JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 27, 577-593 (2011)

Reversible Data Hiding Based on Median Difference Histogram

HSIEN-WEI YANG[1,2], I-EN LIAO[*] AND CHAUR-CHIN CHEN[+]
[1]*Department of Information Management*
Overseas Chinese University
Taichung, 407 Taiwan
[2]*Department of Applied Mathematics*
[*]*Department of Computer Science and Engineering*
National Chung Hsing University
Taichung, 402 Taiwan
[+]*Department of Computer Science*
National Tsing Hua University
Hsinchu, 300 Taiwan

This work proposes a reversible data hiding algorithm that is based on the median difference histogram. The method divides the cover image into non-overlapping identical blocks. In each block, the median pixel is selected to calculate absolute differences between the median pixel value and the values of the other pixels. Then, these differences are used to generate a histogram and the histogram shifting method is adopted to embed data. The method can reduce the values of differences and increase the maximum frequency of the histogram. Therefore, the hiding capacity can be increased. Experimental results are presented to prove the validity of the proposed algorithm.

Keywords: reversible data hiding, histogram shifting, median difference, difference expansion, difference histogram

1. INTRODUCTION

Reversible data hiding [1, 2], also known as lossless data hiding, enables marked media to be restored to their original form without any distortion. This technique is applied in such fields as content authentication of multimedia data, law enforcement, medical imagery and astronomical research. Various reversible data hiding methods have been proposed for grayscale images, and these can be divided into the following categories:

(1) Data compression

Fridrich *et al.* [3, 4] compressed the least significant bit (LSB) plane to obtain additional space for embedding secret data. Celik *et al.* [5-7] improved the method of Fridrich *et al.* and proposed the Generalized-LSB (G-LSB) scheme by compressing the quantization residuals of pixels to yield additional space to embed a message. Awrangjeb and Kankanhalli [8, 9] presented a scheme that detects the textured blocks; extracts the LSBs of the pixel-values from these textured blocks based on the Human Visual System (HVS), and concatenates the authentication information with the compressed bit-string.

Received April 15, 2009; revised April 27, 2009; accepted May 21, 2009.
Communicated by H. Y. Mark Liao.

(2) Difference expansion

Tian [10] presented a method that expands the difference between two neighboring pixels to obtain redundant space for embedding a message. Alattar [11] used the difference expansion of vectors of adjacent pixels to obtain more embedding space. Using Tian's scheme of difference expansion, Chang and Lu [12] calculated the difference between a pixel and the mean value of its neighboring pixels to embed a message. Weng *et al.* [13, 14] used the correlations among four pixels in a quad, and embedded data by expanding the differences between one pixel and each of its three neighboring pixels.

(3) Histogram shifting

Ni *et al.* [15] used the histogram of the pixel values in the cover image to embed secret data into the maximum frequency pixels. Fallahpour and Sedaaghi [16] divided a cover image into several blocks, and embedded secret data into the histogram of each block. Lin and Hsueh [17] applied the histogram shifting of Ni *et al.* to the pixel differences, which were obtained from the absolute differences between pairs of neighboring pixels.

(4) Integer wavelet transform

Xuan *et al.* [18-20] proposed a lossless data hiding technique based on integer wavelet transform, which embeds high capacity data into the most insensitive bit-planes of wavelet coefficients. Yang *et al.* [21] presented a symmetrical histogram expansion scheme in the transform domain of Piecewise-Linear Haar (PLHaar). Data are embedded into the PLHaar coefficients of the images from the pivotal bin of a histogram of PLHaar coefficients symmetrically toward both sides of the histogram.

This study proposes a data hiding method that is based on a median difference histogram. This method uses the similarity of the pixel values of the adjacent pixels, and uses the absolute differences among neighboring pixels in a block to generate a difference histogram. The median pixel in the block is used to calculate the absolute differences to reduce the values of differences, and the optimal block size is determined to increase the maximum frequency in the histogram (where "frequency" is the number of instances of a difference value.) In a histogram shifting data hiding method, the maximum frequency in the histogram yields the embedding capacity. Experimental results reveal that the proposed method has a higher embedding capacity than other tested methods.

The rest of this paper is organized as follows. Section 2 describes the proposed scheme. Section 3 summarizes the experimental results. Section 4 draws conclusions.

2. PROPOSED SCHEME

In the proposed scheme, the cover image is divided into non-overlapping identical $m \times n$-pixel blocks. Each block has a width of m pixels and a height of n pixels, and the proposed scheme finds the optimal m and n that maximize the embedding capacity. In each block, a pixel is selected as the base pixel, and an $m \times n$-pixel block contains $m \times n - 1$ absolute differences – which are those between the base pixel and the other $m \times n - 1$ pixels. The calculated difference when the fixed site pixel in every block is designated the base pixel is called the fixed site difference. The calculated difference when the median pixel in each block is chosen as the base pixel is called the median difference. Fig. 1 dis-

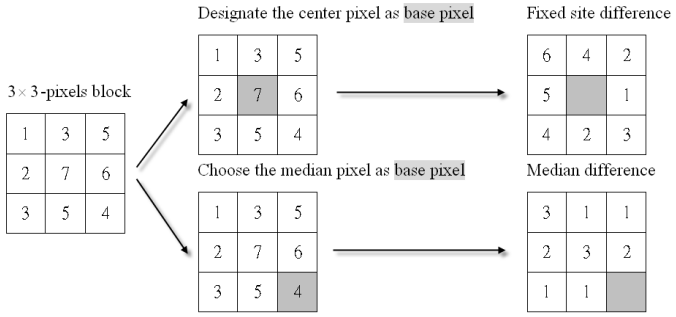


Fig. 1. Example of fixed site difference and median difference for a 3 × 3-pixel block.

plays an example of fixed site difference and median difference for a 3 × 3-pixel block.

The proposed scheme uses the median difference to generate histogram, and the histogram shifting method [15] to embed data. The embedding process, extraction process, and overhead information in the proposed scheme are described below.

2.1 Embedding Process

Fig. 2 displays the embedding process in the proposed scheme. The secret data are embedded into the cover image (called *CI*) by repeatedly performing the embedding process until the secret data have all been fully embedded; finally, the stego image and key are output. The key contains the information that must be used by the receiver to extract the secret data. The embedding process has three steps, which are as follows.

