

Data Hiding in Binary Images with Distortion-Minimizing Capabilities by Optimal Block Pattern Coding and Dynamic Programming Techniques*

I-Shi LEE[†], Member and Wen-Hsiang TSAI^{†a)}, Nonmember

SUMMARY A new method for data hiding in binary images based on block pattern coding and dynamic programming with distortion-minimizing capabilities is proposed. Up to three message data bits can be embedded into each 2×2 block in an input image by changing the block's pixel pattern into another, which represents the value of the message data bits as a code according to a block pattern encoding table. And extraction of hidden message data is accomplished by block pattern decoding. To minimize the resulting image distortion, two optimization techniques are proposed. The first is to use multiple block pattern encoding tables, from which an optimal one is selected specifically for each input image, and the second is to use a dynamic programming algorithm to divide the message data into bit segments for optimal embedding in a sense of minimizing the number of binary bit flippings. Accordingly, not only more data bits can be embedded in an image block on the average, but the resulting image distortion is also reduced in an optimal way. Experimental results are also included to show the effectiveness of the proposed approach.

key words: data hiding, binary image, block pattern encoding, dynamic programming, image distortion minimization

1. Introduction

Many data hiding techniques have been proposed for a variety of digital image applications in recent years [1]–[6]. Most of the techniques were proposed for color and grayscale images because the pixels in such images take a wide range of values and so are easier to use for data hiding. One simple approach is to use the least-significant-bit (LSB) replacement technique [6]. However, data hiding in *binary images* is a more challenging work. The major reason is that such images have drastic contrasts and so it is easier for humans' eyes to find out pixel value changes in them after data hiding works. For binary images, Wu et al. [7] embedded message data in specific image blocks that are selected with higher *flippability scores* by pattern matching. Manipulated

flippable pixels on image region boundaries are then used to embed significant amounts of data without causing noticeable artifacts. Pan et al. [8] changed pixel values in binary image blocks, mapped block contents into given message data, and used a secret key and a weight matrix to protect the hidden data. In an image block of size $m \times n$, the scheme can conceal up to $\lfloor \log_2(m \times n + 1) \rfloor$ bits of data by changing at most two bits in the image block. Tseng and Pan [9] proposed a technique to alter the value of an image bit into a new one identical to that of a neighboring bit, yielding a better hiding effect within a binary image. Koch and Zhao [10] embedded a bit 0 or 1 in a binary image block by changing the number of black pixels in the block to be larger or smaller than that of white ones, respectively. In Refs. [11], [12], message data are concealed into dithered images by maneuvering dithering patterns. Tzeng and Tsai [13] encoded the edge features of a binary image into 4×4 block patterns, and authenticated the image by pattern matching. Tzeng and Tsai [14] also proposed a new feature, *surrounding edge count*, for measuring the structural randomness in a 3×3 image block, and defined *pixel embeddability* from the viewpoint of minimizing image distortion. Accordingly, embeddable image pixels suitable for hiding message data can be selected.

In a binary image, there are two distinct pixel values, 0 and 1, corresponding to *black* and *white* pixels, respectively. When data are embedded into a binary image, some image pixels used for data hiding will be changed from black to white or reversely. The pixel value changes will be called *bit flippings* in the sequel. To embed more data, more bit flippings may be conducted; however, the quality of the resulting image will also get worse. The bit flipping rates of most data hiding methods for binary images are about 50%. We propose in this paper a new data hiding method which has the capability to conceal up to *three* data bits in a 2×2 block, resulting in bit flipping rates lower than 50%. The method can thus be used to embed more data. This is achieved by a block pattern coding technique. On the other hand, while it is desirable to embed more data, the resulting image quality should be maintained in the mean time. For this purpose, two optimization techniques are proposed. The first is to use multiple block pattern encoding tables, from which an optimal one is selected for each input image. The second technique is to use a dynamic programming algorithm to divide the message data stream into appropriate bit segments

Manuscript received October 5, 2006.

Manuscript revised February 4, 2007.

[†]The author is with the Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan 30010. He is also with the Department of Management Information at Northern Taiwan Institute of Science and Technology.

^{††}The author was with the Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan 30010. He is now with the Department of Computer Science and Information Engineering, Asia University, Wufeng, Taichung, Taiwan 41354.

*This work was supported partially by the NSC project, Advanced Technologies and Applications for Next Generation Information Networks (II) – Sub-project 5: Network Security, with Project No. NSC93-2752-E-009-006-PAE.

a) E-mail: whtsai@cis.nctu.edu.tw

DOI: 10.1093/ietisy/e90-d.8.1142

for optimal data embedding in the image blocks in the sense of minimizing the number of bit flippings. As a result, the proposed method can achieve the goals of both increasing the embedded data volume and reducing the resulting image distortion. Furthermore, the method can be used to extract embedded data without referencing the original image.

In the remainder of this paper, the proposed method is described in Sect. 2. Some experimental results are shown in Sect. 3, followed by a conclusion in Sect. 4.

2. Proposed Data Hiding Method

The proposed method is designed to hide message data behind binary images in random fashions controlled by secret keys. The method consists of a data embedding process and a data extraction one. In this section, the ideas behind the proposed method are presented first, followed by the detailed descriptions of the proposed processes. In the sequel, the image into which a message is hidden will be called a *cover image*, and the result a *stego-image*.

