JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 27, 1265-1282 (2011)

A Lossless Large-Volume Data Hiding Method Based on Histogram Shifting Using an Optimal Hierarchical Block Division Scheme*

CHE-WEI LEE[1] AND WEN-HSIANG TSAI[1,2]
[1]*Department of Computer Science*
National Chiao Tung University
Hsinchu, 300 Taiwan
[2]*Department of Information Communication*
Asia University
Taichung, 413 Taiwan

A lossless large-volume data hiding method based on histogram shifting is proposed. The method is based on a scheme of hierarchically dividing a cover image into smaller blocks for data embedding using the histogram shifting technique, which yields a large data hiding capacity and results in a high stego-image quality. A technique for recursive looking-ahead estimation of the maximum data hiding volume at the lowest level of the block-division tree structure is proposed to yield an optimal data hiding result. The technique is shown to break a bottleneck of data-hiding-rate increasing at the image block size of 8×8, which is found to exist in other histogram-shifting methods. Four ways of block divisions are used, and one of them is selected optimally in each tree level of block divisions. A series of experiments have been conducted, and superiority of the proposed method to other related methods is shown by comparing the respective experimental results. A good property of the proposed method observed in the experiments is also pointed out, that is, the data hiding rate yielded by the proposed method increases without degrading the stego-image quality, which is an unusual phenomenon in data hiding researches.

Keywords: lossless data hiding, histogram shifting, block division, stego-image, steganography

1. INTRODUCTION

With the advance of computer networks and signal processing, digital multimedia are spread widely through the Internet nowadays. This causes the security problem of exposing transmitted digital data on the network with the risk of being copied or intercepted illegally. In order to protect the privacy of private data, various cryptographic techniques have been proposed to encrypt these data before conducting data transmission. However, with considerable increasing of the computing powers of modern computers, the security of the data yielded by these techniques is threatened. In addition, though cryptographic techniques encrypt secret messages into unrecognizable forms before transmission, the undisguised appearances of the encrypted message would easily arouse suspicion and bring on unexpected attacks from hackers.

The development of information hiding techniques provides another solution to protecting digital media. Such techniques may be employed to embed private or secret in-

Received March 22, 2010; revised May 27 & July 20, 2010; accepted August 26, 2010.
Communicated by Chung-Lin Huang.
* This work was supported by the National Science Council of Taiwan, R.O.C., project No. 97-2631-H-009-001.

formation into cover media in such a way that the existence of the hidden information is imperceptible but known only to a pre-concerted recipient. The cover media can be of various types of digital data, such as image, text, video, *etc.* Information like private annotations, business logos, and critical intelligence can be embedded into a cover image in an invisible form so that many applications, like ownership claim of digital contents, copyright protection of media, covert communication between parties, *etc.*, can be fulfilled. Information hiding techniques used for covert communication are often called *steganography*, and those for ownership or copyright protection are often called *watermarking*.

In the early phase, conventional steganography [1-4] emphasizes exploring higher hiding capacities and pursuing lower quality degradations in *watermarked images* (also referred to as *stego-images* in the sequel). In general, a small amount of content loss will occur in the stego-image, though often imperceptible. However, such a loss is not desirable in some applications, such as legal documentation, military reconnaissance, high-precision scientific investigation, *etc.*, because it may lead to risks of incorrect decision making. In view of this, a type of novel data hiding technique, which is referred to as *reversible*, *invertible*, *lossless*, or *distortion-free*, has been developed in recent years. In this study, a reversible data hiding method which yields stego-images with good qualities and high data hiding capacities is proposed.

Reversible data hiding techniques can be employed to restore stego-images to their pristine states after the hidden data are extracted. Such techniques can be classified into three groups: (1) based on data compression [5-7]; (2) based on pixel-value difference expansion [8, 9]; and (3) based on histogram shifting [10-13]. The strategy used in the techniques of the first group is to compress message data as well as related information and embed the result directly into the cover image. A method in this group is Barton [5] which compresses the secret message before embedding them into the bit stream of digital data. Celik *et al.* [7] proposed a high-capacity lossless data hiding method which quantizes each image pixel by into L-level scales, compresses the quantization residues, and embeds the secret bits as well as the compressed data into the quantified image by the least-significant-bit (LSB) substitution technique.

The second group of reversible data hiding methods aims to explore the redundancy of pixel values in images. Tian [8] proposed a technique of pixel-value difference expansion by performing fundamental arithmetic operations on pairs of pixels to discover hidable space. However, not all pairs can be expanded for data hiding. A location map is used to indicate whether pairs are expanded or not. An enhanced pixel-value difference expansion method proposed by Kim *et al.* [9] used a refined location map and a new concept of expandability to achieve higher data hiding capacities while keeping the resulting image distortion as low as that yielded by Tian's method [8].

The last group of reversible data hiding methods, to which the proposed method belongs, is based on the concept of histogram shifting. Ni, *et al.* [10] proposed a reversible data hiding method which shifts slightly the part of the histogram between the maximum point (also called the *peak point*) and the minimum point to the right side by one pixel value to create an empty *bin* besides the maximum point for hiding an input message. Advantages of this method include yielding superior hiding capacities and providing higher qualities in stego-images. The knowledge of the maximum point and the minimum point of the histogram is necessary for retrieving the hidden data and restoring the stego-image losslessly to the original state. In addition, the coordinates of the pixels

whose gray values equal to the gray value of the minimum point *b* need be recorded as overhead information when the value of *b* is not zero. Consideration of multiple pairs of maximum and minimum points was also included in the method in order to raise the data hiding capacity, at the sacrifice of the resulting stego-image quality. A problem occurs here when too many of such pairs are selected for data hiding. In such a case, a rapid increase of the size of the overhead information, which cannot be embedded completely in the cover image, might occur. Fallahpour [11] later proposed the idea of decomposing the entire cover image into blocks and using the peak point of the histogram of each block to hide data. The technique of block division successfully improves the data hiding capacity and keeps the stego-image quality at the same level, compared with those of Ni *et al.*'s method [10]. Hwang, *et al.* [12] proposed the concept of slightly adjusting the pixel values located at both sides of a histogram peak to embed message data. There is no need to record the knowledge of the location of the peak point because the peak location will not be changed after data hiding. But a so-called *location map* is still required to keep the information for restoring the cover image. For cases where the values of minimum points are not zero, the location map should include the location of the histogram peak point and those of the local minimum points to the left and right of the peak. The data hiding capacity of Hwang *et al.*'s method, when compared with that of Ni *et al.*'s, is smaller, and the peak signal-to-noise ratio (PSNR) of the stego-image gets worse in some cases [13]. A modification of the method using several pairs of peak points and minimum points instead of just one was also proposed. However, the more of such pairs are selected, the larger the decrease in the data hiding capacity becomes, because more information of the selected minimum points and the reversible points need be kept in the location map. Later, Kuo, *et al.* [13], similar to Fallahpour [11], used the block division technique to increase the data hiding capacity of Hwang, *et al.* [12].

