Digital Signal Processing 21 (2011) 278–286

Contents lists available at ScienceDirect

Digital Signal Processing

www.elsevier.com/locate/dsp

Image self-embedding with high-quality restoration capability

Zhenxing Qian *, Guorui Feng, Xinpeng Zhang, Shuozhong Wang

School of Communication and Information Engineering, Shanghai University, Shanghai 200072, China

ARTICLE INFO

Article history:
Available online 6 May 2010

Keywords:
Data hiding
Self-embedding
Content restoration

ABSTRACT

In this paper, we present a new fragile watermarking method aimed at providing improved restoration capability. A mechanism of block classification is used to compress the original image according to the DCT coefficients, in which blocks corresponding to different types are encoded to variable lengths. The compressed bits are expanded as reference bits, and then embedded into the entire image along with some authentication bits generated from each block. On the receiving side, the authentication bits are extracted to localize the tampered areas, and the reference bits are used to reconstruct a reference image for restoring the contents of the tampered regions. Results show that the proposed method provides a better restoration quality.

1. Introduction

Many fragile watermarking schemes have been developed for digital images to detect accurately the areas that are modified with the counterfeit information [1,2]. In some data hiding methods referred to fragile watermarking, the watermark is referenced to the content of the original image, which is also described as "self-embedding" [3]. Information extracted from the intact regions can be used to recover the unauthenticated parts of the received image.

Fridrich and Goljan encode the DCT coefficients of each block to a fixed length, and then embed them into the LSB of a remote block. When tampered blocks are confirmed, bits extracted from the remote blocks can be used to recover these tampered regions [3]. Cheddad et al. propose a self-embedding algorithm in [4], in which down-sampled half-toning data of the original image are embedded into itself. On the receiving side, bits extracted from the reserved regions are used to reconstruct a low-quality image by inverse half-toning for recovering the tampered areas. Zhang and Wang propose a hierarchical fragile watermarking scheme [5], which can recover the tampered parts without any errors. However, the tampered regions must be less than 3.2% of the total area. In [6], rough retrieval of the original content in the tampered areas can be obtained by iterative projections of the polarity information on a convex set. As detection of tampered areas sometimes needs human assistance in [3], He et al. propose an adjacent-block based statistical detection method for detecting tampered areas automatically [7], based on the method of [3]. In [8], Zhang et al. propose a fragile watermarking scheme by data expansion which is capable of recovering the original principal content, even if 59% of the image area is tampered.

These methods have good ability to restore contents in the tampered regions. Their encoding algorithms for different blocks are basically the same and the bit-length of each block after compression is fixed, no matter the regions are smooth or rough. Certainly the fixed length is overmuch for smooth blocks; but is inadequate for a rough block, which needs more bits to express its content. Thus, these methods result in low-quality restoration to the tampered areas. Meanwhile, in some schemes, the authentication bits hidden in the cover are not enough to locate the tampered region.

In this paper, we propose a novel algorithm of self-embedding with multi-level encoding. Blocks are dynamically classified to several types, according to their degrees of smoothness. For the blocks corresponding to different types, we compress

* Corresponding author.
E-mail addresses: zxqian@shu.edu.cn (Z.X. Qian), grfeng@shu.edu.cn (G.R. Feng), xzhang@shu.edu.cn (X.P. Zhang), shuowang@shu.edu.cn (S.Z. Wang).

1051-2004/$ – see front matter
doi:10.1016/j.dsp.2010.04.006

1	2	6	7	15	16	28	29
3	5	8	14	17	27	30	43
4	9	13	18	26	31	42	44
10	12	19	25	32	41	45	54
11	20	24	33	40	46	53	55
21	23	34	39	47	52	56	61
22	35	38	48	51	57	60	62
36	37	49	50	58	59	63	64

Fig. 1. Zigzag scanning order.

83	2	−1	0	0	0	0	0
1	1	−1	0	0	0	0	0
−1	0	0	0	0	0	0	0
−1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Fig. 2. A block of quantized coefficients.

them into different number of bits so that rougher blocks have more bits and smoother blocks have fewer bits after compression. On average, each block is encoded into 64 bits, expanded to 160 reference bits for error correction [8]. The reference bits are embedded into the 3 LSB layers of the cover with the authentication bits exploited from the original content. On the receiving side, tampered areas can be detected accurately and recovered with high quality.

2. Self-embedding procedure

In the procedure of self-embedding, the original image is compressed into a number of bits by multi-level encoding, and authentication bits are generated from each block. The reference bits and the authentication bits are embedded into the cover, by substituting the 3 least significant bit-planes.

When compressing an original image, we define an algorithm of multi-level encoding, in which blocks are dynamically classified into several types. For different types, the corresponding blocks are encoded to different number of bits. Finally, each block is compressed to 64 bits that is 1 bit per pixel (*bpp*) on average.