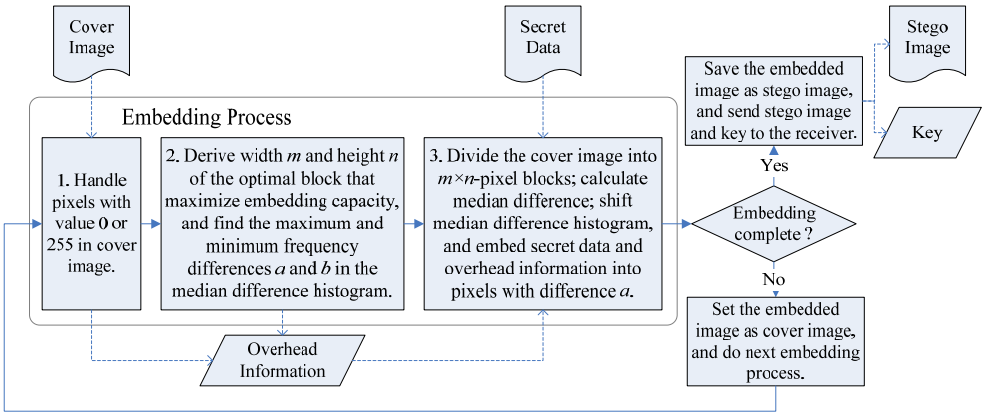


Fig. 2. The embedding process of the proposed scheme.

Step 1: Handle pixels with value 0 or 255 in cover image.

In the embedding process, pixels with a value of 0 or 255, which are called Lower-Upper-Bound (*LUB*) pixels, may cause underflow or overflow in the decrement or increment operations. To avoid these problems, the values of pixels with a value of 0 and 255 are changed to 1 and 254, respectively, before the cover image is divided into blocks. The

modified *CI* is called *CI′*. To recover the LUB pixels in the extraction process, the information about the *LUB* pixels (called *LUBI*) is recorded as overhead information. The *LUBI* includes *F*, *B*, *F′* and *B′*. *F* and *F′* are flag bits that represent whether *CI* includes pixels with values of 0 and 255, respectively. *B* is a bit vector that specifies whether the pixels with values 0 and 1 in *CI* are changed in *CI′*. A value of 0 in *B* means no change, and a value of 1 in *B* means change in the pixel value. Similarly, *B′* is a bit vector that represents whether the pixels with values 254 and 255 in *CI* are changed in *CI′*. A value of 0 in *B′* means no change, and a value of 1 in *B′* means change in the pixel value. Let $h(x)$ denote the number of pixels with gray value x in *CI*, where $0 \leqq x \leqq 255$. *F* and *F′* are defined as follows;

$$F = \begin{cases} 0 & \text{if } h(0) = 0, \\ 1 & \text{if } h(0) > 0. \end{cases}$$

$$F' = \begin{cases} 0 & \text{if } h(255) = 0, \\ 1 & \text{if } h(255) > 0. \end{cases}$$

Let *LBP* be the set of pixels with value 1 in *CI′*; *LBP*(*k*) denote the *k*th pixel in *LBP*; and *B*(*k*) denote the *k*th bit in *B*, where $1 \leqq k \leqq h(0) + h(1)$. Let *UBP* denote the pixels with value 254 in *CI′*; *UBP*(*k′*) denote the *k′*th pixel in *UBP*; and *B′*(*k′*) denote the *k′*th bit in *B′*, where $1 \leqq k' \leqq h(255) + h(254)$. *B*(*k*) and *B′*(*k′*) are specified as

$$B(k) = \begin{cases} 0 & \text{if the value of } LBP(k) \text{ is 1 in } CI, \\ 1 & \text{if the value of } LBP(k) \text{ is 0 in } CI. \end{cases}$$

$$B'(k') = \begin{cases} 0 & \text{if the value of } UBP(k') \text{ is 254 in } CI, \\ 1 & \text{if the value of } UBP(k') \text{ is 255 in } CI. \end{cases}$$

Fig. 3 displays an example of the handling of boundary pixels.

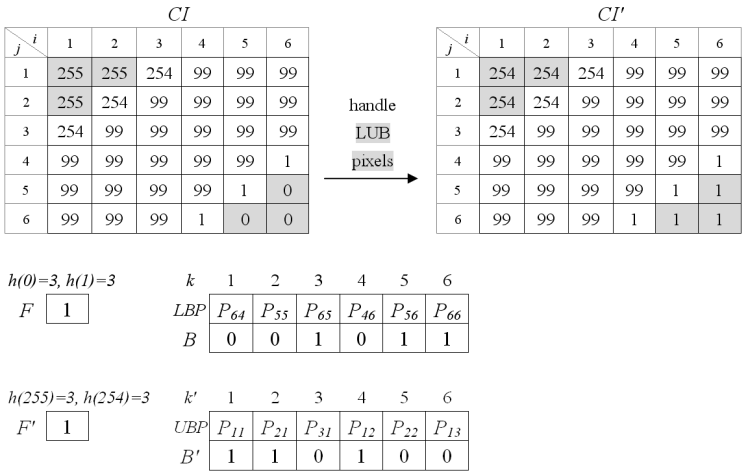


Fig. 3. Example of handling LUB pixels.

Step 2: Derive width m and height n of the optimal block that maximize embedding capacity, and find the maximum and minimum frequency differences a and b in the median difference histogram.

To determine the optimal block ($m \times n$-pixel block) that maximizes embedding capacity, various sizes of block ($s \times t$-pixel blocks, for $1 \leqq s, t \leqq 4$ and $s \times t \geqq 2$) are used to calculate embedding capacities; the block that yields the maximum embedding capacity is selected. For $1 \leqq s, t \leqq 4$ and $s \times t \geqq 2$, divide CI' into $s \times t$-pixel blocks. Let $P'(k)$, $1 \leqq k \leqq s \times t$, denote an array that records and sorts the pixel values in an $s \times t$-pixel block, and let M be the median pixel value in P'.

$$M = P'\left(\left\lceil \frac{s \times t}{2} \right\rceil\right)$$

The list of median difference values in the block is calculated using $\{|P'(k) - M|, 1 \leq k \leq s \times t\} - \{0\}$ ($\{0\}$ is the difference calculated for the median pixel).