2.1 Block Pattern Encoding for Data Embedding

In order to embed a message into a binary cover image, every 2×2 block of the image is regarded as a *pattern* with a 4-bit *binary value* in which each bit of 0 corresponds to a black pixel and each bit of 1 a white one. An illustration is shown in Fig. 1. Therefore, possible binary values of a block pattern are 0000_2 through 1111_2 , where 0000_2 means an entirely black block and 1111_2 an entirely white one.

The proposed data embedding process is based on the use of a *block pattern encoding table* which maps each block pattern into a certain *code* with each code being one, two, or three bits of the message data to be hidden. And data embedding is accomplished by changing block bit values so that the corresponding codes of the resulting block patterns become just the input message data to be embedded. A block pattern encoding table designed for use in this study is shown in Table 1. The idea behind the design of this table is described as follows. It is emphasized, by the way, that such a table is just one of the many possible ones usable for data hiding, and the proposed data embedding process will choose from them an *optimal* one for each specific input image, as described later.

The number of possible patterns in a 2×2 block are 16. This number is much larger than that needed to represent


2x2 block pattern	Corresponding binary value				
<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>b_1</td> <td>b_2</td> </tr> <tr> <td>b_3</td> <td>b_4</td> </tr> </table>	b_1	b_2	b_3	b_4	$b_1b_2b_3b_4$
b_1	b_2				
b_3	b_4				
	0110				

















Fig. 1 Illustration of block patterns and corresponding binary values.

the two message bits ‘0’ and ‘1,’ so we use multiple block patterns to represent a single message data value, allowing the possibility of choosing among the patterns an optimal one to replace the original image block in the data embedding process and thus reducing the image distortion in the resulting block. Furthermore, we wish to embed more data in a block, and for this purpose we use a block pattern to represent more than one bit of message data.

For example, we may use either the block pattern $t_1 = 1101_2$ or the pattern $t_2 = 1110_2$ to represent the two-bit message value $s = 00_2$. In this way, when we want to embed, for example, the message value $s = 00_2$ into a block B with value $v = 0110_2$, we have the *two* alternative block patterns $t_1 = 1101_2$ and $t_2 = 1110_2$, instead of the conventional case of just *one*, from which we can choose $t_2 = 1110_2$ to replace the value $v = 0110_2$ of block B , resulting in a smaller distortion of just a 1-bit error. Contrastively, if only one choice, say, $t_1 = 1101_2$ is allowed, then an error of 3 bits will result, which means a larger distortion in the resulting block. It is such an allowance of multiple choices for block pattern replacement that results in less image distortion in the proposed method.

More generally, we group in this study the 16 possible block patterns in a 2×2 block B into distinct sets according to the *entropy* values of the block patterns, where an entropy value E of a block pattern P is defined as follows:

Table 1 A block pattern encoding table proposed in this study.

Type	Block pattern	Entropy value	Corresponding binary value	Encoded message data
A		0	1111	1
B1		0.811	1110	00
B2		0.811	1101	00
B3		0.811	1011	01
B4		0.811	0111	01
C1		1	0011	011
C2		1	0101	011
C3		1	1010	010
C4		1	1100	010
D1		1	0110	100
D2		1	1001	101
E1		0.811	0001	11
E2		0.811	0010	11
E3		0.811	0100	10
E4		0.811	1000	10
F		0	0000	0

$$E = - \sum_k p_k \log_2 p_k = -p_0 \log_2 p_0 - p_1 \log_2 p_1$$

with p_0 and p_1 being the occurrence probability values of black and white pixels appearing in P computed as one-fourth of their numbers in P , respectively (note that there are four pixels in P). A pattern P in a set with a higher entropy value E is presumably more random in its black and white arrangement, and so is more suitable for hiding more message data without causing a noticeable change. There are three possible entropy values 0, 0.811, and 1 in a 2×2 block by the above definition, so we divide the 16 possible block patterns into three sets. The first set with the entropy value 0 has two distinct block patterns, one being entirely white and the other entirely black. They are denoted as A and F in Table 1 and are used to represent (i.e., to *encode*) the message data of 1 and 0, respectively.

The second set with the entropy value 0.811 includes eight distinct block patterns, which can be classified into two classes, one class being with each pattern including one black pixel and three white ones, and the other class with each pattern including three black pixels and one white one. The first class, denoted as B in Table 1, includes four block patterns, and we use two block patterns of them to encode the message value 00_2 , and the other two of them to encode the message value 01_2 . To decide which two patterns should be selected to encode a message value, we adopt the *mismatch reduction criterion* of making the two selected patterns, after being superimposed, less different in the number of mismatching pixel values. We use the four block patterns of the other class, denoted as E in Table 1, to encode the message values 10_2 and 11_2 in a similar way.

The last set with the entropy value 1 has six distinct block patterns. So far, we have completed, by the use of 10 patterns, the encoding of all possible one-bit and two-bit messages. So the remaining six patterns in the 16 ones may be used to encode three-bit message values. But six patterns are not enough to encode all the eight possible three-bit message values, so we only take care of some of them, following the aforementioned mismatch reduction criterion. Specifically, we use two block patterns to encode each of the two 3-bit message values 011_2 and 010_2 , and finally the last two patterns to encode the message values 100_2 and 101_2 , respectively. The six patterns are denoted as C1 through C4, and D1 and D2 in Table 1.