An important characteristic of the aforementioned histogram-shifting-based data hiding methods is that the existence of more histogram peaks implies a higher data hiding capacity. For techniques of *hiding data besides the peak*, more peaks even signify higher PSNR values in the stego-images because the gray values of those pixels forming the peak will remain unchanged after data embedding. Therefore, in this study, we try to explore the possibility of using the *largest* number of peaks, instead of just using only one, in a block to maximize the data hiding capability and minimize the distortion in the stego-image.

More specifically, based on our detailed experimental observation that the sum of the data hiding capacities provided by using the peaks from a group of sub-blocks in a cover image generally will be larger than that provided by using a single peak in the same cover image, a new reversible histogram-shifting data hiding method which uses a looking-ahead strategy to achieve optimal hierarchical block division for improving data hiding capacities and stego-image qualities is proposed. Each non-overlapping square block in a cover image is divided recursively by four ways of sub-block decompositions. And the optimal way which provides the highest data hiding capacity is chosen by a looking-ahead hiding capacity estimation scheme, which searches the tree formed by the hierarchically divided blocks down to the lowest level. Compared with many existing methods, larger data hiding capacities with lower stego-image quality degradation can be achieved. Good experimental results demonstrating the effectiveness of the proposed method are also shown.

The remainder of this paper is organized as follows. In section 2, a non-recursive

version of the proposed method is described. In section 3, an optimal recursive version of the proposed method is presented. Experimental results and some discussions are included in section 4. Conclusions are made finally in section 5.

2. HIERARCHICAL BLOCK DIVISION FOR HISTOGRAM-SHIFTING DATA HIDING

As mentioned previously, more peaks yield larger data hiding capacities. In Ni *et al.* [10], the number of pixels constituting the peak in the histogram of a cover image is equal to the data hiding capacity because only a single peak in a cover image is used. Thus, Fallahpour [11] divided the cover image into blocks so as to generate a respective peak for each block. This technique of *block division* successfully enhances the data hiding capacity because the total data that can be hidden in multiple blocks is generally larger than that can be hidden in a single cover image, as mentioned previously. Furthermore, the location of the peak in the histogram indicates generally that a great number of pixels are 'gathered' in the neighbor area around the peak point. For this reason, Hwang *et al.* [12] used the two neighboring points beside the peak point to embed data. On the other hand, the block division technique was also used by Kuo *et al.* [13] for improving the performance of Hwang *et al.* [12] – a cover image of size 512 × 512 was divided into four blocks.

In this study, to see the trend of data-hiding-rate increasing by block divisions, we divided the cover image into equal-sized sub-blocks from size 256 × 256 down to 2 × 2, and implemented the method of Kuo *et al.* [13] to test the resulting blocks. A surprising result was observed in the experimental results, *i.e.*, the data-hiding rate increases from the size of 256 × 256 through 8 × 8, and then turns to *decrease* from 4 × 4 to 2 × 2. Figs. 1-3 show this trend. Additionally, the trend of the PSNR values of the three stego-images is also shown as red curves over the bar charts in the figures, which says that the PSNR values keep *increasing* from the size of 256 × 256 all way down to 2 × 2, contrary to the intuition that hiding more data will result in worsening the stego-image quality.

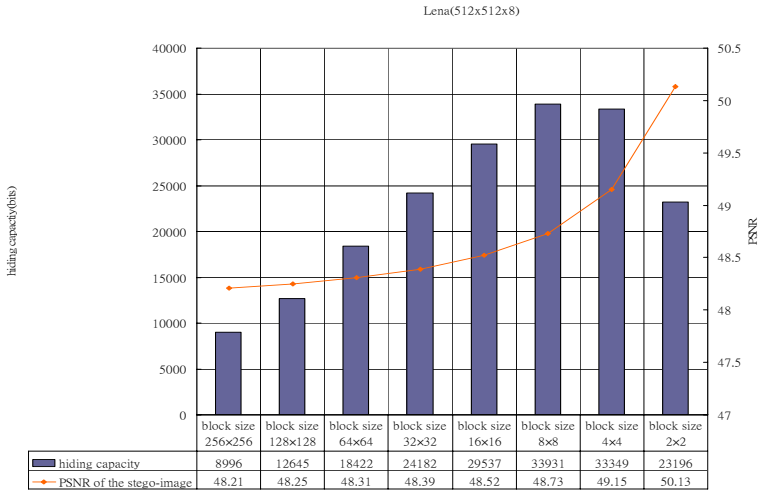
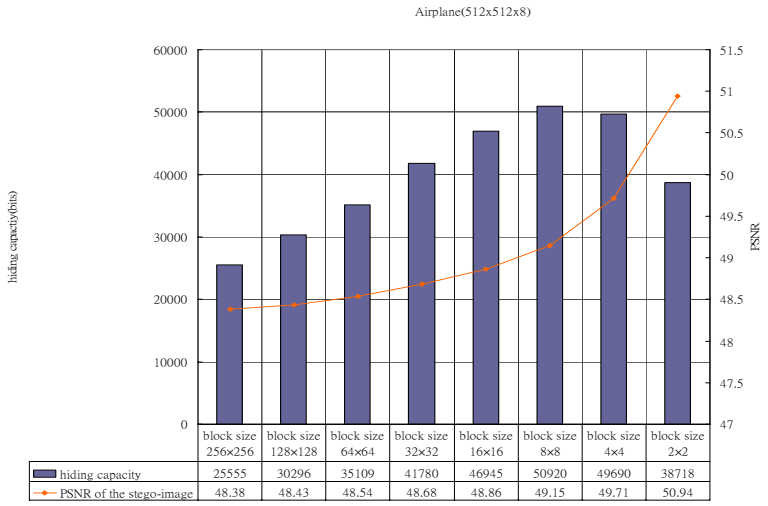
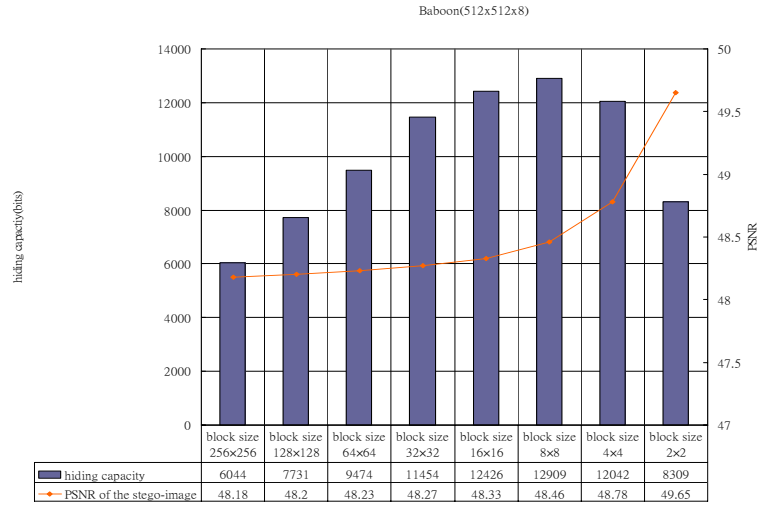