Let $h_{st}(d)$ denote frequency (the number of pixels with an absolute difference) of d, and let C_{st} be the embedding capacity using $s \times t$-pixel block. C_{st} is given by

$$C_{st} = \max(h_{st}(d)), \text{ for } 0 \leq d \leq 255.$$

Let $m, n = \underset{s,t}{\operatorname{argmax}}(C_{st})$ for $1 \leqq s, t \leqq 4$ and $s \times t \geqq 2$, then m represents the optimal block width, n is the optimal block height, and C_{mn} denotes the maximum embedding capacity using the optimal block. Let $a = \underset{a'}{\operatorname{argmax}}(h_{mn}(a'))$ for $0 \leqq a' \leqq 255$, then $h_{mn}(a)$ is the maximum frequency and a is the maximum frequency difference in the difference histogram using the optimal block. Let $b = \underset{b'}{\operatorname{argmax}}(h_{mn}(b'))$ for $a < b' \leqq 255$, then $h_{mn}(b)$ is the maximum frequency and b is the minimum frequency difference in the difference histogram using the optimal block. Find m, n, a and b and use them to embed data in the next step. To divide the stego image into the correct blocks, extract data from the correct pixels, and recover the stego image as the cover image in the extraction process. The block difference information (called *BDI*), which comprises m, n, a and b, must be saved as overhead information.

Step 3: Divide the cover image into $m \times n$-pixel blocks; calculate median difference; shift median difference histogram, and embed secret data and overhead information into pixels with difference a.

The median difference and histogram shifting method [15] are used to embed secret data and overhead information. First, calculate the median difference using an $m \times n$-pixel block. Divide CI' into $m \times n$-pixel blocks, and scan the blocks in raster order. Let $P_S(k)$, $1 \leqq k \leqq m \times n$, denote the sorted pixels in the block (input in raster order, sorted by pixel value, while maintaining the relative order of pixels with equal values). Assume that P'_{xy} is the median pixel.

$$P'_{xy} = P_S\left(\left\lceil \frac{m \times x}{2} \right\rceil\right)$$

Let g'_{ij} denote the value of pixel P'_{ij} in a block. The absolute difference d_{ij} of pixel P'_{ij} is calculated using

$$d_{ij} = |g'_{ij} - g'_{xy}|, \text{ for } i \neq x \text{ or } j \neq y.$$

Then, shift the area between $a + 1$ and $b - 1$ of the median difference histogram to the right by one unit, such that the absolute difference values between $a + 1$ to $b - 1$ are increased by one. Finally, embed bit data into pixels with an absolute difference a. If the embedded bit datum is 1, then the absolute difference value is increased by one. The values of pixels with the increased differences must be increased or reduced by one, depending on the median pixel. Let o_{ij} be the index of P'_{ij} in the raster order in the block. Assume that the sorted pixels in a block are divided into sets S, S_e, M', G_e and G, as defined in Fig. 4. The value of pixels in S and S_e is reduced by one if the absolute difference is increased by one, and the value of pixel in G_e and G is increased by one if the absolute difference is increased by one. The numbers of pixels in S and S_e, and in G_e, and G, are after all pixels in the block have been processed equal those before processing. Since P'_{xy} remains the median pixel in the block after processing, the extraction process can extract data and recover the correct pixel value.

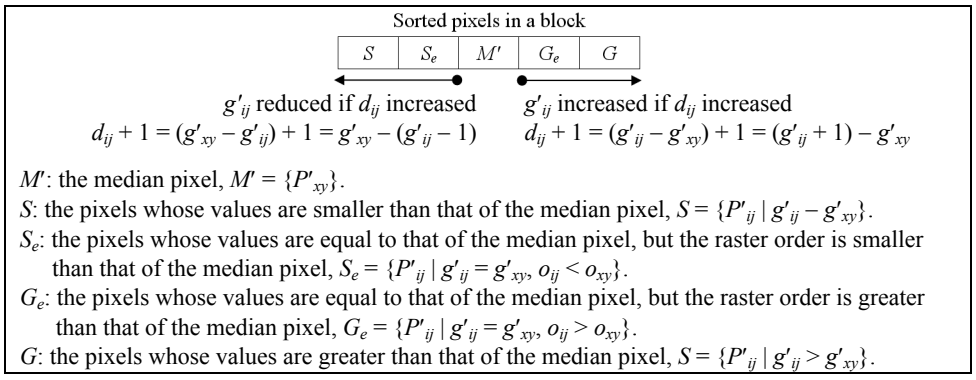


Fig. 4. Changing pixel value when absolute difference value is increased.

2.2 Overhead Information and Key

Figs. 5 and 6 display the processing of overhead information and key in the embedding process and extraction process, respectively. N is the number of embedding processes; EP_k is the kth embedding process; $E'P_k$ is the kth extraction process; SD_k is those secret date that are embedded in EP_k; OI_k is the overhead information that is embedded in EP_k; $LUBI_k$ and BDI_k are the overhead information that is generated in EP_k; marks k.1, k.2, and k.3 indicate the step in EP_k, and C_N is the number of bits embedded in EP_k. The contents of the overhead information and key are

$$OI_1 = LUBI_1$$
$$OI_k = LUBI_k + BDI_{k-1}, \ 2 \leqq k \leqq N$$
$$key = BDI_N + N + C_N$$

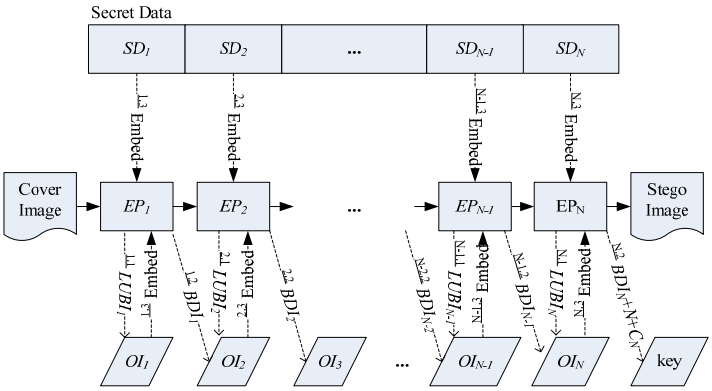


Fig. 5. Processing of overhead information and key in embedding process.

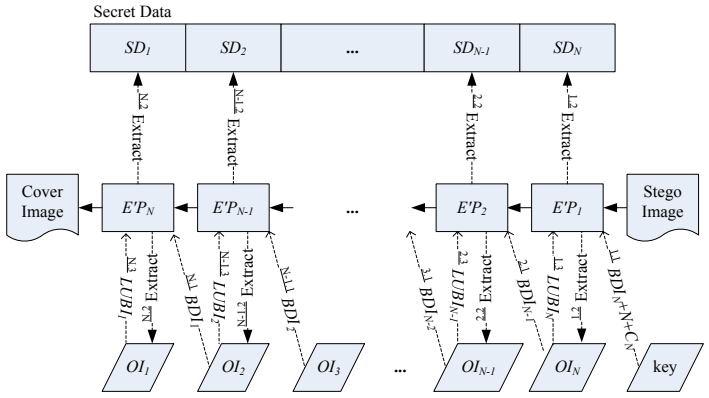


Fig. 6. Processing of overhead information and key in extraction process.