2.1. Block classification

Given a cover image sized $N_1 \times N_2$, the number of pixels is N ($N = N_1 \times N_2$). Divide the image into $r = N/64$ non-overlapping blocks with the same size of 8×8, assuming both N_1 and N_2 are multiples of 8. Transform each block by DCT, and then quantize the coefficients with the typical JPEG quantization table corresponding to the compression factor (Q) of 50.

Zigzag scanning the 8×8 coefficient matrix results in a vector that always contains consecutive zeros in the tail. Fig. 1 shows the order of zigzag scanning. A block of quantized coefficients shown in Fig. 2 are reordered to $[83, 2, 1, -1, -1, -1, 0, -1, 0, -1, 0, 0, 0, \ldots]$. Denote the coefficient vector as $[c_1, c_2, \ldots, c_{64}]$, and the index of the last non-zero coefficient in the vector as i_{LNC}. In this example, $i_{\mathrm{NZC}} = 10$. If all coefficients are zero, $i_{\mathrm{LNC}} = 0$. Then, there would be 65 choices for i_{LNC} ($i_{\mathrm{LNC}} = 0, \ldots, 64$).

This way, for the original image, we obtain an index matrix with the size of $(N_1/8) \times (N_2/8)$, in which each element is the index of the last non-zero coefficient corresponding to the 8×8 block from the image. Calculate the histogram of the index matrix,

$$p_k = 64 \cdot n_k / N, \quad k = 0, \ldots, 64 \tag{1}$$

where n_k is the number of blocks in which i_{LNC} equals k. For example, the histogram of the index matrix corresponding to an image (sized 512×512 as in Fig. 3) is shown in Fig. 4.

According to the value of i_{LNC} of each block, we sort the 8×8 blocks in an ascending order, denote which as $\{\mathbf{B}_1, \mathbf{B}_2, \ldots, \mathbf{B}_r\}$. Divide the indexes ($[0, 64]$) into 6 parts, $\{[0, 3], [4, 6], [7, 9], [10, 12], [13, 16], [17, 64]\}$, denoted as $[L_i, U_i]$, $(i = 1, \ldots, 6)$. Accordingly, blocks belonging to these parts are classified to 6 types of sets, $\{\{\mathbf{B}_1, \mathbf{B}_2, \ldots, \mathbf{B}_{p1}\}; \{\mathbf{B}_{p1+1}, \mathbf{B}_{p1+2}, \ldots, \mathbf{B}_{p2}\}; \ldots, \{\mathbf{B}_{p5+1}, \mathbf{B}_{p5+2}, \ldots, \mathbf{B}_r\}\}$.

For each part, calculate the sum of the histogram values falling into the range,

$$S_i = \sum_{j=L_i}^{U_i} p_j, \quad i = 1, \ldots, 6 \tag{2}$$

Fig. 3. A gray image.

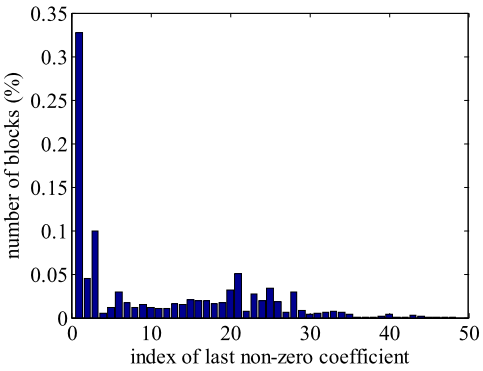

Fig. 4. Histogram of the index matrix of an image.

Table 1
The algorithm of block classification.