Fig. 1. Trend of data hiding capacities and PSNR values versus block sizes in image 'Lena.'



	block size 256×256	block size 128×128	block size 64×64	block size 32×32	block size 16×16	block size 8×8	block size 4×4	block size 2×2
hiding capacity	25555	30296	35109	41780	46945	50920	49690	38718
PSNR of the stego-image	48.38	48.43	48.54	48.68	48.86	49.15	49.71	50.94

Fig. 2. Trend of data hiding capacities and PSNR values versus block sizes in image 'Airplane.'



	block size 256×256	block size 128×128	block size 64×64	block size 32×32	block size 16×16	block size 8×8	block size 4×4	block size 2×2
hiding capacity	6044	7731	9474	11454	12426	12909	12042	8309
PSNR of the stego-image	48.18	48.2	48.23	48.27	48.33	48.46	48.78	49.65

Fig. 3. Trend of data hiding capacities and PSNR values versus block sizes in image 'Baboon.'

From the above observation, the block size of 8 × 8 is seen to be the best choice for maximizing the data hiding capacity while minimizing the image distortion. However, this size may as well be regarded as a bottleneck in the trend of data-hiding-rate increasing. It is desired to break this bottleneck and the investigation result yielded in this study is *positive* – by the proposed method the data hiding rate will be increased again beyond the size of 8 × 8! The basic concept is to divide each 8 × 8 block by four ways into 4 × 8, 8 × 4, or 4 × 4, in addition to keeping the original size of 8 × 8. It is also found in the experimental results of the proposed method that the PSNR value still keeps going up with the decreasing block size.

The four ways of block divisions are shown in Fig. 4. The *optimal* way among the

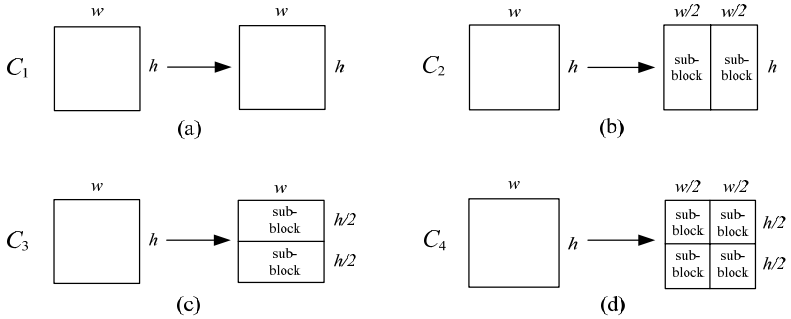


Fig. 4. Four ways of block divisions used in this study.

134	126	129	126	133	130	129	131
134	126	129	126	133	130	129	131
134	126	129	126	133	130	129	131
134	126	129	126	133	130	129	131
134	126	129	126	133	130	129	131
132	125	128	130	130	133	131	130
132	126	133	133	130	130	131	127
132	134	133	134	131	134	131	131

Fig. 5. A block of size 8 × 8 in image 'Lena.'

four, which provides the largest space for data hiding, is chosen by the previously-mentioned looking-ahead scheme for data-hiding-capacity estimation. Fig. 5 shows a block in the image 'Lena' to illustrate a case that a block of 8 × 8 can provide a larger data hiding capacity after it is divided optimally into two 4 × 8 sub-blocks. Originally, the peak point of the entire block is found at the gray value of 126 with a data hiding capacity of only 2 bits. This can be seen from the fact that there are only two pixels with gray values 125 and 127 at the two sides of gray value 126. But after dividing the block horizontally by the way of C_3 as shown in Fig. 4, totally a data hiding capacity of 13 bits can be generated from both the upper and the lower sub-blocks with peak points 129 and 131, respectively, as can be seen from the gray values in the two sub-blocks where there are 13 pixels with gray values 128, 130, and 132, which are located next to gray values 129 and 131.

In the following, the proposed non-recursive method is described in details as two algorithms, one for data embedding and the other for data extraction. A key is generated by the data embedding algorithm (Algorithm 1). The key is used in the data extraction algorithm (Algorithm 2) for security control; without the key, the embedded message data cannot be extracted successfully.

Algorithm 1 Data embedding (a non-recursive version).

Input: a cover image I divided into blocks with size $n \times n$, and a message M in bit string form to be embedded.

Output: a stego-image I' with M embedded, a key K in the form of an integer number sequence, and a location map S_M.

Step 1: (*Block decomposition*) divide each block B in I by the four ways C_1, C_2, C_3, and C_4 as shown in Fig. 4.

Step 2: (*Looking-ahead estimation of data hiding capacities*) compute the data hiding capacity for each case C_i of block divisions, i = 1, 2, 3, and 4, by the following way.

2.1 For each sub-block I_j in C_i, perform the following operations.

(1) Generate the histogram h of I_j.

(2) Find the peak in h and its location x_0 (a gray value).

(3) Sum up the values $h(x_0 - 1)$ and $h(x_0 + 1)$ as d_j.

2.2 Sum up all values of d_j to get the data hiding capacity D_i for C_i.

Step 3: (*Optimal decision*) select the C_i with the maximum capacity D_i among D_1 through D_4, denote it as C_m, and record integer m as a *component k* in the key K for the currently-processed block B.