2.2 Sketch of Proposed Idea of Data Hiding

In the proposed data embedding process, the more data we embed in a 2×2 block, the worse the resulting image quality becomes. Therefore, we must control the number of destructed pixels in a block to reduce the resulting image distortion. The idea of the proposed data embedding process is sketched as four major steps in the following, which includes two folds of distortion minimization.

- (1) *Dividing the input image into blocks*: We divide the input image into 2×2 blocks with every two neigh-

boring blocks being separated by a 1-pixel-wide line, as shown in Fig. 2. The 1-pixel-wide band around each 2×2 block is said to be the *neighborhood* of the block.

- (2) *Selecting a random list of embeddable blocks for data embedding*: We then use a secret key K as well as a random number generator f to select randomly a sequential list of *embeddable blocks*. A block B is said to be *embeddable* in this study if the following two conditions are satisfied: (a) the neighborhood of B is not entirely black or white, (b) B has not been selected for data embedding yet. The way we adopt to generate the random list of embeddable blocks is as follows: (a) concatenate all blocks obtained in Step (1) above in sequence; (b) use K and f to generate sequentially a random number $f(K)$, divide it by the total number of blocks, and take the remainder as a block number, denoted by N ; (c) check block N to see if it is an embeddable block; if not, then perform the same process until an embeddable block is obtained; (d) append the obtained embeddable block to the end of the desired random list; (e) stop the process when a sufficient number of embeddable blocks for data embedding are obtained.
- (3) *Using multiple block pattern encoding tables for the first-fold distortion reduction*: We generate all possible block pattern encoding tables and select an *optimal one* for use in the data embedding process, in the sense of introducing *the least distortion*.
- (4) *Applying optimal search techniques for the second-fold distortion reduction*: Finally we apply the dynamic programming technique to segment the input message data stream *optimally* into a series of codes and embed them in the input image, according to a *cost function* designed in advance for measuring the degree of the pattern change in each image block. This reduces the resulting distortion further in a *global sense*.

2.3 Use of Multiple Block Pattern Encoding Tables

The first distortion-reduction technique using multiple block pattern encoding tables, as mentioned previously in the third major step of the proposed data embedding process, is based on the idea that a single block pattern encoding table will not be suitable for every input binary image. If a binary image is destroyed very seriously in the data embedding process using a specific block pattern encoding table like Table 1, it will be necessary to use another table with other combi-

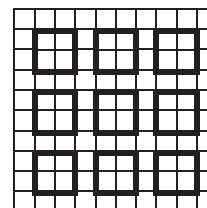


Fig. 2 Division of input image into 2×2 blocks with separating lines (grids with bold boundaries are 2×2 blocks for data embedding).

nations of block patterns and encoded hidden message data. For example, assume that a binary message value $v = 101_2$ is to be embedded into a sequence of three randomly selected image blocks with patterns 0000, 0100, and 1111 by the use of Table 1. The data embedding process, as illustrated in Fig. 3 (a), will select *optimally* the block pattern type D2 = 1001, which encodes the three-bit message value $v = 101_2$, to replace the first selected block with pattern 0000, resulting in a flipping of two bits. However, if we construct a new table by replacing the encoded message data of type A in Table 1 with those of type F and replacing those of types B1 through B4 with those of types E1 through E4, respectively, then the use of the new table to hide the message value $v = 101_2$ will result in *no* bit flipping because now we can, as illustrated in Fig. 3 (b), select in sequence *optimally* the new pattern type F = 0000 (encoding the message data of 1) and the new pattern type E3 = 0100 (encoding the message data of 01₂) to encode together the message data $v = 101_2$. This means that the proposed idea of *adaptive table generations and selections* for use in data embedding indeed helps distortion reduction. More generally, by enumerating all possible ways for exchanging the encoded message data of certain types in Table 1 with those of the other types in the following way, we can get 128 distinct block pattern encoding tables (numbered from 0 to 127) for selection in the data embedding process to minimize the distortion:

- exchange the message data “0” with “1”;
- exchange the message data “00” with “01”;
- exchange the message data “10” with “11”;
- exchange the message data “010” with “011”;
- exchange the message data “100” with “101”;
- exchange the message data “00” and “01” with “10” and “11,” respectively;
- exchange the message data “010” and “011” with

“100” and “101,” respectively.

2.4 Proposed Distortion-Minimizing Cost Function and Search Techniques for Optimal Solutions

The cost function proposed in this study for use in the proposed data embedding process to minimize image distortion is *the total number of bit flippings* in the resulting stego-image. In Table 1, block patterns can be used to encode one, two, or three message bits. Accordingly, when we hide a binary message value v , we have the three choices of embedding into a block the first one, two, or three bits in the prefix of v . To determine how many bits should be embedded, we *may* calculate first the cost function value for each of the three cases, and then replace the currently selected block with the block pattern which corresponds to the optimal case with the minimum cost function value. This method provides a quick way for data embedding. However, it is actually a *greedy search* and in general *does not* yield an optimal solution.