If $TH > 0.5$,	
a.	If $S_1 > 0.5$, $\{S'_1, S'_2, \ldots, S'_6\} = \{0.5, 0, 0, 0, 0, 0.5\}$, classify the corresponding blocks to $\{\{\mathbf{B}_1, \mathbf{B}_2, \ldots, \mathbf{B}_{r/2}\}; \{\times\}; \{\times\}; \{\times\}; \{\times\}; \{\mathbf{B}_{r/2+1}, \mathbf{B}_{r/2+2}, \ldots, \mathbf{B}_r\}\}$. Here, $\{\times\}$ means that no block belongs to the corresponding type. Otherwise go to step *a*.
b.	If $S_1 + S_2 > 0.5$, $\{S'_1, S'_2, \ldots, S'_6\} = \{S_1, 0.5\text{-}S_1, 0, 0, 0.5\text{-}S_1, S_1\}$, classify the corresponding blocks to $\{\{\mathbf{B}_1, \mathbf{B}_2, \ldots, \mathbf{B}_{p1}\}; \{\mathbf{B}_{p1+1}, \mathbf{B}_{p1+2}, \ldots, \mathbf{B}_{r/2}\}; \{\times\}; \{\times\}; \{\mathbf{B}_{r/2+1}, \mathbf{B}_{r/2+2}, \ldots, \mathbf{B}_{r-p1}\}; \{\mathbf{B}_{r-p1+1}, \mathbf{B}_{r-p1+2}, \ldots, \mathbf{B}_r\}\}$. Otherwise go to step *b*.
c.	$\{S'_1, S'_2, \ldots, S'_6\} = \{S_1, S_2, 0.5\text{-}S_1\text{-}S_2, 0.5\text{-}S_1\text{-}S_2, S_2, S_1\}$, classify the corresponding blocks to $\{\{\mathbf{B}_1, \mathbf{B}_2, \ldots, \mathbf{B}_{p1}\}; \{\mathbf{B}_{p1+1}, \mathbf{B}_{p1+2}, \ldots, \mathbf{B}_{p2}\}; \{\mathbf{B}_{p2+1}, \mathbf{B}_{p2+2}, \ldots, \mathbf{B}_{r/2}\}; \{\mathbf{B}_{r/2+1}, \mathbf{B}_{r/2+2}, \ldots, \mathbf{B}_{r-p2}\}; \{\mathbf{B}_{r-p2+1}, \mathbf{B}_{r-p2+2}, \ldots, \mathbf{B}_{r-p1}\}; \{\mathbf{B}_{r-p1+1}, \mathbf{B}_{r-p1+2}, \ldots, \mathbf{B}_r\}\}$.
Else	
d.	If $S_6 > 0.5$, $\{S'_1, S'_2, \ldots, S'_6\} = \{0.5, 0, 0, 0, 0, 0.5\}$, classify the corresponding blocks to $\{\{\mathbf{B}_1, \mathbf{B}_2, \ldots, \mathbf{B}_{r/2}\}; \{\times\}; \{\times\}; \{\times\}; \{\times\}; \{\mathbf{B}_{r/2+1}, \mathbf{B}_{r/2+2}, \ldots, \mathbf{B}_r\}\}$. Otherwise go to step *e*.
e.	If $S_6 + S_5 > 0.5$, $\{S'_1, S'_2, \ldots, S'_6\} = \{S_6, 0.5\text{-}S_6, 0, 0, 0.5\text{-}S_6, S_6\}$, classify the corresponding blocks to $\{\{\mathbf{B}_1, \mathbf{B}_2, \ldots, \mathbf{B}_{r-p5}\}; \{\mathbf{B}_{r-p5+1}, \mathbf{B}_{r-p5+2}, \ldots, \mathbf{B}_{r/2}\}; \{\times\}; \{\times\}; \{\mathbf{B}_{r/2+1}, \mathbf{B}_{r/2+2}, \ldots, \mathbf{B}_{p5}\}; \{\mathbf{B}_{p5+1}, \mathbf{B}_{p5+2}, \ldots, \mathbf{B}_r\}\}$. Otherwise go to step *f*.
f.	$\{S'_1, S'_2, \ldots, S'_6\} = \{S_6, S_5, 0.5\text{-}S_6\text{-}S_5, 0.5\text{-}S_6\text{-}S_5, S_5, S_6\}$, classify the corresponding blocks to $\{\{\mathbf{B}_1, \mathbf{B}_2, \ldots, \mathbf{B}_{r-p5}\}; \{\mathbf{B}_{r-p5+1}, \mathbf{B}_{r-p5+2}, \ldots, \mathbf{B}_{r-p4}\}; \{\mathbf{B}_{r-p4+1}, \mathbf{B}_{r-p4+2}, \ldots, \mathbf{B}_{r/2}\}; \{\mathbf{B}_{r/2+1}, \mathbf{B}_{r/2+2}, \ldots, \mathbf{B}_{p4}\}; \{\mathbf{B}_{p4+1}, \mathbf{B}_{p4+2}, \ldots, \mathbf{B}_{p5}\}; \{\mathbf{B}_{p5+1}, \mathbf{B}_{p5+2}, \ldots, \mathbf{B}_r\}\}$.

where p is the histogram value calculated in Eq. (1). For example, the set $\{S_1, S_2, \ldots, S_6\}$ of the image mentioned above is $\{0.472, 0.047, 0.044, 0.032, 0.072, 0.333\}$, and the total number of blocks corresponding to each part is $\{1933, 193, 180, 133, 294, 1363\}$.

Then we calculate the sum of S_1, S_2 and S_3,

$$TH = \sum_{i=1}^{3} S_i \tag{3}$$

With the value of TH, we adjust $\{S_1, S_2, \ldots, S_6\}$ to a new set $\{S'_1, S'_2, \ldots, S'_6\}$. Accordingly, the blocks are re-classified. The algorithm is shown in Table 1.