Step 4: (*Data embedding*) for each sub-blocks I_j in C_m, perform the following steps.

4.1 Generate the histogram h of I_j.

4.2 Find the peak in h and its location x_0.

4.3 Collect all pixels in I_j with gray values smaller than x_0 as a set S_L, and those with gray values larger than x_0 as a set S_R.

4.4 Collect all pixels in S_L whose gray values are zero as a set S_L^0, and all those whose gray values are 255 in S_R as a set S_R^{255}.

4.5 (*Histogram shifting*) decrement the gray value of each pixel in S_L by one except those in S_L^0, and increment the gray value of each pixel in S_R by one except those in S_R^{255}.

4.6 (*Location map generation*) put S_L^0 and S_R^{255} into a set S_M, and call it the *location map* of M.

4.7 (*Bit embedding*) take a data bit m_ℓ sequentially from M, scan the pixels in I_j in a raster-scan order, and take an *unprocessed* pixel with gray value v, and perform the following operation until all the bits in M are processed:

(1) if $v = x_0 - 2$, then
increment v by one if $m_\ell = 1$ or keep v unchanged if $m_\ell = 0$;

(2) if $v = x_0 + 2$, then
decrement v by one if $m_\ell = 1$ or keep v unchanged if $m_\ell = 0$.

Step 5: (*Key generation*) concatenate the previously-generated values of k's of all the blocks in I in the block-processing order into an integer number sequence $k_1k_2k_3$ … as the desired key K.

To enhance the level of security, many data hiding methods make use of keys to control the data extraction process [14, 15]. In general, data are embedded into the cover medium in a specific order which is determined by the key. A key is usually constructed by meaningless numbers, and serves as a seed for a random number generator which produces a series of numbers to specify the data embedding order. In this study, we use differently a scheme of forming the key by the block division information yielded by the method, as done in Algorithm 1. Similar schemes were also adopted in [10, 16, 17].

In addition, about the generated location map, its size will depend on the numbers of pixels with values 0 and 255 existing in the test image. In five test images selected randomly for use in our experiments, the sizes of the location maps generated from them are

all 0. But if other images are used, the sizes of the location maps might not be zero. Generally speaking, pixels with extreme values 0 and 255 are limited in number in common images, and for them the proposed method will yield location maps of small sizes. An example will be shown later in section 4 where we describe our experimental results.

Now, the proposed data extraction process is described in the following.

Algorithm 2 Data extraction (corresponding to Algortihm 1).
Input: a stego-image I' with blocks of size $n \times n$, the location map S_M, and the key K.
Output: the message data M embedded in I' and the original cover image I.
Step 1: (*Initialization*) set $M = \varepsilon$ (an empty string).
Step 2: Take out the blocks in I' in order, and divide each of them, denoted by B', in the way of the block division specified by the corresponding component k in the key K if k is not an empty string (*i.e.*, not ε); if k is ε, then stop this algorithm.
Step 3: (*Data extraction*) for each sub-blocks I_j' in B', perform the following steps.
3.1 Generate the histogram h' of I_j'.
3.2 Find the peak in h' and its location x_0'.
3.3 (*Bit extraction*) scan the pixels in I_j' in a raster-scan order, take an unprocessed pixel with gray value v, and perform the following operation:
(1) if $v = x_0' - 2$ or $x_0' + 2$, then extract a bit "0" and append it to the end of M;
(2) if $v = x_0' - 1$, then extract a bit "1," append it to the end of M, and decrement v by one;
(3) if $v = x_0' + 1$, then extract a bit "1," append it to the end of M, and increment v by one.
3.4 Collect all pixels in I_j' with gray values smaller than x_0' as a set S_L', and those with gray values larger than x_0' as a set S_R'.
3.5 (*Reverse histogram shifting*) perform the following steps to recover the cover image I.
(1) Increment the gray value of each pixel in S_L' by one, and decrement the gray value of each pixel in S_R' by one.
(2) Use the content (S_L^0, S_R^{255}) of the location map S_M to restore the original gray values of the pixels by setting the gray value of each pixel in S_L^0 to be 0 and that of each pixel in S_R^{255} to be 255.

By Algorithm 2, the hidden data can be extracted successfully, and the original cover image recovered losslessly.

3. OPTIMAL RECURSIVE EXTENSION

Some experiments were conducted with Algorithms 1 and 2 by using the block size of 8×8, and good results were obtained, as will be reported later. However, it was observed in the experimental process that further divisions of certain blocks of 4×4 by the same technique of 4-way divisions C_1 through C_4 as mentioned previously will yield even better data hiding rates. This inspired our idea of extending the previously-proposed *non-recursive* method into a recursive version by considering *hierarchical* block divisions from a selected *initial* block size down to the size of 2×2. Optimal algorithms for implementing such an idea were designed and will be presented subsequently. Before

that, we present first a corresponding *recursive* version of the step of looking-ahead estimation of the data hiding capacity (step 2 of Algorithm 1) as an algorithm in the following.

Algorithm 3 Recursive looking-ahead estimation of the data hiding capacity of a block.
Input: a block image B with a size of at least 4×4.
Output: the maximum data hiding capacity D in B and a key component k for B.
Step 1: (*Initialization*) set $k = \varepsilon$ (an empty string).
Step 2: Divide B by the four cases C_1 through C_4.
Step 3: Compute the data hiding capacity D_i for each case C_i, i = 1-4, in the following way.
3.1 Compute D_i for i = 1, 2, and 3 as follows,
A.1 for each sub-block I_j in C_i, perform the following steps:
(1) generate the histogram h of I_j;
(2) find the peak in h and its location x_0;
(3) sum up $h(x_0 - 1)$ and $h(x_0 + 1)$ as d_j;
A.2 sum up all d_j's to get the data hiding capacity D_i for C_i;
A.3 set the corresponding component k_i of k for C_i as i.
3.2 If the sizes of the four sub-blocks in C_4 are all 2×2 (the minimum size for which no more block division is performed), then compute D_4 in the following way:
B.1 compute the data hiding capacity D_1' of each sub-block in C_4 by the process (A.1) described above;
B.2 compute D_4 as $D_4 = D_1' + D_1' + D_1' + D_1'$ (four times D_1');
B.3 set the corresponding component k_4 of k for C_4 as $k_4 = 4 <1, 1, 1, 1>$;
otherwise, compute D_4 in the following way:
B.1′ for each sub-block B_i' of C_4, go to step 3 to perform the step itself *recursively* to compute the data hiding capacity D_i' and the component k_i' of k for B_i';
B.2′ compute D_4 as $D_4 = D_1' + D_2' + D_3' + D_4'$;
B.3′ set the corresponding component k_4 of k for C_4 as $k_4 = 4 <k_1', k_2', k_3', k_4'>$.
3.3 Find the largest value D_m among D_1 through D_4, and take $D = D_m$ and $k = k_m$ as the output and exit.