The *LUBI* comprises F, B, F', B', as explained in step 1 of the embedding procedure. F and F' each occupy one bit of embedding space. If $F = 0$, then B need not be saved. If $F = 1$, then B occupies $h(0) + h(1)$ bits of embedding space. If $F' = 0$, then B' need not be saved. If $F' = 1$, then B' occupies $h(254) + h(255)$ bits of embedding space. The *BDI* comprises a, b, m and n. The values of a and b are between 0 and 255, so a and b each occupy 8 bits of embedding space. The values of m and n are between 1 and 4, so m and n each occupy 2 bits of embedding space. Since the value of N is assumed to be less than 256, N occupies 8 bits of embedding space. The embedding capacity in an embedding process is lower than the number of pixels in the cover image. Let W and H are the width and height of the cover image, respectively. C_N occupies $\lceil \log_2 W \times H \rceil$ bits of embedding space.

2.3 Extraction Process

Fig. 7 displays the proposed extraction process. The data are extracted from the stego image (called *SI*) by repeatedly using the extracting process until the data have all been extracted, and the cover image and secret data are output. The extraction process has the following three steps.

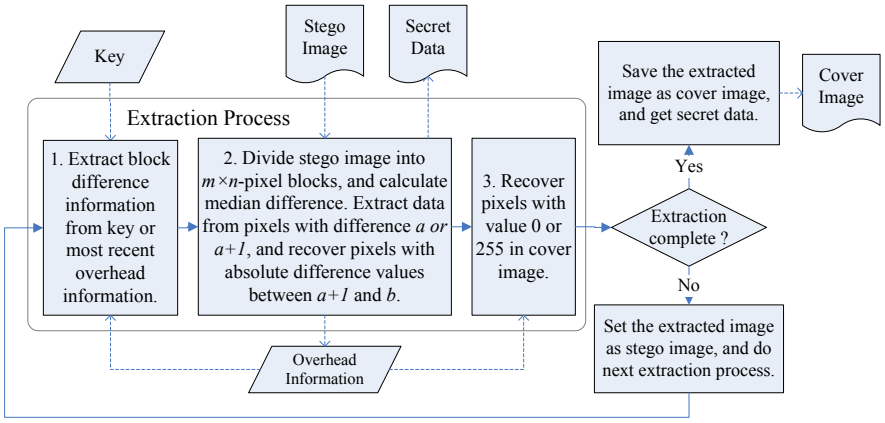


Fig. 7. Proposed extraction process.

Step 1: Extract block difference information from key or most recent overhead information.

In $E'P_1$, obtain BDI_N (includes m, n, a and b), N, and C_N from the key. In $E'P_k$, obtain BDI_k from OI_k, $2 \leqq k \leqq N$.

Step 2: Divide stego image into $m \times n$-pixel blocks, and calculate median difference. Extract data from pixels with difference a or $a + 1$, and recover pixels with absolute difference values between $a + 1$ and b.

First, calculate median difference using $m \times n$-pixel block. Divide SI into $m \times n$-pixels blocks and scan the blocks in raster order. Assume that P'_{xy} is the median pixel in the sorted pixels of the block. Let g'_{ij} be the value of pixel P'_{ij} in a block. The absolute difference d_{ij} of pixel P'_{ij} is calculated using

$$d_{ij} = |g'_{ij} - g'_{xy}|, \text{ for } i \neq x \text{ or } j \neq y.$$

Then, extract bit data 0 and 1 from pixels with absolute difference a and $a + 1$, respectively. The extracted data include the overhead information and some of the secret data. In $E'P_1$, extract C_N bits of data, but in $E'P_k$, $2 \leqq k \leqq N$, extract all of the embedded data.

Finally, recover the pixels whose absolute difference values were shifted in step 3 of the embedding process. The absolute difference value between $a + 1$ to b is reduced by one. In other words, the values of pixels with the reduced differences must be incremented or reduced by 1, as determined by the median pixel. If the pixel value is less than the median pixel value, then the pixel value is increased by one. If the pixel value exceeds the median pixel value, then the pixel value is reduced by one. The modified SI is called SI'.

Step 3: Recover pixels with value 0 or 255 in cover image.

Let $h'(x)$ denote the number of pixels with gray value x in SI', where $1 \leqq x \leqq 254$. Let $g(P_{ij})$ denote the gray value of pixel P_{ij} (jth row and ith column) in SI'. First, obtain $LUBI$ (F, B, F' and B') from the extracted data (E) in step 1. Let F be the first bit in E. If $F = 1$, then let B be the following $h(1)$ bits in E. Now, let F' be the next bit in E. If $F' = 1$, then let B' be the following $h(254)$ bits in E. In the first $N - 1$ instances of extraction, let a, b, m and n be the next 8, 8, 2 and 2 bits in E, respectively, and use them in the next ex-