After the above operations, 6 types of blocks are re-classified, corresponding to $\{S'_1, S'_2, \ldots, S'_6\}$, where $S'_1 = S'_6$, $S'_2 = S'_4$, and $S'_3 = S'_5$. For example, the set $\{S'_1, S'_2, \ldots, S'_6\}$ of the image in Fig. 3 is adjusted to $\{0.472, 0.028, 0, 0, 0.028, 0.472\}$, and the blocks corresponding to these types are $\{\{\mathbf{B}_1, \mathbf{B}_2, \ldots, \mathbf{B}_{1933}\}; \{\mathbf{B}_{1934}, \ldots, \mathbf{B}_{2048}\}; \{\times\}; \{\times\}; \{\mathbf{B}_{2049}, , \ldots, \mathbf{B}_{2163}\}; \{\mathbf{B}_{2164}, \ldots, \mathbf{B}_{4096}\}\}$.

2.2. Data compression

For different types of blocks, we define 6 different tables to assign numbers of bits for DC and AC coefficients, as shown in Fig. 5, where DC coefficient is converted to unsigned binary number, and AC coefficients are converted to signed binary number. Meanwhile, in order to separate the type of a block, a header is added to the head of the encoded bits of the coefficients, as shown in Fig. 6. The headers corresponding to $\{S'_1, S'_2, \ldots, S'_6\}$ are $\{001, 010, 011, 100, 101, 110\}$.

Table 2 shows the number of coefficients (DC & AC) chosen for encoding and the total number of bits after encoding for each kind of blocks. For example, as the block shown in Fig. 2, if it corresponds to S'_4, the encoding data are (100 01010011 0000010 0000001 100000 000001 100000 00000 10000 00000 10000 0000 000); the number of coefficients to be encoded is 12, the encoded length for the block is 70.

Assuming the encoding length for a block corresponding to the type S'_i is R_i. According to Table 1, the original image is compressed into N bits, as calculated in the following equation:

8	8	0	0	0	0	0	0
8	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

(a)

8	7	6	0	0	0	0	0
7	6	0	0	0	0	0	0
6	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

(b)

8	7	6	5	0	0	0	0
7	6	5	0	0	0	0	0
6	5	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

(c)

8	7	6	5	0	0	0	0
7	6	5	0	0	0	0	0
6	5	0	0	0	0	0	0
5	3	0	0	0	0	0	0
4	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

(d)

8	7	6	5	4	2	0	0
7	6	5	4	0	0	0	0
6	5	4	0	0	0	0	0
5	4	0	0	0	0	0	0
4	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

(e)

8	7	6	5	4	3	0	0
7	6	5	4	3	0	0	0
6	5	4	3	0	0	0	0
5	4	3	0	0	0	0	0
4	3	0	0	0	0	0	0
3	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

(f)

Fig. 5. Tables for assigning the length of coefficients for blocks corresponding to different types: (a) is the length table for the blocks corresponding to S'_1, (b) is for S'_2, (c) is for S'_3, (d) is for S'_4, (e) is for S'_5, and (f) is for S'_6.

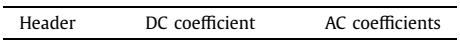

Fig. 6. The structure of encoded bits for each block.

Table 2
Encoding information for different type of blocks.

Types	Number of coefficients chosen for encoding	Encoding length for each block
S'_1	3	27
S'_2	6	43
S'_3	9	58
S'_4	12	70
S'_5	16	85
S'_6	22	101

$$\sum_{i=1}^{6} R_i \cdot S'_i \cdot r = \sum_{i=1}^{3} (R_i + R_{7-i}) \cdot S'_i \cdot r = 128 \cdot N/64 \cdot \sum_{i=1}^{3} S'_i = N \tag{4}$$

2.3. Data hiding

As the original image is compressed to N bits, pseudo-randomly divide these bits into $N/64$ subsets with a secret key. Denote these subsets as $\{\mathbf{M}_1, \mathbf{M}_2, \ldots, \mathbf{M}_{N/64}\}$, each of which has 64 bits. From the secret key, derive a pseudo-random binary matrix $\mathbf{F}$ sized 160×64 used to generate 160 reference-bits. The calculation is shown in Eq. (5), and the arithmetic is modulo-2,

$$\mathbf{P}_i^T = \mathbf{F} \cdot \mathbf{M}_i^T, \quad i = 1, 2, \ldots, N/64 \tag{5}$$

where "T" is the transpose operator. Details of the data expansion method are described in [8], which brings redundancy to the initial data for error correction.

Thus, we have $5N/2$ reference bits for the original image. Pseudo-randomly permute these bits using the secret key, and divide them into $N/64$ groups with 160 bits per group. Then each 8×8 block in the original image is mapped to a group in order.

For each 8×8 block, collect 320 bits from the 5 MSB layers of the original content. Combine these bits with the 160 bits of the corresponding group, and feed them into a hash function to generate 32 bits for authentication.

Substitute the 3 LSB bits of each 8×8 block with the 160 reference-bits and 32 authentication-bits. Thus the self-embedding is completed.