To see this, for example, for the previously-mentioned example in which the message value v of 101_2 is embedded in three selected blocks with patterns 0000, 0100, and 1111 according to Table 1, by the above-mentioned greedy search algorithm we first replace the block with pattern 0000 by the block pattern E3 = 0100 to embed two bits 10_2 . The computed cost function value is 1 because a bit is flipped here. This cost is a *local minimal one*. Next, we replace the block with pattern 0100 by the block pattern A = 1111 to embed the last bit 1 of v , and get a local minimal cost value of 3. The total cost value is then $1 + 3 = 4$. Now, if we do not use the greedy search algorithm at the beginning, and replace instead the first block with pattern 0000 by the block pattern D2 = 1001 to embed three bits 101_2 directly, then the total cost value will be reduced to 2 which is smaller than the previous one of 4. This shows that there indeed exists at least one solution better than that found by the greedy search method. Figure 4 illustrates the data embedding process for this example. This is also true for many other examples, as found in this study. And so, the search of an optimal solution is meaningful, for which the proposed method in this study is *dynamic programming*.

In the proposed dynamic programming algorithm (abbreviated as DPA in the sequel), *edit distances* are defined to minimize the cost function, as described in the following. Assume that the input message data to be hidden are in the form of an n -bit string S_1 with $S_1[i]$ denoting its i th bit. Also, let the list of n randomly selected 2×2 blocks for data embedding be expressed as another string S_2 with $S_2[i]$ denoting its i th block. For convenience, let $S_k[i \sim j]$ denote a substring of S_k with bits or blocks $S_k[i]$ through $S_k[j]$, where $k = 1$ or 2 and $i, j = 1, 2, \dots, n$.

Only one type of edit operation, namely, *replacement*, is used in the proposed DPA to specify the image block replacement operations involving S_1 and S_2 in the proposed data embedding process. The edit distance between S_1 and

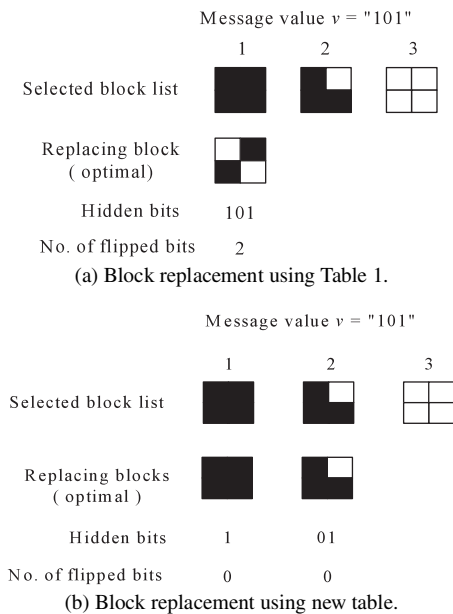


Fig. 3 An example of proposed data embedding process.

S_2 is defined, according to the previous discussions, as the minimum cost to transform S_2 into S_1 by editing operations according to a certain block pattern encoding table used in the embedding process. Let C be an $n \times n$ cost matrix with its element $C[i, j]$ denoting the minimum cost to transform a substring $S_2[j \sim n]$ of S_2 into a substring $S_1[i \sim n]$ of S_1 . Then $C[1, 1]$ is the value of the minimum cost to transform S_2 into S_1 . Also, let RC be a replacement cost function with each of its element $RC(\ell, i, j)$ denoting the cost for replacing the j th block $S_2[j]$ of S_2 with the block pattern which encodes the initial ℓ bits of the substring $S_1[i \sim n]$ of S_1 , where $\ell = 1, 2$, or 3 . By the above definitions, the value $C[i, j]$ is recursively just the minimum of all possible values of $RC(\ell, i, j) + C[i + \ell, j + 1]$, where $\ell = 1, 2$ or 3 . Then, according to the dynamic programming technique, the minimum cost may be computed by the following algorithm.

Algorithm 1. Computing cost for minimizing distortion in proposed data embedding process by DPA.

Input: (1) an n -bit message data string S_1 ; (2) a string S_2 of n randomly selected blocks; (3) a block pattern encoding table T ; (4) an $n \times n$ cost matrix $C[i, j]$, for $i, j = 1, 2, \dots, n$; (5) an $n \times n$ type matrix R with its element $R[i, j]$ being used for recording the block pattern in T used for replacing $S_2[j]$ in calculating $C[i, j]$; and (6) an $n \times n$ segmentation matrix N with its element $N[i, j]$ being used for recording the number of initial bits in the prefix of $S_1[i \sim n]$ used in calculating $C[i, j]$.

Output: $C[i, j]$, $R[i, j]$, and $N[i, j]$ for all $i, j = 1, 2, \dots, n$.

Steps:

1. Set all $C[i, j]$ initially to be ∞ for all $i, j = 1, 2, \dots, n$.
2. Starting from $i = n$ and $j = n$, for each pair of (i, j) with $i, j = 1, 2, \dots, n$ perform the following steps.
 - 2.1 If $C[i, j]$ is equal to ∞ , continue the next step; else calculate the next $C[i, j]$.
 - 2.2 Take $C[i, j]$ to be the minimum of the three cost values, $RC(1, i, j) + C[i + 1, j + 1]$, $RC(2, i, j) +$

$C[i + 2, j + 1]$, $RC(3, i, j) + C[i + 3, j + 1]$; and record the corresponding number of the processed initial bits in the prefix of $S_1[i \sim n]$ in $N[i, j]$, and the corresponding type of the used block pattern of T in $R[i, j]$.