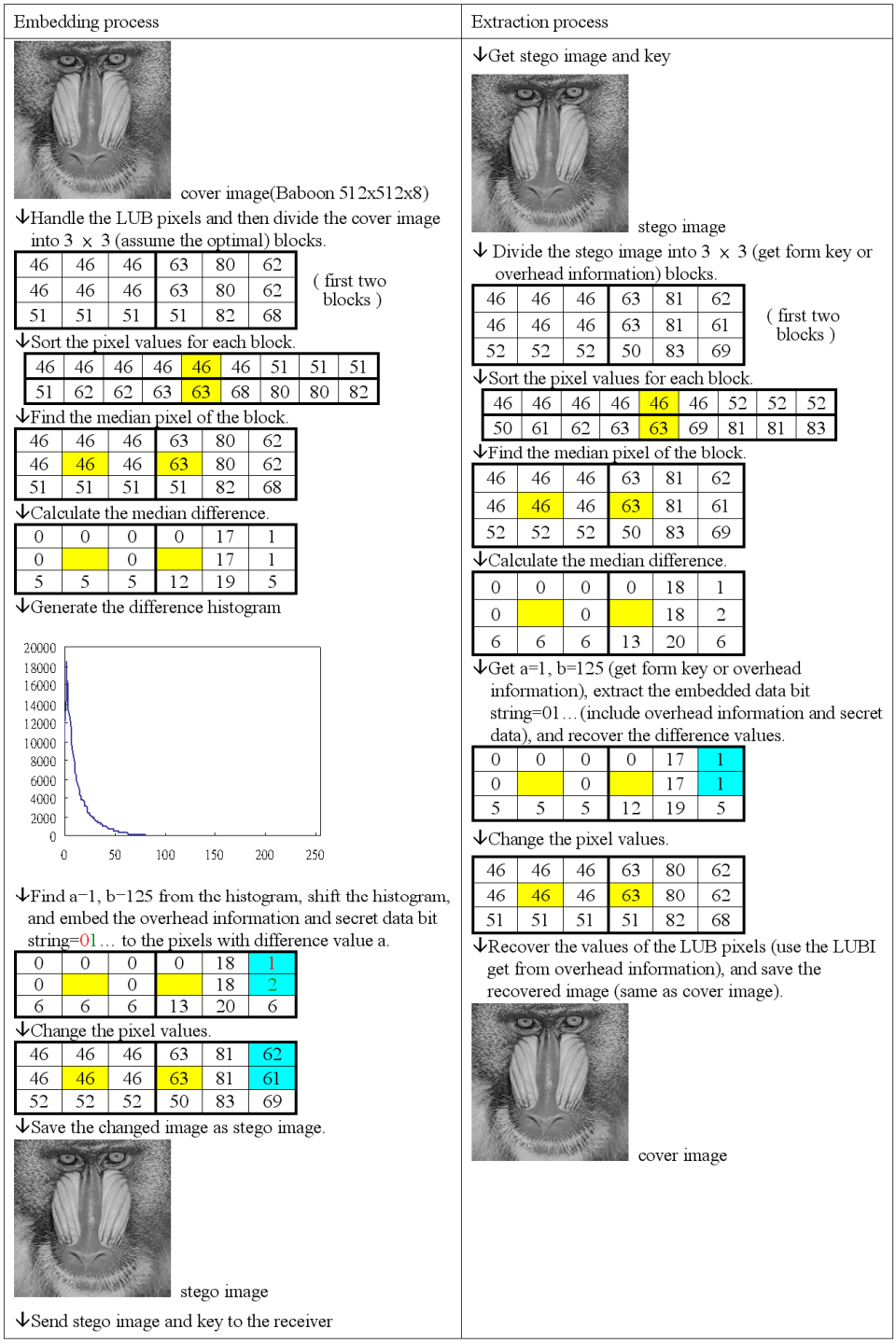


Fig. 8. Example of proposed scheme applied to Baboon (512 × 512 × 8) image.

traction process. Then, accumulate the remaining data in E as secret data. Continue to recover boundary pixels of the cover image. Let LBP be the pixels with value 1 in SI'; $LBP(k)$ denote the kth pixel in LBP, and $B(k)$ denote the kth bit in B, where $1 \leqq k \leqq h'(1)$. Let UBP be the pixels with value 254 in SI'; $UBP(k')$ denote the k'th pixel in UBP, and $B'(k')$ denote the k'th bit in B', where $1 \leqq k' \leqq h'(254)$. The LUB pixels in CI are recovered by using $B(k)$ and $B'(k')$ to specify $g(LBP(k))$ and $g(UBP(k'))$:

$$g(LBP(k)) = \begin{cases} 0 & \text{if } B(k) = 1, \\ 1 & \text{if } B(k) = 0. \end{cases}$$

$$g(UBP(k')) = \begin{cases} 255 & \text{if } B'(k') = 1, \\ 254 & \text{if } B'(k') = 0. \end{cases}$$

Fig. 8 displays an example of the proposed scheme applied to Baboon (512 × 512 × 8) image.

3. EXPERIMENTAL RESULTS

The methods of Ni *et al.* [15], F&S4 (Fallahpour and Sedaaghi with 4 blocks) [16], F&S16 (Fallahpour and Sedaaghi with 16 blocks) [16], L&H (Lin and Hsueh) [17], fixed site difference and the proposed scheme (median difference) were implemented in the Java programming language. The performance of each method was tested using ten 512 × 512 grayscale images as cover images and part of a 512 × 512 grayscale image as secret data, as shown in Fig. 9. Performance was evaluated using the payload (embedding capacity minus overhead information size = size of embedded secret data) and PSNR (peak signal to noise ratio), which is calculated as follows,

$$PSNR = 10 \times \log_{10} \left(\frac{255 \times 255 \times W \times H}{\sum_{i=0}^{W \times H - 1} (x_i - y_i)^2} \right)$$

where W and H are the width and height of the cover image, respectively, and x_i and y_i denote the pixel values of the cover and stego images, respectively. A higher payload and PSNR represents better performance.

First, the payloads of fixed site difference and median difference methods are compared. Table 1 presents the average payloads (bits) for the test images using different block sizes ($m \times n$, for $1 \leqq m, n \leqq 4$ and $m \times n \geqq 2$) of the fixed site difference method in EP_1, designating every pixel of the block as base pixel. An $m \times n$ block contains $m \times n$ fixed site differences. For example, a block of size four contains blocks of three sizes – 1 × 4, 2 × 2 and 4 × 1 – so block size 4 contains 12 fixed site differences. The average pay load of the 12 fixed site differences when the Baboon image is used with a block size of 4 is 12505. Table 1 shows that the block size that maximizes the payload varies with the image, approximately centered on block size 9. Table 2 presents the average payloads (bits) of the test images with various block sizes ($m \times n$, for $1 \leqq m, n \leqq 4$ and $m \times n \geqq 2$) obtained using the median difference method in EP_1. Since the block only has one median pixel, an $m \times n$ block contains only a median difference. It yields a result similar to

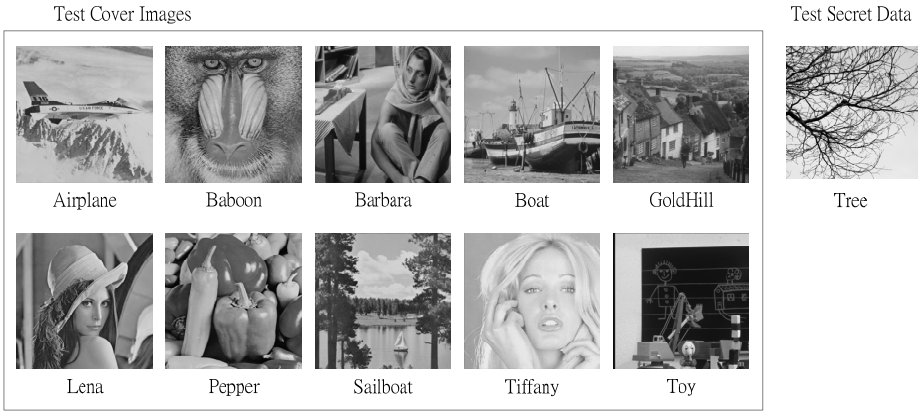


Fig. 9. 512 × 512 grayscale cover images and secret data image tested in experiments.