3. Content restoration

On the receiving side, the hidden bits are extracted from the image, including the authentication-bits and reference-bits, assuming the size of the image is not altered. In the following, we identify the tampered blocks using the authentication-bits, and reconstruct a reference image for restoration.

Before the content restoration, we need to find whether the blocks are tampered or not. For each 8×8 block, 192 bits are extracted from the 3 LSB layers. Among the 192 bits extracted from each block, 32 authentication-bits and 160 reference-bits can be separated according to the secret key. Collect 320 bits from the 5 MSB layers, combine them with the extracted 160 reference-bits, and feed into the hash function to generate 32 bits. If the calculated 32 bits are different from the extracted authentication-bits, the block is tampered. Otherwise, the block is not tampered.

Totally $5N/2$ reference-bits are extracted from the received image. Reorder these bits using the secret key, and divide the bits into $N/64$ subsets. As the tampered blocks are determined, only part of bits is useful in each subset. For the k-th subset, denote the number of useful bits as v, and the subset becomes $[p_{k,e(1)}, p_{k,e(2)}, \ldots, p_{k,e(v)}]$. $\mathbf{F}^{(k)}$ is the matrix consisting of the rows in $\mathbf{F}$ corresponding to $e(i)$, where $i \in [1, v]$. According to (5), solve the following equation:

$$[p_{k,e(1)}, p_{k,e(2)}, \ldots, p_{k,e(v)}]^T = \mathbf{F}^{(k)} \cdot [m_{k,1}, m_{k,2}, \ldots, m_{k,64}]^T, \quad k = 1, 2, \ldots, N/64 \tag{6}$$

Considering the percent of tampered blocks as α, as described in [8], the possibility of $[m_{k,1}, m_{k,2}, \ldots, m_{k,64}]$ having a unique solution is

$$P_{\mathbf{B}} = \sum_{i=0}^{160} \left\{ \binom{160}{i} \cdot (1-\alpha)^i \cdot \alpha^{160-i} \cdot \prod_{j=1}^{64} \left(1 - \frac{2^{j-1}}{2^i}\right) \right\} \tag{7}$$

Since there are totally $N/64$ subsets, we can recover the original reference-bits with the possibility

$$P_{\mathbf{I}} = P_{\mathbf{B}}^{N/64} \tag{8}$$

$P_{\mathbf{I}}$ depends on α and N. The smaller the values of α and N, the higher the probability $P_{\mathbf{I}}$ will be. For example, in an image sized 512×512, when the tampering rate α is no larger than 0.35, $P_{\mathbf{I}}$ equals 1; that is to say, a complete image can be reconstructed with the extracted bits if the tampering rate is smaller than 35% of the total size of the image.

After solving the equation, the reference-bits are retrieved. According to the header of each segment, we can determine the length of the coefficients, and determine the part number this segment belongs to according Table 1. This way, we can derive the string of the encoded bits of coefficients for the tampered blocks. Then, for each string, calculate DCT coefficients using the corresponding tables for assigning the length of coefficients in Fig. 5. After de-quantization, inverse DCT and rounding operation, we can restore the tampered blocks.

4. Experimental results

The proposed method is implemented in the experiment using gray images sized 512×512 as test covers. In Fig. 7, the image "Ruler" is used to test the compressing algorithm described in Section 2 and compare to the algorithms in [7] and [8]. The compression rate of the algorithm in [8] is 0.84 bits per pixel (*bpp*), and the reconstructed image is shown in Fig. 7(b) with PSNR = 13.9 dB. The method in [7] compressed the image with the compression rate of 1 *bpp*, and PSNR of the reconstructed image equals 17.6 dB, which is shown in Fig. 7(d). In the proposed method, the compression rate is equal to that in [7], and PSNR of the reconstructed image is 21.3 dB, shown in Fig. 7(f). Figs. 7(c), 7(e) and 7(g) provide actual-sized contents in the rectangular areas in the figures (b), (d) and (f), respectively.

All blocks of an image are assigned with a fixed length in [7] and [8], which is overmuch for smooth blocks while inadequate for rough blocks. In the proposed method, we use different bits for different types of blocks, more bits for the rough blocks and fewer bits for smooth blocks. The proposed assignment provides a better reconstruction quality, which is assured in this experiment.

Two images "Portofino" and "Baboon" are used as host image for self-embedding, which are shown in Fig. 8. Fig. 9 shows the watermarked image of both images, PSNR of which are 37.7 dB and 37.9 dB, respectively. We modify the watermarked image by replacing the real content with fake information by Photoshop, shown in Fig. 10. Fig. 11 gives the detection result of the tampered blocks, in which the blocks in black are "tampered blocks", and the tampering rates are respectively 16.6% and 18.8%. As the tampered regions are smaller than 35% of the whole size, complete images can be reconstructed for reference with the extracted bits. The reconstructed images are shown in Fig. 12, PSNRs due to which are 34.4 dB and 25.1 dB. Finally, the tampered images are recovered in Fig. 13. Experiments show that the recovered images have good recovery quality.