Algorithm 4 Data embedding (a recursive version).
Input: a cover image I with blocks of size $n \times n$ and a message M in bit string form to be embedded.
Output: a stego-image I', a key K, and a location map S_M.
Step 1: For each block B in I, perform the following steps.
1.1 (*Looking-ahead estimation of data hiding capacities*) compute the data hiding capacity D and a *key component* k for B by Algorithm 3.
1.2 If the first-level component ℓ in k is 1, 2, or 3, divide B by the corresponding way C_ℓ into sub-blocks and perform data embedding as described in step 4 of Algorithm 1 for each sub-block; otherwise, continue.
1.3 If k is not empty (not an empty string ε), take out the next-level (the ith-level) quad-tree components $<k_i^1, k_i^2, k_i^3, k_i^4>$ sequentially from k; otherwise, go to

step 2.

1.4 For each k_i^j, j = 1-4, perform the following operations:

if $k_i^j = \ell$ =1, 2, or 3, then

divide B by the way of C_ℓ into sub-blocks and perform data embedding as described in step 4 of Algorithm 1 for each sub-block;

otherwise, with k_i^j in the form 4 $<k_{i+1}^1, k_{i+1}^2, k_{i+1}^3, k_{i+1}^4>$, go to step 1.4 to perform the step itself *recursively* for each k_{i+1}^j, j = 1-4.

Step 2: (*Key generation*) concatenate in order the key components k's of all the blocks in I generated in step 1 to form a sequence $k_1k_2k_3$ …, and take the final image and $k_1k_2k_3$ …, as the output stego-image I' and key K, respectively.

Algorithm 5 Data extraction (corresponding to Algorithm 4).

Input: a stego-image I' with blocks of size $n \times n$, the key K, and the location map S_M.

Output: the message data M hidden in I' and the cover image I.

Step 1: (*Initialization*) set $M = \varepsilon$ (an empty string).

Step 2: For each block B in I', perform the following steps.

2.1 Take from K sequentially the key component k which corresponds to B.

2.2 If the first-level component ℓ in k is 1, 2, or 3, then divide B by the way of C_ℓ into sub-blocks and perform data extraction (including the operation of appending the extracted bit to M) as described in step 3 of Algorithm 2 for each sub-block; otherwise, continue.

2.3 If k is not empty (*i.e.*, not an empty string ε), take out the next-level (the i-level) quad-tree components $<k_i^1, k_i^2, k_i^3, k_i^4>$ sequentially from k; otherwise, stop this algorithm.

2.4 For each of k_i^j, j = 1-4, perform the following operation:

if $k_i^j = \ell$ = 1, 2, or 3, then

divide B by the way of C_ℓ into sub-blocks and perform data extraction (including the operation of appending the extracted bit to M) as described in step 3 of Algorithm 2 for each sub-block;

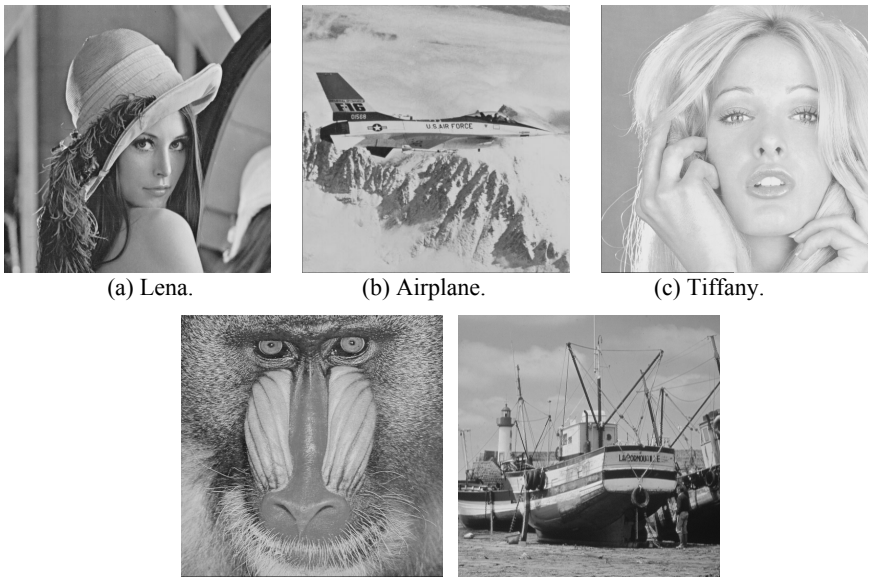
otherwise, with k_i^j in the form 4 $<k_{i+1}^1, k_{i+1}^2, k_{i+1}^3, k_{i+1}^4>$, go to step 2.4 to perform the step itself recursively for each of k_{i+1}^j.

As mentioned previously, the proposed hierarchical block division process can be expressed as a construction of a tree structure, on which we explore to the *bottom tree level* and return the best composition of block divisions *backward* for each initial block in the proposed recursive data-hiding capacity estimation algorithm (Algorithm 3). This is equivalent to an optimal tree search and the estimated data hiding capacity may be regarded as an optimal solution for the proposed recursive data embedding process (Algorithm 4). Additionally, the non-recursive algorithms (Algorithms 1 and 2) may be considered as special cases of the recursive ones (Algorithms 4 and 5) because we explore the tree structure only to the second level in the non-recursive algorithms.

4. EXPERIMENTAL RESULTS

A series of experiments have been conducted to test the proposed algorithms on images, some of which are shown in Fig. 6. Each test image shown in the figure is a

grayscale one of size 512 × 512 with the gray values ranging from 0 through 255. For the purpose of comparing our results with those of other methods, we have implemented the algorithms of Ni *et al.* [10], Hwang *et al.* [12], and Kuo *et al.* [13]. We show first the statistics of some experimental results of our implementation of Kuo *et al.* [13] in Table 1. Each test image was divided into blocks of sizes from 256 × 256 down to 2 × 2. As can be observed, as the number of blocks in each test image increases, both the PSNR of the resulting stego-image and the data hiding capacity increase until a bottleneck at the block size of 8 × 8 is encountered. This phenomenon has also been mentioned previously in section 2 and illustrated by Figs. 1-3.