Table 1. Average payloads for various block sizes obtained using fixed site difference in EP_1.

	Block size (m x n, 1<=m,n<=4 and m x n>=2)							
	2	3	4	6	8	9	12	16
Baboon	9686	11526	12505	13657	13522	13577	13348	13111
Barbara	17140	20950	23102	25441	25539	25717	25479	25166
Boat	25814	32968	36524	39768	40828	41110	41505	41669
GoldHill	16915	21156	22960	24954	24959	25156	24796	24360
Jet	33099	41456	45510	50331	50816	51492	51324	51002
Lena	22362	28015	30798	33972	34263	34552	34440	34165
Pepper	26933	32937	35875	39604	39389	39887	39226	38513
Sailboat	16667	21758	24310	27231	27645	28142	28195	28041
Tiffany	28085	35324	38842	43149	43516	44326	44206	43921
Toy	26308	33754	37458	40770	41865	42223	42642	42766
average	22301	27984	30788	33888	34234	34618	34516	34271

The maximum payload of each image

Table 2. Average payloads for various block sizes obtained using median difference in EP_1.

	Block size (m x n, 1<=m,n<=4 and m x n>=2)							
	2	3	4	6	8	9	12	16
Baboon	9686	16071	16046	17866	17829	18248	17578	17196
Barbara	17140	27522	28901	32228	32509	33409	32651	32499
Boat	25814	41573	44436	49467	51359	52749	52910	53418
GoldHill	16915	28340	29001	32206	32225	33338	32492	31646
Jet	33099	47463	51115	58054	59203	60777	60827	60774
Lena	22362	36548	38279	43321	44190	45568	45057	44869
Pepper	26933	41884	43818	50020	50545	51932	51070	50491
Sailboat	16667	28722	30421	34801	35702	36859	36842	36896
Tiffany	28085	43952	46659	53275	54449	56241	56131	55835
Toy	26308	43246	45875	51262	53168	55085	55022	55362
average	22301	35532	37455	42250	43118	44421	44058	43899

The maximum payload of each image

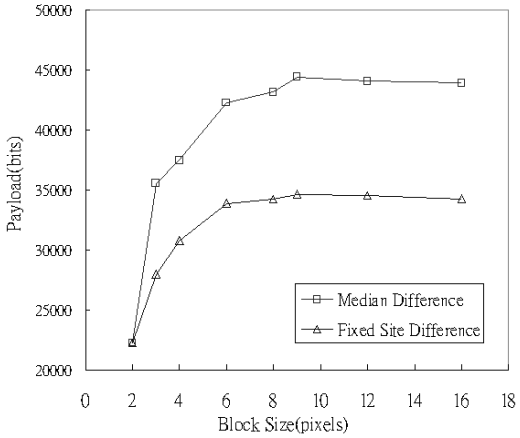


Fig. 10. Comparison between average payloads obtained using fixed site difference and median difference.

Table 3. Performance of proposed method using various block sizes.

block size	3		4			6		8		9	12		16	optimal
CI \ m×n	1x3	3x1	1x4	2x2	4x1	2x3	3x2	2x4	4x2	3x3	3x4	4x3	4x4	optimal
Airplane	1.055	1.083	1.072	1.151	1.115	1.190	1.196	1.215	1.236	1.242	1.182	1.259	1.195	1.351
Baboon	0.414	0.499	0.399	0.483	0.466	0.478	0.483	0.484	0.486	0.496	0.491	0.488	0.452	0.515
Barbara	0.676	0.601	0.672	0.681	0.601	0.739	0.720	0.718	0.689	0.716	0.721	0.709	0.715	0.801
Boat	0.953	0.909	0.977	0.971	0.952	1.017	1.019	1.048	1.063	1.076	1.083	1.093	1.041	1.191
GoldHill	0.685	0.762	0.705	0.764	0.763	0.777	0.796	0.781	0.818	0.820	0.754	0.767	0.753	0.858
Lena	0.947	0.832	0.962	0.948	0.835	0.998	1.010	1.034	0.972	1.025	0.979	1.011	1.027	1.072
Pepper	0.919	0.936	0.917	1.022	0.914	1.080	1.071	1.059	1.039	1.083	1.076	1.062	1.070	1.192
Sailboat	0.736	0.743	0.748	0.739	0.714	0.812	0.816	0.795	0.796	0.863	0.820	0.821	0.827	0.898
Tiffany	1.089	1.030	1.119	1.143	1.059	1.204	1.187	1.245	1.220	1.258	1.214	1.204	1.216	1.302
Toy	1.047	0.995	1.077	1.074	1.074	1.143	1.156	1.125	1.141	1.229	1.172	1.182	1.183	1.216
average	0.852	0.839	0.865	0.898	0.849	0.944	0.945	0.950	0.946	0.981	0.949	0.960	0.948	1.040

The maximum payload of each image

The maximum payload (except optimal block) of each image

that obtained using fixed site difference; on average, a block size of 9 (3 × 3 block) maximizes the payload. Fig. 10 compares the average payloads obtained using fixed site difference and median difference. The figure reveals that the median difference yields a higher payload than fixed site difference, since the median difference reduce the values of differences, and increase the maximum frequency of the histogram.

The performance of the proposed method was evaluated using various block sizes. The optimal blocks were obtained in step 2 of the embedding process. This embedding process was repeated until the PSNR approached 30dB. Table 3 presents the results of this process, where the unit of payload is bpp (bits per pixel). The 3 × 3 block yielded the largest average payload, 0.981bpp, but the optimal block had maximum payload, 1.040bpp. In each image comparison, the optimal block size performed best for the first nine images, but not the Toy image. Therefore, the proposed method performs better when the optimal block size is used than when a stable block size is used.

Table 4. Comparison of performance of Ni, F&S4, F&S16, L&H, and proposed method.