Table 3 compares several fragile watermarking schemes with the restoration capability on the same image "Lena". The method in [5] has the best quality of reconstructed image; but the tampered regions must be less than 3.2%. PSNR of the

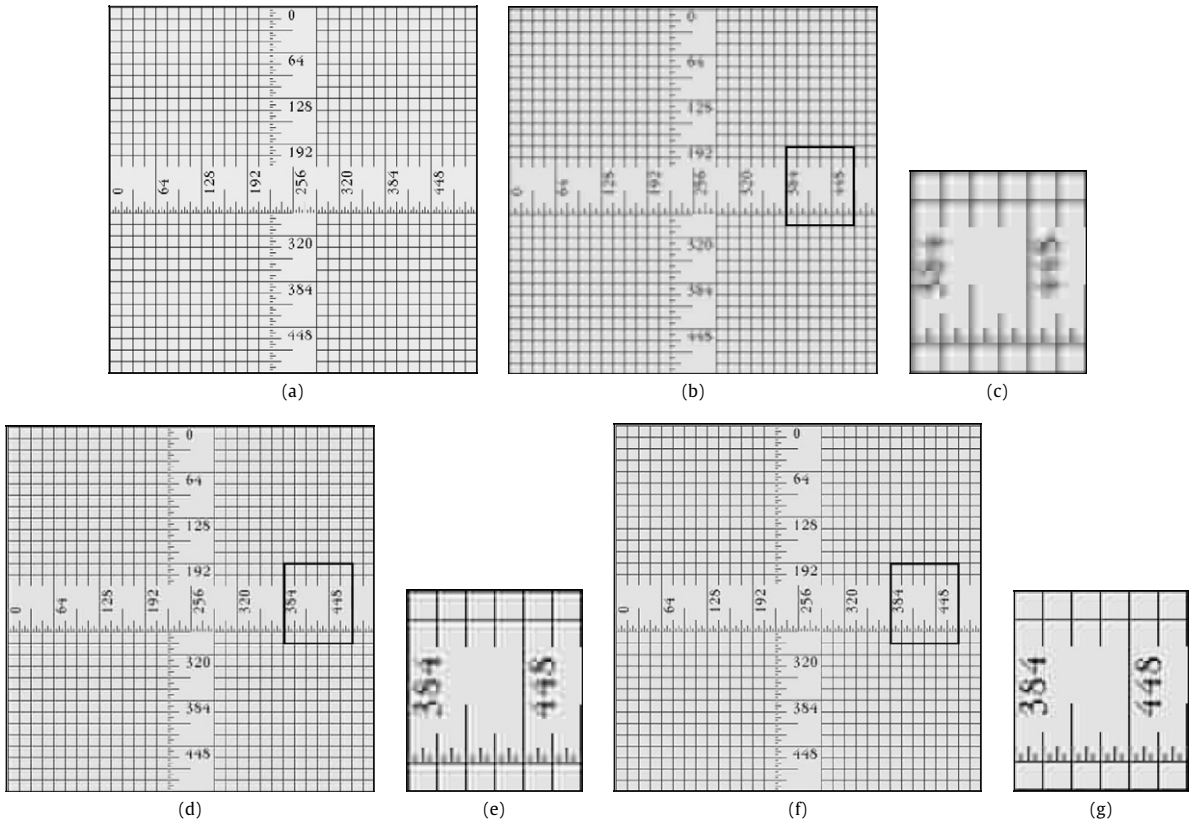

Fig. 7. Simulation results: (a) The original image; (b) Reconstructed image using the compression algorithm in [8] with PSNR = 13.9 dB; (c) The true size content in the rectangular area of (b); (d) Reconstructed image using the compression algorithm in [7] with PSNR = 17.6 dB; (e) The true size content in the rectangular area of (d); (f) Reconstructed image using the proposed algorithm with PSNR = 21.3 dB; (g) The true size content in the rectangular area of (f).

Fig. 8. Original images Portofino and Baboon.

reconstructed image is better than that in [7], with the same compression rate (1 *bpp*) for embedding, although PSNR of the watermarked image is smaller. The method in [7] can identify the tampered blocks with a probability more than 98% even the tampered area is up to 70% of the watermarked image. However, information of one block is embedded into another block in [7], restoration cannot be executed if both blocks are tampered. More comparisons of the restoration quality are shown in Table 4. In terms of PSNR of the restored areas, the proposed method can recover the tampered areas with the higher quality than the methods in [7] and [8].