In the above algorithm, the initial bits in the prefix of $S_1[i \sim n]$ and the used block pattern type in each recursive step are recorded respectively in matrices N and R , which will be used in the data embedding process as described in the next algorithm.

Algorithm 2. Proposed data embedding process using block pattern encoding tables and DPA.

Input: a binary image I , a message data string S_1 with n bits, a secret key K as well as a random number generator f , and 128 block pattern encoding tables.

Output: a stego-image S , an optimal block pattern encoding table T_{opt} , the number L_{opt} of a list of selected blocks for data embedding, and a minimal total cost C_{min} .

Steps:

1. Get a list of n embeddable 2×2 blocks from the input image I in a way described in Sect. 2.2 using the input secret key K and the random number generator f .
2. For each block pattern encoding table T among the input 128 ones, calculate the cost matrix $C[i, j]$, the type matrix $R[i, j]$, and the segmentation matrix $N[i, j]$ for all $i, j = 1, 2, \dots, n$ using T by Algorithm 1.
3. Take the desired value of C_{min} to be the minimum of the 128 values of $C[1, 1]$ computed in the last step, and the desired value of T_{opt} to be the corresponding block pattern encoding table used in computing C_{min} .
4. Use Table T_{opt} to embed the message data string S_1 into I to get the desired stego-image S in a way specified by the type matrix R and the segmentation matrix N corresponding to C_{min} . Take the desired value of L_{opt} to be the number of the selected blocks for embedding the message data.

2.5 Proposed Data Recovery Process

The goal of the proposed data recovery process is to extract the embedded message data from a stego-image. Before the proposed data extraction process is started, an involved block pattern encoding table is simplified in advance into

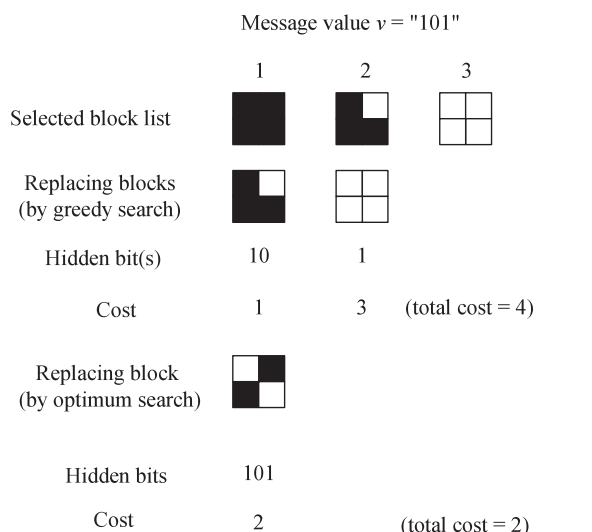


Fig. 4 An example of proposed data embedding process.

Table 2 An extraction table (table number $T = 0$).

Corresponding binary value of block pattern	Encoded message data	Corresponding binary value of block pattern	Encoded message data
1111	1	0111	01
1110	00	0110	100
1101	00	0101	011
1100	010	0100	10
1011	01	0011	011
1010	010	0010	11
1001	101	0001	11
1000	10	0000	0

an *extraction table*. For example, the extraction table corresponding to Table 1 is as shown in Table 2. This form of extraction table makes it easier to carry out the data extraction work, as described in the following.

Algorithm 3. Message data recovery process.

Input: a stego-image I' presumably including a message bit stream S ; the secret key K as well as the random number generator f used in the data embedding process; the table T_{opt} ; and the number L_{opt} of the selected block list used in the data embedding process.

Output: the message bit stream S .

Steps:

1. Extract a list of 2×2 embeddable blocks from the stego-image I' by the use of the secret key K , the random number generator f , and the number L_{opt} of the selected block list by the way described in Sect. 2.2.
2. For each 2×2 embeddable block P in I' , compute the corresponding binary value v of the block pattern P , and decode v as the extracted result by looking v up in the extraction table T'_{opt} generated from Table T_{opt} to get the corresponding encoded message data bits embedded in the block.
3. Concatenate all the extracted data bits in sequence as the desired message bit stream S .

3. Experimental Results

Some experimental results of applying the proposed method are shown in Figs. 5, 6, and 7. Figures 5 (a), 6 (a) and 7 (a) show three binary cover images of the sizes 687×534 ,

