embedded data to restore the original content (before data embedding). Such reversibility to get back original content is highly desirable in sensitive imagery, such as military data and medical data.

Compared with authentication techniques in cryptography, reversible data embedding does not change the file syntax of a digital content. A legacy viewer or player can still view or play the embedded digital content. Secondly, the embedded data becomes an inherent part of the content, and is robust against (lossless) file format conversion, in contrast to a cryptography authentication hash has to be appended as extra metadata. In addition, reversible data embedding provides high capacity data embedding without increasing the storage space (file size) of the digital content.

In this paper, we present a reversible data embedding method for digital images. Our method can be applied to digital audio and video as well. We employ two techniques, difference expansion (DE) [7] and generalized least significant bit (G-LSB) embedding [2], to achieve very high embedding capacity, while keep the distortion low. We calculate the differences of neighboring pixel values, and select some difference values for DE. The original G-LSBs of difference values, the location of expanded difference values, and a payload (which includes an authentication hash of the original image) will all be embedded into the difference values, where the extra storage space is obtained by DE.

In this paper we will consider grayscale images only. For color images, one can embed the data into each color component individually. Or one can decorrelate the dependence among different color components, and then embed the data into the decorrelated components, as in [6].

2. REVERSIBLE DATA EMBEDDING

A general reversible data embedding diagram is illustrated in Fig. 1. We embed a payload into a digital image *I* by (slightly) modifying its pixel values, and obtain the embedded image *I'*. Before sending it to the decoder, *I'* might or might not have been tampered by some intentional or unintentional attack. If the decoder finds that no tampering happened in *I'*, i.e., *I'* is authentic, then the decoder can remove the embedded payload from *I'* to restore the original image, which results in a new image *I"*. By definition of reversible data embedding, the restored image *I"* will be **exactly** the same as the original image *I*, pixel by pixel, bit by bit.

A basic approach of 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 original values in the embedding area) is larger than that of the embedding area, most reversible data embedding techniques [1, 2, 3, 5] 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 [6] we presented a DE technique, which removes the need of

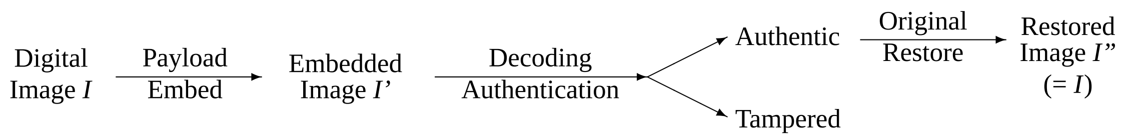

Fig. 1. Reversible data embedding diagram.

lossless compression on original values in the embedding area. The DE technique discovers extra storage space by exploring the high redundancy in the image content.

2.1. Reversible Integer Transform

We start with a simple reversible integer transform. For a grayscale-valued pair (x, y), $x, y \in \mathbf{Z}$, $0 \le x, y \le 255$, define their integer average and difference as

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

where the symbol $\lfloor\cdot\rfloor$ is the floor function meaning "the greatest integer less than or equal to". 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 transform (1) and (2) are also called integer Haar wavelet transform, or the S transform [9].

We will select the magnitudes of difference values h for embedding. As grayscale values are bounded in $[0, 255]$,

$$0 \le l + \left\lfloor \frac{h+1}{2} \right\rfloor \le 255,\ 0 \le l - \left\lfloor \frac{h}{2} \right\rfloor \le 255$$

which is equivalent to

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

To prevent overflow and underflow problems, the difference value h (after data embedding) must satisfy Condition (3).

2.2. Changeable and Expandable Difference Value

Given an integer L, $L \in \mathbf{Z}$, $L \ge 2$, the (L-level) G-LSB, g, of a difference value h, is the reminder of its magnitude divided by L,

$$g := |h| - \left\lfloor \frac{|h|}{L} \right\rfloor \cdot L$$

The G-LSBs of *changeable* difference values will be the selected embedding area for our method.

Definition 1 *For a grayscale-valued pair* (x, y)*, we say its difference value* h *is* L-changeable *if*

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

During data embedding, the G-LSB g might be replaced by a value from the reminder set $\{0, 1, \cdots, L-1\}$. Due to constraint (3), some large reminders might cause an overflow or underflow. Thus we could only replace g with a value from a partial reminder set $\{0, 1, \cdots, M\}$, with $g \le M \le L - 1$, where M is determined by l and $\left\lfloor \frac{|h|}{L} \right\rfloor$.

Modifying G-LSBs of L-changeable h (without compression) does not provide extra storage space. We gain extra storage space from *expandable* difference values.