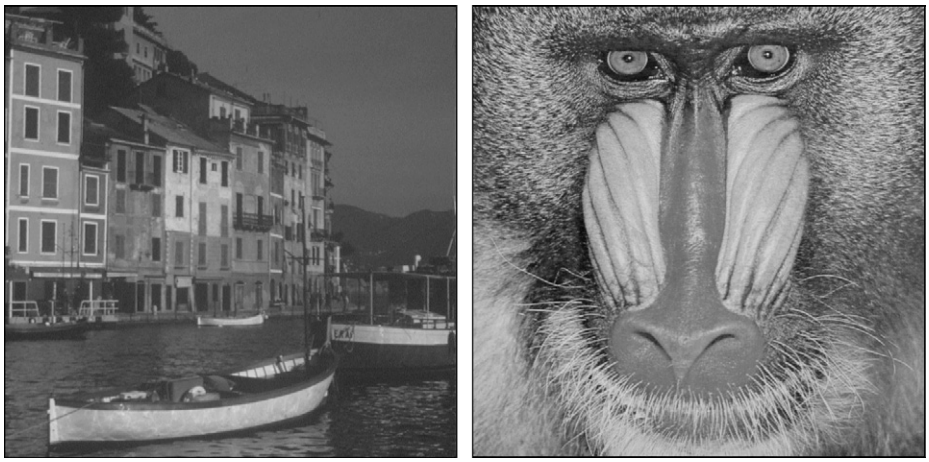

Fig. 9. Watermarked images of Portofino and Baboon.

Fig. 10. Tampered version of Portofino and Baboon.

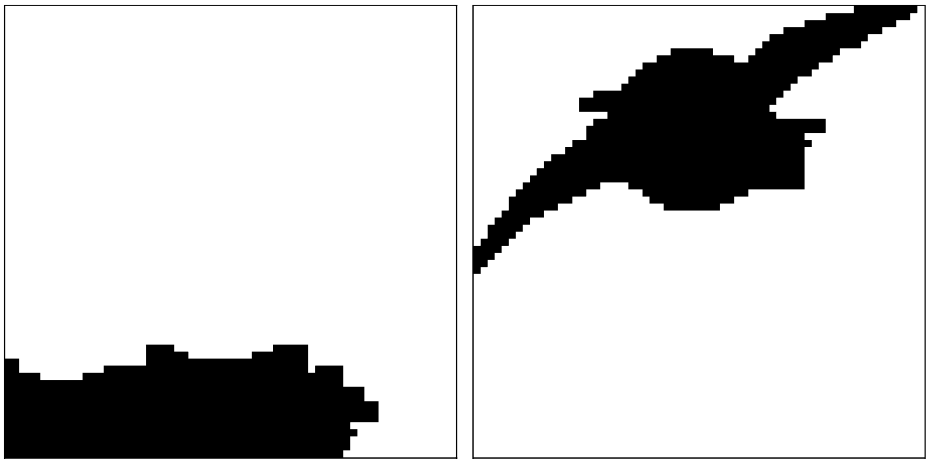

Fig. 11. Results of tampering detection.

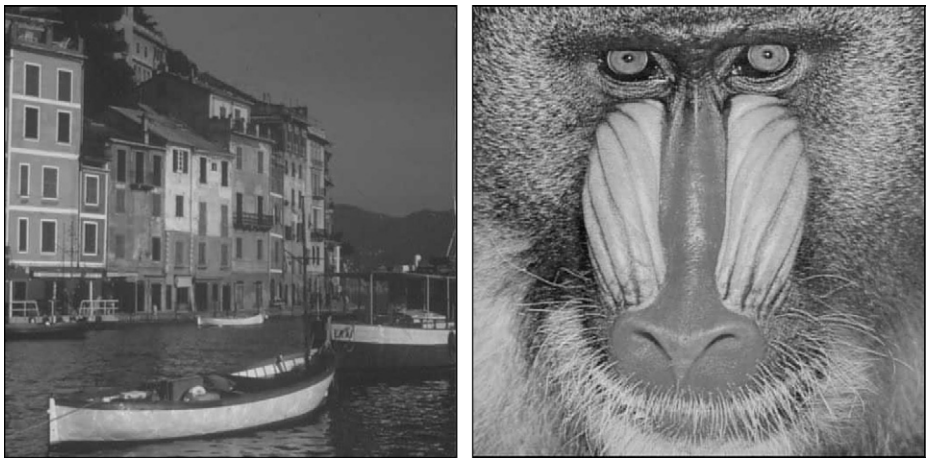

Fig. 12. Reconstructed images with bits extracted from Fig. 10.

Fig. 13. Restored images of Portofino and Baboon.

Table 3
Comparison of restoration capability among different schemes.

Methods	PSNR of the watermarked images (dB)	PSNR of the reconstructed image (dB)	Condition of restoration
Method in [5]	28.7	$+\infty$	$<3.2\%$
Method in [7]	51.1	32.2	Regions storing the original information of tampered areas must be reserved.
Method in [8]	37.9	29.9	$<59\%$
Proposed method	37.9	35.0	$<35\%$

Table 4
Comparison of restoration quality among different methods.