(a) Lena. (b) Airplane. (c) Tiffany.

(d) Baboon. (e) Boat.

Fig. 6. Test images used in experiments.

Table 1. Statistics of experimental results of the method of Kuo *et al.* [13] showing a bottle-neck of data-hiding-rate increasing at block size 8 × 8.

512 × 512 (No. of blocks)	256 × 256 (4)	128 × 128 (16)	64 × 64 (64)	32 × 32 (256)	16 × 16 (1024)	8 × 8 (4096)	4 × 4 (16384)	2 × 2 (65536)
Lena (PSNR of stego-image)	8996 (48.21)	12645 (48.25)	18422 (48.31)	24182 (48.39)	29537 (48.52)	33931 (48.73)	33349 (49.15)	23196 (50.13)
Airplane (PSNR of stego-image)	25555 (48.38)	30296 (48.43)	35109 (48.54)	41780 (48.68)	46945 (48.86)	50920 (49.15)	49690 (49.71)	38718 (50.94)
Baboon (PSNR of stego-image)	6044 (48.18)	7731 (48.20)	9474 (48.23)	11454 (48.27)	12426 (48.33)	12909 (48.46)	12042 (48.78)	8309 (49.65)
Tiffany (PSNR of stego-image)	16409 (48.28)	21617 (48.35)	28875 (48.45)	35272 (48.55)	40449 (48.69)	44490 (48.92)	43022 (49.38)	30901 (50.39)
Boat (PSNR of stego-image)	12931 (48.25)	18897 (48.32)	26875 (48.41)	33562 (48.50)	38382 (48.62)	40379 (48.81)	38234 (49.19)	25955 (50.13)

Table 2. Comparison of results of proposed method and related lossless histogram-shifting data hiding methods.

Method		Kuo *et al.*'s method		Proposed non-recursive method (initial block size of 8 × 8)		Proposed non-recursive method (initial block size 4 × 4)		Proposed recursive method (initial block size 8 × 8)	
Block size		Blocks of size 8 × 8		Combination of blocks with sizes 8 × 8, 8 × 4, 4 × 8 and 4 × 4		Combination of blocks with sizes 4 × 4, 4 × 2, 2 × 4 and 2 × 2		Combination of blocks with sizes 8 × 8, 4 × 8, 8 × 4, 4 × 4, 4 × 2, 2 × 4 and 2 × 2	
Hiding capacity	Quality of stego-image	Bits	PSNR	Bits	PSNR	Bits	PSNR	Bits	PSNR
Lena (512 × 512)		33931	48.73	41257	48.90	42166	49.36	45342	49.13
Airplane (512 × 512)		50920	49.15	59397	49.37	58375	49.96	63787	49.64
Baboon (512 × 512)		12909	48.46	17455	48.57	17650	48.92	19863	48.71
Tiffany (512 × 512)		44490	48.92	52297	49.10	51900	49.60	56191	49.33
Boat (512 × 512)		40379	48.81	46833	48.95	45487	49.37	49752	49.12

Table 3. Comparison of results of proposed non-recursive algorithms and related methods.

method		Ni *et al.*		Hwang *et al.*		Kuo *et al.*		Proposed non-recursive method (initial block size of 8×8)	
Image or block size		Image of size 512 × 512		Image of size 512 × 512		Blocks of size 8 × 8		Combination of blocks with sizes 8 × 8, 8 × 4, 4 × 8 and 4 × 4	
Hiding capacity	Quality of stego-image	Bits	PSNR	Bits	PSNR	Bits	PSNR	Bits	PSNR
Lena (512×512)		5409	48.22	5304	48.18	33931	48.73	41257	48.90
Airplane (512×512)		18700	48.45	17502	48.28	50920	49.15	59397	49.37
Baboon (512×512)		5932	48.28	5793	48.18	12909	48.46	17455	48.57
Tiffany (512×512)		10153	48.30	9942	48.21	44490	48.92	52297	49.10
Boat (512×512)		10546	49.25	9709	48.21	40379	48.81	46833	48.95

The proposed method uses the combination of different block divisions C_1 through C_4 to break this bottleneck. First, the non-recursive Algorithms 1 and 2 were utilized, and the results are listed in the middle-left part of Table 2 which indeed shows the breakthrough effect. Specifically, the initial block size of 8 × 8 was used first in the experiments. As can be seen, both the data hiding capacity and the PSNR value resulting from each of the five test images are raised, compared with those yielded by Kuo *et al.* [13]. For example, for the image of Lena, the bottleneck of the data hiding capacity is 33931 bits with the PSNR 48.73 dB, and the resulting capacity of the proposed algorithms is 41257 bits with the PSNR 48.90 dB.

To see further the effectiveness of Algorithms 1 and 2, we tried the use of a smaller initial block size of 4 × 4 and the results are again good enough to break the bottleneck for all the five test images, as can be seen from the middle-right part of Table 2. For example, the bottleneck of 33931 bits with the PSNR 48.73dB this time is improved to become 42166 bits with the PSNR 49.36 dB for the image of Lena. Both results of the data hiding capacity and the PSNR value are even higher than the previous results of 41257 bits and 48.90dB using the initial block size of 8 × 8. This means that hiding more data does not always degrade the stego-image quality, a phenomenon contrary to the general understanding that more hidden data will worsen the stego-image quality, as mentioned previously.

However, it is noted that the above superiority of the data hiding capability of using initial blocks of size 4 × 4 over 8 × 8 is *not* always true, as can be seen from the data for some other tested images. Specifically, for the images of Airplane, Tiffany, and Boat, the data hiding capacities of using initial blocks of size 4 × 4 are lower than those of using initial blocks of size 8 × 8. However, this phenomenon indicates that it is possible to obtain even better data hiding capacities and PSNR values for the initial block size of 8 × 8 by dividing the blocks *through two levels*, down to 4 × 4 and then 2 × 2. This is the reason why we extended the non-recursive algorithms, Algorithms 1 and 2, into the recursive ones, Algorithms 3 through 5. But instead of using all 4 × 4 and 2 × 2 blocks, we use divisions of C_1 through C_4 to gain even better effects (*i.e.*, we did not always cut a square block into 4 smaller square ones using C_4 only).