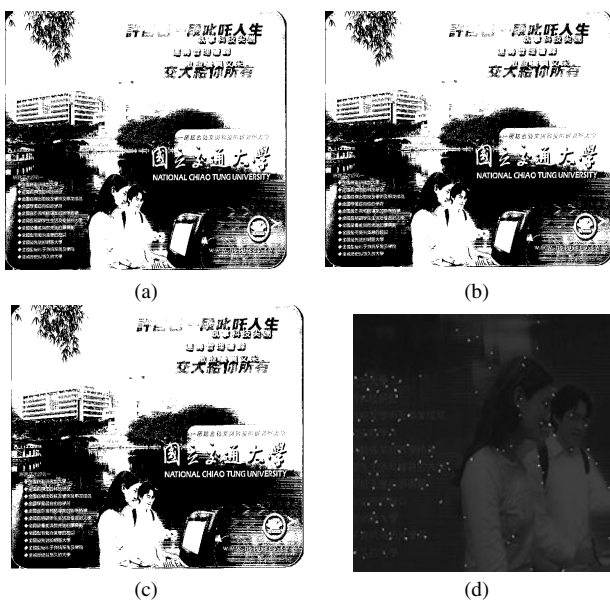


Fig. 5 Input binary images, output stego-images with message data, and difference images. (a) Binary image “NCTU”. (b) Stego-image using greedy search and optimal encoding table. (c) Stego-image using DPA and optimal encoding table. (d) An enlarged part of difference image between (a) and (c) in which the white spots are difference pixels.

512×512 , and 2320×3408 , respectively. Two streams of message data were generated by a random fashion. One is a stream of 2432 bits, which was embedded into each of the binary images shown in Figs. 5 (a), and 7 (a). The other is 992-bit long, which was embedded into the binary image shown in Fig. 6 (a). The stego-images obtained by embedding the message data using the greedy search algorithm and the optimal encoding table among the 128 ones are shown in Figs. 5 (b), 6 (b) and 7 (b), respectively. And the stego-images after embedding the message using the DPA and the optimal encoding table among the 128 ones are shown in Figs. 5 (c), 6 (c) and 7 (c), respectively. Figure 6 (d) shows the difference between Figs. 6 (a) and 6 (c) in terms of white pixels. And Figs. 5 (d), and 7 (d) show similarly enlarged versions of parts of the differences between Figs. 5 (a) and 7 (a) and Figs. 5 (c) and 7 (c), respectively, for better inspection effects. Note that the original input images are included in Figs. 5 (d), 6 (d) and 7 (d) in gray values as the backgrounds to show more clearly the difference spots.

Table 3 shows the statistical data of the stego-images of Figs. 5 (b), 5 (c), 6 (b), 6 (c), 7 (b), and 7 (c), in which we list the selected encoding table numbers, the numbers of used blocks, the minimum cost values, the message data length and the distortion rate. The minimum cost values show that the DPA using an optimal encoding table among the 128 ones is the best, the greedy search algorithm using an optimal encoding table among the 128 ones is the next, and the greedy search algorithm using just a fixed encoding table is

Table 3 Statistics of three stego-images processed by proposed algorithms.

Stego-image	Algorithm	Table number	No. of used blocks	Cost value	Message data length	Distortion rate
NCTU	Greedy search (using a fixed table)	0	1528	1153	2432	0.47
	Greedy search (using optimal table among 128 ones)	16	1526	1115		0.46
	DPA (using optimal table among 128 ones)	26	1418	954		0.39
Lena	Greedy search (using a fixed table)	0	621	431	992	0.43
	Greedy search (using optimal table among 128 ones)	30	637	401		0.40
	DPA (using optimal encoding table among 128 ones)	41	582	369		0.37
Patent	Greedy search (using a fixed table)	0	1439	1037	2432	0.43
	Greedy search (using optimal encoding table among 128 ones)	70	1530	1007		0.41
	DPA (using optimal table among 128 ones)	24	1433	924		0.38



Fig. 6 Input binary images, output stego-images with message data, and the difference images. (a) Binary image “Lena”. (b) The stego-image using the greedy search algorithm and the optimal encoding table. (c) The stego-image using the DPA and the optimal encoding table. (d) The difference image between (a) and (c) in which the white spots are difference pixels.

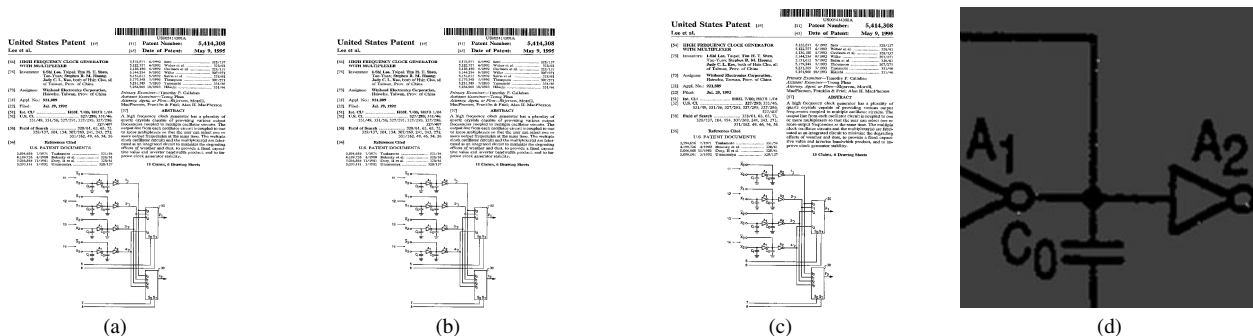


Fig. 7 Input binary images, output stego-images with message data, and difference images. (a) Binary image “Patent.” (b) Stego-image resulting from greedy search and optimal encoding table. (c) Stego-image using DPA and optimal encoding table. (d) An enlarged part of difference image between (a) and (c) in which the white spots are difference pixels.

the worst. For other images, similar results can be observed. For the images shown here, the average number of message data embedded in a block, using the DPA, is about 1.7 bits. And the distortion rate computed as the ratio of the number of bit flippings to the length of the message data, using the

DPA is in the range from 37% to 39%. Furthermore, we tested 19 images that are obtained from an image database of the USC, and the results are listed in Table 4. As shown there, the average number of message data embedded in a block, using the DPA, is about 1.939 bits. And the average

Table 4 Statistics of 19 stego-images processed by proposed DPA.