Images	PSNR in the restored areas		
	Method of [8]	Method of [7]	Proposed method
Lena	29.9 dB	32.2 dB	35.0 dB
Pepper	28.8 dB	31.7 dB	34.0 dB
F16	27.2 dB	30.9 dB	34.9 dB
Sailboat	26.2 dB	28.8 dB	31.5 dB
Goldhill	29.8 dB	30.7 dB	32.6 dB
Couple	27.1 dB	29.3 dB	31.8 dB
Baboon	22.2 dB	22.9 dB	25.1 dB
Barbara	24.2 dB	24.8 dB	26.0 dB
Zelda	36.6 dB	37.3 dB	38.5 dB

5. Conclusions

This paper proposes a novel image self-embedding method with high-quality content restoration capability. We dynamically classify the blocks into different types, according to their degrees of smoothness. The corresponding blocks are encoded into different numbers of bits for reference with rougher blocks having more bits and smooth blocks having fewer bits. We then hide the authentication-bits and reference-bits into the 3 LSB layers of the image.

On the receiving side, after locating the tampered block with authentication-bits, the reference-bits are extracted from the un-tampered blocks to provide sufficient information for restoration of the principal content in the host image. Because we adaptively assign variable length for different kinds of blocks, the reconstructed image has a better quality than that of some previous methods which use the fixed length for all blocks.

The proposed method aims at providing a fragile watermarking method, any changes to the watermarked image are not allowed. As a result, this method has no ability to resist the noise or compression attacks. We are trying to do some investigation on semi-fragile watermarking for image recovery in the future.

Acknowledgments

This work was supported by the Natural Science Foundation of China (60872116, 60832010, and 60773079), and the High-Tech Research and Development Program of China (2007AA01Z477).

References

[1] P.W. Wong, N. Memon, Secret and public key image watermarking schemes for image authentication and ownership verification, IEEE Trans. Image Process. 10 (2001) 1593–1601.
[2] S. Suthaharan, Fragile image watermarking using a gradient image for improved localization and security, Pattern Recognition Lett. 25 (2004) 1893–1903.
[3] J. Fridrich, M. Goljan, Protection of digital images using self embedding, in: Proceedings of Symposium on Content Security and Data Hiding in Digital Media, 1999.
[4] A. Cheddad, et al., A secure and improved self-embedding algorithm to combat digital document forgery, Signal Process. 89 (12) (2009) 2324–2332.
[5] X. Zhang, S. Wang, Fragile watermarking with error-free restoration capability, IEEE Trans. Multimedia 10 (8) (2008) 1490–1499.
[6] X. Zhu, A. Ho, P. Marziliano, A new semi-fragile image watermarking with robust tampering restoration using irregular sampling, Signal Process. Image Commun. 22 (5) (2007) 515–528.
[7] H. He, et al., Adjacent-block based statistical detection method for self-embedding watermarking techniques, Signal Process. 89 (8) (2009) 1557–1566.
[8] X. Zhang, S. Wang, G. Feng, Fragile watermarking scheme with extensive content, in: IWDW 2009, in: LNCS, vol. 5703, 2009, pp. 268–278.

Zhenxing Qian received the B.S. degree in 2003 and the Ph.D. degree in 2007 from University of Science & Technology of China (USTC). Since 2009, he has been with the faculty of the School of Communication and Information Engineering, Shanghai University. His research interests include data hiding, image processing, and digital forensics.

Guorui Feng received the B.S. and M.S. degrees in computational mathematics from Jilin University, Changchun, P.R. China, in 1998 and 2001, respectively, and Ph.D. degree in Electronic Engineering from Shanghai Jiaotong University, Shanghai, P.R. China in 2005. From March 2006 to December 2006, he was an assistant professor in East China Normal University. From January 2007 to December 2007, he was a Research Fellow in Nanyang Technological University. From July 2008 till now, he was been working as an assistant professor in Shanghai University, P.R. China. His research interests are in the areas of signal processing and computational intelligence.

Xinpeng Zhang received the B.S. degree in computational mathematics from Jilin University, China, in 1995, and the M.E. and Ph.D. degrees in communication and information system from Shanghai University, China, in 2001 and 2004, respectively. Since 2004, he has been with the faculty of the School of Communication and Information Engineering, Shanghai University, where he is currently a Professor. His research interests include information hiding, image processing and digital forensics. He has published more than 100 papers in these areas.

Shuozhong Wang received the B.S. degree from Peking University, Beijing, China, in 1966 and the Ph.D. degree from University of Birmingham, Birmingham, England, in 1982. Currently, he is a Professor of Shanghai University. His research interests include image processing, audio processing, and information hiding.