The experimental results of applying the recursive algorithms, Algorithms 3 through 5, to the five text images are shown in the rightmost part of Table 2, in which, for example, the hiding capacity of image Lena now is 45342 bits which is higher than those yielded by Algorithms 1 and 2 (41257 and 42166, respectively, as mentioned previously).

For a more complete comparison with related methods, including Ni *et al.* [10] and Hwang *et al.* [12] in addition to Kuo *et al.* [13], we show the experimental results of our implementations of them in Table 3. As can be seen, the proposed recursive algorithms produce better results in both the aspects of hiding capacity and image quality. For a clearer visualization of the data in Table 3, we use bar-chart diagrams as well to show as examples the results of images Lena and Airplane in Figs. 7 and 8, respectively.

To see the relationship between image characteristics and the performances of the previously-mentioned histogram-shifting methods (including the proposed one), we observed first that the image Baboon in Fig. 6 (d) consists of many types of texture with less smooth regions; the images Lena and boat are with less textures; and the images Airplane and Tiffany have large smooth regions. Next, it is not difficult to see that different types of test images will yield different data hiding capacities and stego-image qualities, because the performances of histogram-shifting methods are related to the characteristics of test images. More specifically, an image with more smooth regions usually includes more pixels with similar grayscale values, and so yields highly possibly a large peak value in the histogram. This means that the "bin" at the peak has a large volume of pixels, resulting in a large capacity for data hiding. On the contrary, a test image with more textured regions will provide a smaller data hiding capacity. The experimental results described previously in Tables 1-3 can be seen to support this phenomenon.

In more detail, by Ni's method [10], the volume of the bin at the peak in the histogram of a cover image will be equal to the data hiding capacity; therefore, images with

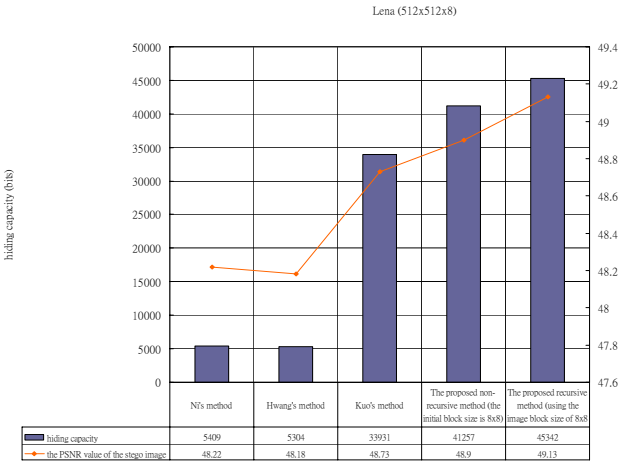


Fig. 7. Illustration of comparison of hiding capacity versus image distortion in mentioned methods using image Lena (including non-recursive and recursive versions of proposed method).

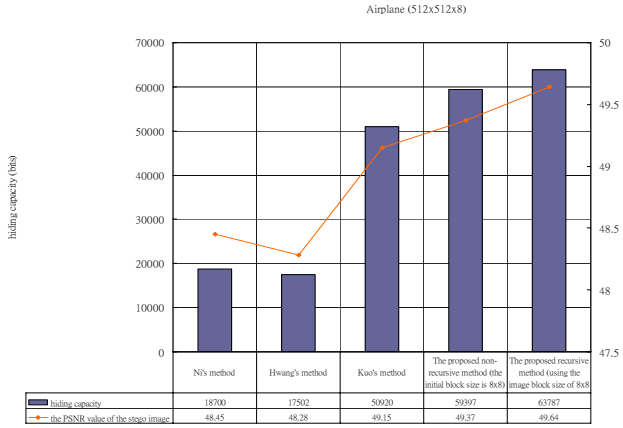


Fig. 8. Illustration of comparison of hiding capacity versus image distortion in mentioned methods using image Airplane (including non-recursive and recursive versions of proposed method).

more smooth regions such as Tiffany and Airplane will provide larger hiding capacities than those with less smooth regions such as Baboon as shown in Column 2 of Table 3. Likewise, by the methods of Hwang *et al.* [12] and Kuo *et al.* [13], both sides of the peak in the histogram of a cover image are used for data hiding, so images characterized as smooth will provide larger data hiding capacities than those with textured regions, as can be seen from Columns 3 and 4 of Table 3 where the data hiding capacity yielded by Tiffany is larger than that by Lena, for example.

As to the proposed method, including the non-recursive version and the recursive one, the same phenomenon is observed as well in Columns 3-5 of Table 2. Specifically, the image Airplane provides the largest data hiding capacity, whereas the image Baboon, as expected, provides the smallest hiding capacity, among the five given images shown

in Fig. 6.

It was found in this study that a lower bound for the PSNR value of the stego-image yielded by the proposed non-recursive Algorithms 1 and 2 can be estimated deterministically, and the estimation is conducted here. As an example, consider one block of the 4096 ones of size 8 × 8 in a cover image of size 512 × 512. In the extreme case, the block is not divided further, that is, the way of C_1 is applied to the block. Denote the histogram of the given block as h and the location of its peak as x_0. In the worst case, the value of $h(x_0)$ is 1, and except the gray value x_0 at the peak, the other 63 pixels' gray values in the block will be incremented or decremented by 1 after the process of data embedding is performed. In other words, at least one pixel's gray value is kept unchanged in each block. Therefore, the gray values of at least 4096 pixels will be kept the same in the stego-image. The mean square error (MSE) of the stego-image and the corresponding lower bound of the PSNR value thus can be computed by Eqs. (1)-(2), respectively, in the following, where N is the total number of pixels of the cover image, and $n \times n$ is the size of the blocks:

$$MSE = \frac{1}{N}\sum_{i=1}^{N-\frac{N}{n\times n}} (\pm 1)^2 = \frac{1}{N}\times(N-\frac{N}{n\times n})$$
$$= \frac{1}{512\times 512}\times[(512\times 512)-\frac{512\times 512}{8\times 8}] = 0.984\,, \tag{1}$$

$$PSNR = 10\times \log_{10}(255\times 255/MSE) = 48.2 \text{ dB}. \tag{2}$$

The PSNR is quite high. If the initial block size is changed from 8 × 8 to 4 × 4, then the number of blocks become 4096 × 4 = 16384 and the corresponding PSNR value can be computed similarly as follows,

$$MSE = \frac{1}{N}\sum_{i=1}^{N-\frac{N}{n\times n}} (\pm 1)^2 = \frac{1}{N}\times(N-\frac{N}{n\times n})$$
$$= \frac{1}{512\times 512}\times[(512\times 512)-\frac{512\times 512}{4\times 4}] = 0.937\,, \tag{3}$$

$$PSNR = 10\times \log_{10}(255\times 255/MSE) = 48.41. \tag{4}$$

which is even higher. It can be deduced from the above computation process that a smaller size of blocks produces a higher lower-bound value of the PSNR. As a summary, the above analysis indicates that the proposed non-recursive algorithms, Algorithms 1 and 2, yield good stego-image qualities.