Images	Ni [15]		F&S4 [16]		F&S16 [16]		L&H [17]		Proposed	
	Payload	PSNR	Payload	PSNR	Payload	PSNR	Payload	PSNR	Payload	PSNR
Airplane	0.48	30.5	0.62	30.5	0.77	30.4	0.81	30.4	1.35	30.5
Baboon	0.18	30.1	0.20	30.4	0.23	30.4	0.38	30.3	0.52	30.7
Barbara	0.16	30.4	0.17	30.3	0.25	30.1	0.44	30.1	0.80	30.3
Boat	0.34	30.4	0.39	30.4	0.54	30.2	0.64	30.6	1.19	30.0
GoldHill	0.15	30.5	0.21	30.2	0.28	30.5	0.56	30.1	0.86	30.2
Lena	0.17	30.1	0.26	30.1	0.38	30.2	0.62	30.2	1.07	30.5
Pepper	0.18	30.3	0.21	30.2	0.34	30.4	0.73	30.3	1.19	30.3
Sailboat	0.30	30.6	0.31	30.4	0.48	30.3	0.53	30.3	0.90	30.3
Tiffany	0.36	30.2	0.49	30.4	0.60	30.4	0.76	30.1	1.30	30.3
Toy	0.59	30.3	0.64	30.1	0.84	30.2	0.70	30.5	1.22	30.5
average	0.29	30.3	0.35	30.3	0.47	30.3	0.62	30.3	1.04	30.4

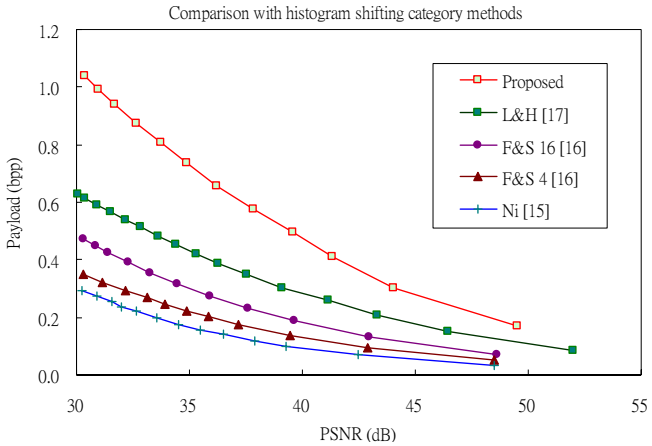


Fig. 11. Comparison of average performance of histogram shifting category methods.

The proposed method was then implemented with median difference and the optimal block. The performance was measured and compared with those of Ni [15], F&S4 [16], F&S16 [16] and L&H [17]. The embedding process was repeated until the PSNR approached 30dB. Table 4 presents the results of this process, where the unit of payload is bpp (bits per pixel). For each image and on average, the proposed method outperformed the other methods. The proposed method had an average payload of 1.04bpp when PSNR = 30.4dB. Fig. 11 compares the mean performance of histogram shifting category methods. In the figures, each node represents the value of payload and PSNR in each period of the embedding process. The figures reveal that the proposed method outperforms the other four histogram-shifting category methods.

Finally, experimental data were obtained using the other categories of reversible data hiding methods that were described in section 1, and compared with the experimental data obtained using the proposed method. Figs. 12, 13 and 14 compare the average performance of the proposed method with that of the data compression, difference expansion and integer wavelet transform category methods, respectively. The figures reveal that the proposed method outperforms these other methods.

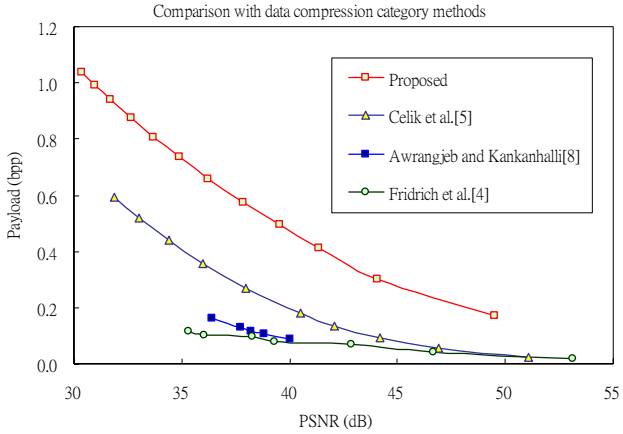


Fig. 12. Comparison of average performance of proposed method with that of data compression category methods.

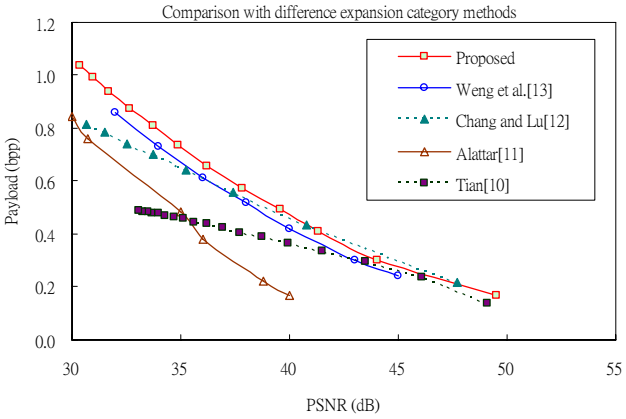


Fig. 13. Comparison of average performance of proposed method with that of difference expansion category methods.

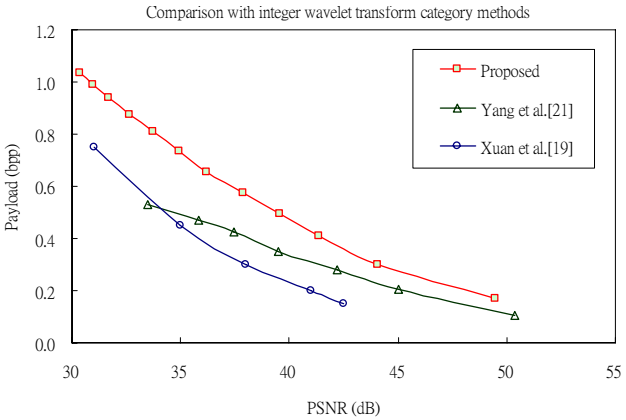


Fig. 14. Comparison of average performance of proposed method with that of integer wavelet transform category methods.

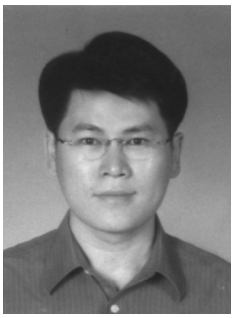
4. CONCLUSIONS

The proposed method utilizes the difference between the values of neighboring pixels in a block to generate a difference histogram. The median pixel in the block is used to calculate the differences to increase the maximum frequency of the histogram, and the optimal block size is determined to yield the maximum embedding capacity. Experimental results reveal that the proposed method outperforms many other reversible data hiding methods.