Image No. of USC	Table number	No. of used blocks	Cost value	Message data length	Distortion rate	Embedding density
4.2.03	22	482	369	992	0.37	2.06
4.2.03	66	1241	851	2342	0.36	1.89
4.2.03	26	498	353	*992	0.36	1.99
4.2.06	66	516	351	992	0.35	1.92
5.2.08	57	500	356	992	0.36	1.98
5.2.09	41	527	352	992	0.35	1.88
5.2.10	6	523	336	992	0.34	1.90
7.1.01	24	508	361	992	0.36	1.95
7.1.03	70	521	346	992	0.35	1.90
7.1.04	70	507	362	992	0.36	1.96
7.1.05	6	530	343	992	0.35	1.87
7.1.06	8	525	349	992	0.35	1.89
7.1.07	18	533	345	992	0.35	1.86
7.1.08	66	508	353	992	0.36	1.95
7.1.09	57	507	353	992	0.36	1.96
7.1.10	18	514	352	992	0.35	1.93
boat.512	57	507	360	992	0.36	1.96
elain.512	6	499	358	992	0.36	1.99
house	6	495	355	992	0.36	2.00
average					0.356	1.939

distortion rate using the DPA is 35.6%, which is smaller than 50% yielded by most existing data hiding methods for binary images. The image with number 4.2.03 in Table 4 was used for three times to embed secret message data of two different lengths and three different contents (*992 and 992 mean two streams of message data with identical lengths of 992 bits but with different contents).

4. Discussions

A novel optimal method for hiding message data into binary images with a distortion minimization effect and a larger data embedding capability has been proposed. An optimal block pattern encoding table is chosen from 128 alternative ones for use in the proposed data embedding process to minimize distortion in the stego-image. The method can minimize further the distortion using the dynamic programming technique and can embed up to three bits in a 2×2 image block. By the proposed method, not only more data can be embedded in a binary image, but also the distortion rate of the stego-image can be effectively reduced. The space and time complexities of the proposed DPA are both of the order of n^2 . It may cost more time to embed a long secret code. But one can alternatively use the other proposed method, the greedy search algorithm, that takes only linear computation time and still minimize distortion in the stego-image in a suboptimal way. On the other hand, in certain applications there is no need of real-time processing and optimality in data embedding volumes is the main concern. Then the proposed method is applicable.

The proposed method is based on the use of 2×2 blocks. It may be extended by processing larger blocks because then, the number of block patterns which can be selected to encode messages will become larger as well, resulting in greater reductions of image distortions. However, there is a tradeoff here, i.e., the resulting data embedding ca-

capacity will decrease. Other future works may be directed to designing a better cost function from the perspective of the human visual system, imposing more constraints on the cost function to yield better image quality, and finding a better way to design encoding tables to reduce stego-image distortion further.

References

- [1] S. Katzenbeisser and F.A.P. Petitcolas, *Information Hiding Techniques for Steganography and Digital Watermarking*, Artech House, Boston, MA, 2000.
- [2] D. Kundur, "Energy allocation principles for high capacity data hiding," Proc. IEEE International Conf. on Image Processing, vol.1, pp.423–426, Vancouver, Canada, Sept. 2000.
- [3] S. Lin and D.J. Costello, Jr., *Error Control Coding: Fundamentals and Applications*, Prentice-Hall, Englewood Cliffs, NJ, 1983.
- [4] L.M. Marvel, J.C.G. Boncelet, and C.T. Retter, "Spread spectrum image steganography," IEEE Trans. Image Process., vol.8, no.8, pp.1075–1083, Aug. 1999.
- [5] M. Swanson, M. Kobayashi, and A. Tewfik, "Multimedia data-embedding and watermarking technologies," Proc. IEEE, vol.86, pp.1064–1088, 1998.
- [6] D.C. Wu and W.H. Tsai, "Spatial-domain image hiding using an image differencing," IEE Proc., Vis. Image Signal Process., vol.147, no.1, pp.29–37, 2000.
- [7] M. Wu, E. Tang, and B. Liu, "Data hiding in digital binary image," Proc. IEEE International Conference on Multimedia & Expo 2000 (ICME'00), New York, 2000.
- [8] H.K. Pan, Y.Y. Chen, and Y.C. Tseng, "A secure data hiding scheme for two-color images," IEEE ISCC 2000, pp.750–755, 2000.
- [9] Y.C. Tseng and H.K. Pan, "Secure and invisible data hiding in 2-color images," Proc. IEEE INFOCOM 2001 Conference on Computer Communications, no.1, pp.887–896, 2001.
- [10] E. Koch and J. Zhao, "Embedding robust labels into images for copyright protection," Proc. International Congress on Intellectual Property Rights for Specialized Information, Knowledge and New Techniques, pp.242–251, Munich, Germany, 1995.
- [11] K. Matsui and K. Tanaka, "Video-steganography: How to secretly embed a signature in a picture," IMA Intellectual Property Project, pp.187–205.
- [12] H.C. Wang, "Data hiding techniques for printed binary images," Proc. IEEE International Conf. on Information Technology: Coding and Computing, pp.55–59, Las Vegas, NV, April 2001.
- [13] C.H. Tzeng and W.H. Tsai, "A new technique for authentication of image/video for multimedia applications," Proc. ACM Multimedia 2001 Workshops-Multimedia and Security: New Challenges, pp.23–26, Ottawa, Ontario, Canada, Oct. 2001.
- [14] C.H. Tzeng and W.H. Tsai, "Hiding authenticable general digital information behind binary images with reduced distortion," Proc. 2nd Workshop on Digital Archives Technologies, Taipei, Taiwan, July 2003.