Definition 2 *For a grayscale-valued pair* (x, y)*, we say its difference value* h *is* L-expandable *if*

$$|h| \cdot L + 1 \le \min(2(255 - l), 2l + 1)$$

In a base L representation, an L-expandable h could add one extra number b after its G-LSB. More precisely, h could be replaced by h', without causing an overflow or underflow

$$h' = \text{sign}(h) \cdot (|h| \cdot L + b)$$

Again, due to constraint (3), b could be a value from a partial reminder set $\{0, 1, \cdots, M\}$, with $1 \le M \le L - 1$, and M is determined by l and $|h|$. Thus for each L-expandable difference value, one could gain $\log_2(M+1)$ extra bits. The reversible operation from h to h' is called *difference expansion*. An L-expandable h is L-changeable. After DE, the expanded h' is L-changeable. Note that if $h = 0$, the conditions on L-changeable and L-expandable are equivalent.

We could also embed on the difference value h directly instead of its magnitude. For this purpose, we modify L-changeable (and consequently L-expandable) for $h < 0$ as below, define DE as $h' = h \cdot L + b$, where $0 \le b \le L - 1$, and carry a slightly different scheme

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

2.3. Embedding Algorithm

An 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 through all pixels of the image or just a portion of it. (To achieve maximum embedding capacity, we could embed a payload with one pairing, then embed another payload with another pairing on the embedded image. For example, we could embed columnwise first, then embed rowwise.) We apply the integer transform (1) to each pair.

Next we create four disjoint sets of difference values:

1. EZ: contains all L-expandable $h = 0$.
2. EN: contains all L-expandable $h \notin$ EZ.
3. CNE: contains all L-changeable $h \notin$ EZ $\cup$ EN.
4. NC: contains all not L-changeable h.

Each difference value will fall into one and only one set. EZ $\cup$ EN $\cup$ CNE contains all changeable difference values.

The third step is to create a location map of selected expandable difference values. All difference values in EZ will be selected for DE. For EN, depending on the payload size, some difference values will be selected for DE. A discussion on expandable difference values selection can be found in [7]. 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 an h in either EN1 or EZ, we assign a value 1; for an h in EN2, CNE, or NC, we assign a value 0. The location map will be losslessly compressed by a JBIG2 compression or run-length coding. The compressed bit stream is denoted as $\mathcal{L}$. An end of message symbol is appended at the end of $\mathcal{L}$.

Fourth, we collect original values of G-LSBs of difference values in EN2 and CNE. For each h in EN2 or CNE, its G-LSB g will be collected into a bit stream $\mathcal{C}$. We employ the L-ary to Binary conversion method of [2] to convert g to a binary bit stream. The L-ary to Binary conversion is a division scheme of unit interval, similar to arithmetic coding. As h is L-changeable, we determine M, where g could be replaced by a value from the partial reminder set $\{0, 1, \cdots, M\}$ without causing an overflow or underflow. We convert g to the interval $[\frac{g}{M+1}, \frac{g+1}{M+1})$. This interval will be further refined by the next G-LSBs, and so on, until we go through all G-LSBs in EN2 $\cup$ CNE. Then we decode the final interval to a binary bit stream. By using L-ary to Binary conversion, the representation of G-LSBs will be more compact, which results in a smaller bit stream size of $\mathcal{C}$. Note that if $|h| \leq L - 1$, after its g is collected, we also store its sign, sign(h), in the bit stream $\mathcal{C}$. (If we define DE as $h' = h \cdot L + b$, there will be no need to store the signs of difference values.)

Fifth, we embed the location map $\mathcal{L}$, the original G-LSBs $\mathcal{C}$, and a payload $\mathcal{P}$ (which includes an authentication hash, for example, a message authentication code). We combine them together into one binary bit stream $\mathcal{S}$,

$$\mathcal{S} = \mathcal{L} \cup \mathcal{C} \cup \mathcal{P}$$

We use the inverse L-ary to Binary conversion to convert the binary bit stream $\mathcal{S}$ to M-ary, with M determined for each expandable difference value in EZ and EN1, and each changeable difference value in EN2 and CNE. The embedding (by replacement) will be

- EZ: $|h| = b$, where b is the M-ary symbol from the inverse L-ary to Binary conversion, and the sign of h could be assigned pseudo randomly.
- EN1: $h = \text{sign}(h) \cdot (|h| \cdot L + b)$.
- EN2 or CNE: $h = \text{sign}(h) \cdot \left(\left\lfloor \frac{|h|}{L} \right\rfloor \cdot L + b \right)$.
- NC: no change on the value of h.

After all embedding is done, we apply the inverse integer transform (2) to obtain the embedded image.

2.4. Decoding Algorithm

First we do the pairing using the same pattern as in the embedding, and apply the integer transform (1) to each pair.

Next we create two disjoint sets of difference values:

1. C: contains all L-changeable h.
2. NC: contains all not L-changeable h.

The set C (after embedding) will be identical to EZ $\cup$ EN $\cup$ CNE (before embedding). This invariant feature (which is an invariant set in this case) is the key for exact recovery.

Third we collect all G-LSBs of difference values in C. We employ the L-ary to Binary conversion to convert it into a binary bit stream $\mathcal{B}$.