REFERENCES

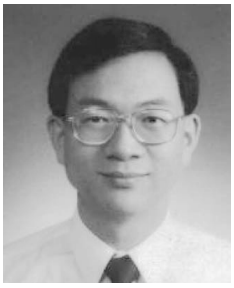
1. M. Awrangjeb, "An overview of reversible data hiding," in *Proceedings of International Conference on Computer and Information Technology*, 2003, pp. 75-79.
2. Y. Q. Shi, Z. Ni, D. Zou, C. Liang, and G. Xuan, "Lossless data hiding: fundamentals, algorithms and applications," in *Proceedings of IEEE International Symposium on Circuits and Systems*, Vol. II, 2004, pp. 33-36.
3. J. Fridrich, M. Goljan, and R. Du, "Invertible authentication," in *Proceedings of SPIE Security and Watermarking of Multimedia Contents*, 2001, pp. 197-208.
4. J. Fridrich, M. Goljan, and R. Du, "Lossless data embedding-new paradigm in digital watermarking," *EURASIP Journal on Applied Signal Processing*, Vol. 2, 2002, pp. 185-196.
5. M. U. Celik, G. Sharma, A. M. Tekalp, and E. Saber, "Reversible data hiding," in *Proceedings of IEEE International Conference on Image Processing*, Vol. 2, 2002, pp. 157-160.
6. M. U. Celik, G. Sharma, A. M. Tekalp, and E. Saber, "Lossless generalized-LSB data embedding," *IEEE Transactions on Image Processing*, Vol. 14, 2005, pp. 253-266.
7. M. U. Celik, G. Sharma, and A. M. Tekalp, "Lossless watermarking for image authentication: A new framework and an implementation," *IEEE Transactions on Image Processing*, Vol. 15, 2006, pp. 1042-1049.
8. M. Awrangjeb and M. S. Kankanhalli, "Lossless watermarking considering the human visual system," *Lecture Notes in Computer Science*, Vol. 2939, 2004, pp. 329-336.
9. M. Awrangjeb and M. S. Kankanhalli, "Reversible watermarking using a perceptual model," *Journal of Electronic Imaging*, Vol. 14, 2005, pp. 1-8.
10. J. Tian, "Reversible data embedding using a difference expansion," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 13, 2003, pp. 890-896.
11. A. M. Alattar, "Reversible watermark using the difference expansion of a generalized integer transform," *IEEE Transactions on Image Processing*, Vol. 13, 2004, pp. 1147-1156.
12. C. C. Chang and T. C. Lu, "A difference expansion oriented data hiding scheme for restoring the original host images," *The Journal of Systems and Software*, Vol. 79, 2006, pp. 1754-1766.
13. S. Weng, Y. Zhao, J. S. Pan, and R. Ni, "A novel reversible watermarking based on an integer transform," in *Proceedings of IEEE International Conference on Image Processing*, Vol. 3, 2007, pp. 241-244.
14. S. Weng, Y. Zhao, J. S. Pan, and R. Ni, "Reversible watermarking based on invariability and adjustment on pixel pairs," *IEEE Signal Processing Letters*, Vol. 15, 2008,

pp. 721-724.
15. Z. Ni, Y. Q. Shi, N. Ansari, and W. Su, "Reversible data hiding," *IEEE Transactions on Circuits Systems and Video Technology*, Vol. 16, 2006, pp. 354-362.
16. M. Fallahpour and M. H. Sedaaghi, "High capacity lossless data hiding based on histogram modification," *IEICE Electronics Express*, Vol. 4, 2007, pp. 205-210.
17. C. C. Lin and N. L. Hsueh, "Hiding data reversibly in an image via increasing differences between two neighboring pixels," *IEICE Transactions on Information and Systems*, Vol. E90-D, 2007, pp. 2053-2059.
18. G. Xuan, J. Chen, J. Zhu, Y. Q. Shi, Z. Ni, and W. Su, "Lossless data hiding based on integer wavelet transform," in *Proceedings of IEEE Workshop on Multimedia Signal Processing*, 2002, pp. 9-11.
19. G. Xuan, Y. Q. Shi, Z. C. Ni, J. Chen, C. Yang, Y. Zhen, and J. Zheng, "High capacity lossless data hiding based on integer wavelet transform," in *Proceedings of International Symposium on Circuits and Systems*, Vol. 2, 2004, pp. 29-32.
20. G. Xuan, Y. Q. Shi, C. Yang, Y. Zheng, D. Zou, and P. Chai, "Lossless data hiding using integer wavelet transform and threshold embedding technique," in *Proceedings of IEEE International Conference on Multimedia and Expo*, 2005, pp. 1520-1523.
21. L. Yang, P. Hao, and C. Zhang, "Progressive reversible data hiding by symmetrical histogram expansion with piecewise-linear haar transform," in *Proceedings of IEEE International Conference on Acoustics*, *Speech*, *and Signal Processing*, Vol. II, 2007, pp. 265-268.



Hsien-Wei Yang (楊仙維) received his M.S. degree and Ph.D. from the Department of Applied Mathematics, National Chung Hsing University, Taiwan, in 1999 and 2009, respectively. He is currently an Assistant Professor in the Department of Information Management, Overseas Chinese University, Taiwan. His current research interests include image processing and information hiding.

I-En Liao (廖宜恩) received his B.S. degree in Applied Mathematics from National Cheng-Chi University, Taiwan, in 1978, and both his M.S. in Mathematics and the Ph.D. in Computer and Information Science from the Ohio State University in 1983 and 1990, respectively. He is currently a Professor in the Department of Computer Science of National Chung Hsing University, Taiwan. His research interests are in database tuning, data mining, XML database, and bioinformatics. He is a member of the ACM and the IEEE Computer Society.



Chaur-Chin Chen (陳朝欽) received his B.S. degree in Mathematics from National Taiwan University, Taipei, in 1977, and M.S. degrees in both Mathematics and Computer Science and a Ph.D. degree in Computer Science, all from Michigan State University (MSU), East Lansing, in 1982, 1984, and 1988, respectively. He worked as a research associate at MSU in 1989. He is currently a Professor in the Department of Computer Science at National Tsing Hua University. He was a visiting scholar at MSU in 1997. His current research interests are information hiding, biometrics, and microarray image data analysis.