I-Shi Lee was born in Taipei, Taiwan, R.O.C., in 1961. He received the B.S. degree in electronic engineering from National Taiwan University of Science and Technology, Taipei, Taiwan, Republic of China in 1987, the M.S. degree in the Department of Computer Science and Information Science at National Chiao Tung University in 1989. In 1992, he joined the Department of Management Information at Northern Taiwan Institute of Science and Technology and acted as a lecturer from 1992 to now. He

also works in the Computer Vision Laboratory of the Department of Computer and Information Science at National Chiao Tung University as a research assistant from August 1999, and is currently working toward his Ph.D. degree there. His recent research interests include pattern recognition, watermarking, and image hiding.



Wen-Hsiang Tsai was born in Tainan, Taiwan, Republic of China (R.O.C.) in May 10, 1951. He received the B.S. degree in electrical engineering from National Taiwan University, Taipei, Taiwan, Republic of China in 1973, the M.S. degree in electrical engineering (with major in computer science) from Brown University, Providence, Rhode Island, U.S.A. in 1977, and the Ph.D. degree in electrical engineering (with major in computer engineering) from Purdue University, West Lafayette, Indiana, U.S.A.

in 1979. Dr. Tsai joined the faculty of National Chiao Tung University, Hsinchu, Taiwan in November 1979, and stays there until 2004. He is currently a Professor in the Department of Computer Science and Information Science, Asia University and the President of the University. Professor Tsai has been an associate professor of the Department of Computer Engineering (now called Department of Computer Science and Information Engineering) and the Acting Director of the Institute of Computer Engineering. In 1984, he joined the Department of Computer and Information Science and acted as the Department Head from 1984 through 1988. He has also been the Associate Director of the Microelectronics and Information System Research Center from 1984 through 1987, the Dean of General Affairs from 1995 to 1996, the Dean of Academic Affairs of the University from 1999 to 2001, and the Vice President of the National Chiao Tung University from 2001 to 2004. He has served as the Chairman of the Chinese Image Processing and Pattern Recognition Society at Taiwan from 1999 to 2000. Outside the campus, Professor Tsai has served as a Consultant to several major research institutions in Taiwan. He has acted as the Coordinator of Computer Science in National Science Council, and a member of the Counselor Committee of the Institute of Information Science of Academia Sinica in Taipei. He has been the Editor of several academic journals, including Computer Quarterly (now Journal of Computers), Proceedings of the National Science Council, Journal of the Chinese Engineers, International Journal of Pattern Recognition and Artificial Intelligence, Journal of Information Science and Engineering, and Pattern Recognition. He was the Editor-in-Chief of Journal of Information Science and Engineering from 1998 through 2000. Professor Tsai's major research interests include image processing, pattern recognition, computer vision, virtual reality, and information copyright and security protection. So far he has published 257 academic papers, including 107 journal papers and 150 conference papers. He is also granted 6 R.O.C. or U.S.A. patents. Dr. Tsai has supervised the thesis studies of 26 Ph.D. students and 101 master students. Professor Tsai has received many awards, including one Distinguished Research Award, four Outstanding Research Awards, and three Special Researcher Awards, all of the National Science Council in 1987 through 2004. He also received an Academic Award from the Ministry of Education. He was the recipient of the 13th Annual Best Paper Award of the Pattern Recognition Society of the U.S.A. He was elected as an Outstanding Talent of Information Science and Technology of the R.O.C. in 1986, received the Best Teacher Award of the Ministry of Education in 1989, and was the recipient of the Distinguished Official Award of the Ministry of Education in 1994. He was the recipient of many Academic Paper Awards made by several academic societies, including two by the Computer Society of the Republic of China in 1989, and thirteen by the Chinese Image Processing and Pattern Recognition Society. He has also received in the past twenty years eleven Ph.D. and Master's Thesis Supervision Awards from the Acer Long-Term Foundation, the Xerox Taiwan Company, the Federation of Image Product Companies, the Electrical Engineers Society at Taiwan, and the Information Science Society at Taiwan. Dr. Tsai is a senior member of the IEEE of the U.S.A., and a member of the Chinese Image Processing and Pattern Recognition Society, the Medical Engineering Society of the Republic of China, and the International Chinese Computer Society.