In the proposed method, we have to generate a location map for image recovery. To see the size of a location map, since the sizes of such maps generated from the five test images in Figs. 6 (a)-(e) are all 0, we tested an additional image "Gray" (Coleen Gray) as shown in Fig. 9 and counted the numbers of pixels with grayscale values 0 and 255 in it

to be 23 and 0, respectively. Accordingly, the size of the location map may be easily computed to be 17 × (23 + 0) = 391 where the number 17 denotes the total bits used for expressing the coordinates of *X* (needing 8 bits for the image width 246 in the *X*-direction) and *Y* (needing 9 bits for the image length 340 in the *Y*-direction) for an "overflow/underflow" pixel, while the number 23 in "(23 + 0)" means that the number of pixels with gray value 0 is 23 and that with gray value 255 is 0. And the data embedding capacity yielded by the proposed method is 15399. Comparatively, the size of the location map is very small with respect to the data embedding capacity.

Fig. 9. Image "Gray" with size 246 × 340.

5. CONCLUSIONS

A lossless data hiding method based on histogram shifting has been proposed, which not only embeds large-volume data into cover images, but also produces stego-images with high qualities by using a strategy of hierarchical block division. The bottleneck of data-hiding-rate increasing at the block size of 8 × 8 found in existing methods is broken by the proposed non-recursive algorithms. And the proposed recursive versions of the algorithms enhance the performance further both in the data hiding capacity and the PSNR value, which result from the proposed scheme of recursive looking-ahead estimation of the data hiding capacity. The estimation process is a kind of optimal tree search under the quad-tree structure constructed by the hierarchical block division scheme, and so yields an optimal data hiding result under the tree structure. The experimental results show the effectiveness of the proposed method. Future researches may be directed to investigating more block division types for further improvement on the data hiding capacity, applying the histogram shifting technique to other information hiding applications, reducing the key size, eliminating the use to the location map, *etc.*

REFERENCES

1. W. Bender, D. Gruhl, N. Morimoto, and A. Lu, "Techniques for data hiding," *IBM Systems Journal*, Vol. 35, 1996, pp. 313-336.
2. D. C. Wu and W. H. Tsai, "A steganographic method for images by pixel-value differencing," *Pattern Recognition Letters*, Vol. 24, 2003, pp. 1613-1626.

3. I. C. Lin, Y. B. Lin, and C. M. Wang, "Hiding data in spatial domain images with distortion tolerance," *Computer Standards and Interfaces*, Vol. 31, 2009, pp. 458-464.
4. K. J. Kim, J. H. Jung, and K. Y. Yoo, "A high capacity data hiding method using PVD and LSB," *Computer Science and Software Engineering*, Vol. 3, 2008, pp. 876-879.
5. J. M. Barton, "Method and apparatus for embedding authentication information within digital data," U. S. Patent 5646997, 1997.
6. M. U. Celik, G. Sharma, A. M. Tekalp, and E. Saber, "Reversible data hiding," in *Proceedings of International Conference on Image Processing*, Vol. 1, 2002, pp. 71-76.
7. 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.
8. J. Tian, "Reversible data embedding using a difference expansion," *IEEE Transactions on Circuits Systems and Video Technology*, Vol. 13, 2003, pp. 890-896.
9. H. J. Kim, V. Sachnev, Y. Q. Shi, J. Nam, and H. G. Choo, "A novel difference expansion transform for reversible data embedding," *IEEE Transactions on Information Forensics and Security*, Vol. 3, 2008, pp. 456-465.
10. 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.
11. M. Fallahpour and M. H. Sedaaghi, "High capacity lossless data hiding based on histogram modification," *IEICE Electronics Express*, Vol. 4, 2007, pp. 205-210.
12. J. H. Hwang, J. W. Kim, and J. K. Choi, "A reversible watermarking based on histogram shifting," in *Proceedings of the 5th International Workshop on Digital Watermarking*, LNCS 4283, 2006, pp. 348-361.
13. W. C. Kuo, D. J. Jiang, and Y. C. Huang, "A reversible data hiding scheme based on block division," in *Proceedings of International Congress on Image and Signal Processing*, Vol. 1, 2008, pp. 365-369.
14. J. Fridrich, M. Goljan, and R. Du, "Invertible authentication watermark for JPEG images," in *Proceedings of International Conference on Information Technology: Coding and Computing*, 2001, pp. 223-227.
15. J. Fridrich, M. Goljan, and R. Du, "Lossless data embedding for all image formats," in *Proceedings of SPIE Photonics West*, *Electronic Imaging*, *Security and Watermarking of Multimedia Contents*, Vol. 4675, 2002, pp. 572-583.
16. J. L. Liu, D. C. Lou, M. C. Chang, and H. K. Tso, "A robust watermarking scheme using self-reference image," *Computer Standards and Interfaces*, Vol. 28, 2006, pp. 356-367.
17. C. C. Lin, W. L. Tai, and C. C. Chang, "Multilevel reversible data hiding based on histogram modification of difference images," *Pattern Recognition*, Vol. 41, 2008, pp. 3582-3591.

Che-Wei Lee (李哲瑋) receives the B.S. degree in Civil Engineering and the M.S. degree in Electrical Engineering from National Cheng Kung University, Tainan, Taiwan, in 2002 and 2005, respectively. He is a Ph.D. student in the Department of Computer Science at National Chiao Tung University since 2005. His research interests include information hiding, image processing, and video technologies.

Wen-Hsiang Tsai (蔡文祥) received the B.S. degree in Eelectrical Engineering from National Taiwan University, Taipei, Taiwan, R.O.C., the M.S. degree from Brown University, Providence, RI, and the Ph.D. degree from Purdue University, West Lafayette, IN.

Currently, he is a Chair Professor with the Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan, R.O.C., and was the President of Asia University, Taichung, Taiwan. So far he has published 141 journal papers and 226 conference papers. His research interests include image processing, computer vision, information security, and autonomous vehicle applications.