Fourth, from the binary bit stream, we decode the location map. With the location map, we restore the original values of difference values as follows:

- if $h \in$ C, the location map value is 1, then $h = \text{sign}(h) \cdot \left\lfloor \frac{|h|}{L} \right\rfloor$.
- if $h \in$ C, the location map value is 0, and $h = 0$, decode an M-ary symbol b from $\mathcal{B}$, and decode a sign value s from $\mathcal{B}$, then $h = s \cdot b$.
- if $h \in$ C, the location map value is 0, and $1 \leq |h| \leq L - 1$, then $h = \text{sign}(h) \cdot b$, and the next sign value from $\mathcal{B}$ should correctly match sign(h).
- if $h \in$ C, the location map value is 0, and $|h| \geq L$, then $h = \text{sign}(h) \cdot \left(\left\lfloor \frac{|h|}{L} \right\rfloor \cdot L + b \right)$.
- if $h \in$ NC, the location map value should be 0, no change on the value of h.

After all difference values have been restored to their original values, we apply the inverse integer transform (2) to reconstruct a restored image. For content authentication, we extract the payload $\mathcal{P}$ from $\mathcal{B}$ (which will be the remaining after Step 4). We then 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. Note that we use a slightly different order of operations from Fig. 1. We reconstruct a restored image *I"* first, then authenticate the content of the embedded *I'*.

3. EXPERIMENTAL RESULTS

For the maximum embedding capacity results listed below, we expand all expandable difference values (EN1 = EN). The image is embedded twice, first with a columnwise pairing, then a rowwise pairing.

Table 1 lists the results on the 512×512, 8 bits per pixel (bpp), grayscale Lena image. For $L = 2, 3, 4$, the maximum embedding capacity (in bits), along with its bit rate, are shown. At $L = 2$, the maximum embedding capacity (259927 bits) is already above the best result in the literature, which is around 200000 bits.

L	2	3	4
max capacity (bits)	259927	394981	466605
bit rate (bpp)	0.9915	1.5067	1.7800

Table 1. Maximum embedding capacity of Lena.

Table 2 lists the results for $L = 2, 3$ on the Mandrill image, which is also 512×512, 8 bpp, grayscale. Due to its irregular texture, Mandrill is more difficult to reversibly embed. The reported best result on its embedding capacity is about 50000 bits, which is easily surpassed by our method.

L	2	3
max capacity (bits)	231930	264359
bit rate (bpp)	0.8847	1.0084

Table 2. Maximum embedding capacity of Mandrill.

To embed a payload with a smaller size than the maximum embedding capacity, we reduce the size of EN1, until the targeted embedding capacity is met. For example, to embed a payload of 138856 bits in Lena, there are 116029 expandable $h \in$ EN at $L = 2$ with columnwise pairing. We assign 106635 of them in EN1, and the rest in EN2. The PSNR of the embedded image is 35.45 dB, which is higher than other methods with a payload of the same size.

Another feature of our method is the true fidelity at half resolution. As the integer average of each pair of pixel values are unchanged during embedding, the mean of each 2×2 block of the embedded image will be the same as the original image, except for rounding errors of integer averages. Thus at half resolution, the visual quality is imperceptible.

4. CONCLUSIONS

We present a high capacity reversible data embedding algorithm. As DE brings extra storage space, compression on original values of the embedding area is not needed. With compression (such as a linear prediction and entropy coding in [9]), the maximum embedding capacity will be even higher, with the expanse of complexity.

5. REFERENCES

[1] J. M. Barton, "Method and apparatus for embedding authentication information within digital data," *United States Patent, 5,646,997*, 1997.

[2] M. U. Celik, G. Sharma, A. M. Tekalp, and E. Saber, "Reversible data hiding," in *Proc. of ICIP*, Sept. 2002, vol. II, pp. 157–160.

[3] J. Fridrich, M. Goljan, and R. Du, "Lossless data embedding - new paradigm in digital watermarking," *EURASIP Journal on Applied Signal Processing*, vol. 2002, no. 2, pp. 185–196, Feb. 2002.

[4] C. W. Honsinger, P. W. Jones, M. Rabbani, and J. C. Stoffel, "Lossless recovery of an original image containing embedded data," *United States Patent, 6,278,791*, 2001.

[5] T. Kalker and F. M. J. Willems, "Capacity bounds and constructions for reversible data hiding," in *Proc. of the 14th International Conference on Digital Signal Processing*, July 2002, vol. 1, pp. 71–76.

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

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

[8] C. De Vleeschouwer, J. F. Delaigle, and B. Macq, "Circular interpretation of histogram for reversible watermarking," in *Proc. of IEEE 4th Workshop on Multimedia Signal Processing*, 2001.

[9] A. Said and W. A. Pearlman, "An image multiresolution representation for lossless and lossy compression," *IEEE Transactions on Image Processing*, vol. 5, no. 9, pp. 1303–1310, Sept. 1996.