Pattern Recognition 42 (2009) 1604 – 1611

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

Data hiding in grayscale images by dynamic programming based on a human visual model☆

I-Shi Lee[a,1], Wen-Hsiang Tsai[a,b,*]

[a]Department of Computer Science and Information Engineering, National Chiao Tung University, Hsinchu 30010, Taiwan
[b]Department of Information Communication, Asia University, Taichung 41354, Taiwan

ARTICLE INFO

Article history:
Received 7 September 2006
Received in revised form 22 June 2008
Accepted 11 January 2009

Keywords:
Data hiding
Grayscale image
Human visual system
Block pattern encoding
Dynamic programming

ABSTRACT

A new method for data hiding in grayscale images based on a human vision model with distortion-minimizing capabilities is proposed. Each of the eight bit planes of an input grayscale image is viewed as a binary image, into which message data are embedded horizontally. Two optimization techniques, namely, block pattern coding and dynamic programming, are proposed for image distortion minimization. Experimental results show good performs of the proposed method.

1. Introduction

Data hiding in images is a useful technique for secret communication. Many data hiding techniques have been proposed recently [1–3]. A common approach is least-significant-bit (LSB) replacement, which embeds message data in the LSB planes of an image. The image into which a message is hidden is called a *cover image*, and the result a *stego-image*. Wang et al. [4] embedded a binary image in the fifth LSB plane of a cover image using a genetic algorithm and a local pixel adjustment method to lower the distortion in the stego-image. Chang et al. [5] used dynamic programming to obtain an optimal solution for the LSB substitution method. Chan and Cheng [6,7] presented an optimal pixel adjustment process to improve the quality of the stego-image acquired by Wang's schemes. Thien and Lin [8] embedded data in images digit by digit using a modulus function, which improves LSB substitution not only in eliminating false contours but also in reducing image distortion. Lee and Chen [9] applied variable-sized LSB insertion to estimate the maximum embedding capacity by a human visual system (HVS) property, and to maintain image fidelity by removing false contours in smooth image regions. Liu et al. [10] presented a novel bit plane-wise data hiding scheme using variable-depth LSB substitution and employed post-processing to eliminate the resulting noticeable artifacts.

Most of the above methods lack consideration of using precise human visual models in improving the data hiding effect. Instead, Wu and Tsai [11] presented a method based on the HVS by modifying quantization scales according to variation insensitivity from smooth to contrastive to improve stego-image quality. And Lie and Chang [12] presented an adjusted LSB technique with the number of LSBs adapting to the pixels of different grayscales.

On the other hand, some steganalysis techniques were developed to detect secret messages among stego-images. Lyu and Farid [13] developed a universal blind detection scheme to detect hidden messages in stego-images, which uses wavelet-like decomposition to build higher-order statistical models of natural images and adopts the support vector machine as an optimal classifier to separate stego-images from cover images. The method demonstrates good performance on JPEG images and the selected statistics is rich enough to detect hidden data in the results yielded by a very wide range of steganographic methods. In addition, to detect data hidden in LSBs in the spatial domain, it is observed that the basic LSB substitution method changes pixel values only between $2i$ and $2i+1$ in the i-th bit plane of the pixel value. This leads to an effective

☆This work was supported partially by the NSC Advanced Technologies and Applications for Next Generation Information Networks (II)—Sub-Project 5: Network Security, Project no. NSC-96-2752-E-009-006-PAE and partially by the NSC Project NSC96-2422-H-009-001.

* Corresponding author at: Department of Computer Science and Information Engineering, National Chiao Tung University, Hsinchu 30010, Taiwan.
Tel.: +886 3 5728368; fax: +886 3 5734935.
E-mail addresses: gis87809@cis.nctu.edu.tw (I.-S. Lee), whtsai@cis.nctu.edu.tw (W.-H. Tsai).
[1] Also with Department of Management Information at Northern Taiwan Institute of Science and Technology.

0031-3203/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.patcog.2009.01.014

steganalytic technique, the RS method proposed by Fridrich et al. [14], which not only can expose the presence of secret data but also can estimate the length of the embedded data.

In this study we propose a method to embed data into a grayscale image, based on the use of a new HVS model to estimate the number of usable bits of each pixel in the cover image. Furthermore, a block pattern encoding method is proposed to embed up to three data bits in a 2×2 block of the bit planes without yielding visible image quality degrading. This is achieved by using two optimization techniques. The first technique utilizes multiple block pattern encoding tables, from which an optimal one is chosen for each input image; and the second technique uses dynamic programming to divide the message data stream into appropriate bit segments for optimal data bit embedding in the image blocks to minimize a cost function. The proposed method can extract embedded data without referencing the original image.

In the remainder of this paper, we introduce the idea behind the proposed method in Section 2. In Section 3, we describe the adopted HVS model and the corresponding cost function. In Section 4, the proposed data hiding method is described. The corresponding data recovery process is proposed in Section 5. Some experimental results are given in Section 6, followed by discussions and conclusions in Section 7.

2. Embedding data in bit planes of grayscale images

Eight bits represent a pixel's intensity in a grayscale image. The *bit plane* formed by the same bit of each pixel in the grayscale image is a binary image. Fig. 1 shows the eight bit planes of each of three 128×128 grayscale images. The LSB plane bp_0 is almost fully randomized. If a message is embedded in bp_0, the result will appear almost unaltered. On the contrary, random noise is less in a more-significant-bit plane. The most-significant-bit plane bp_7 contains almost no noise, and data cannot be embedded easily in it without causing significant visual changes. Embedding data into bit planes in the order of bp_0, $bp_1,\ldots,bp_7$ is called *horizontal data hiding*, contrastive with traditional *vertical data hiding* which embeds data into the bits $b_7, b_6,\ldots,b_0$ of each pixel in the order of b_0 through b_7, where b_0 is the LSB of the pixel. Comparatively, horizontal data hiding can reduce more distortion in the stego-image, as revealed in the results of this study.

On the other hand, embedding data directly in bit planes will cause visible damages to the edges in the bit planes. To overcome this difficulty, in this study we design a new cost function which considers certain perception characteristics of the HVS, and adopt a method proposed in Lee and Tsai [15] for data embedding. Each bit plane is regarded to have a different weight in its capability for data hiding, and the new cost function is designed accordingly to measure the degree of distortion resulting from pixel value changes, as described next.

3. Cost function for distortion measurement

Two HVS characteristics may be exploited for reducing image distortion in stego-images. First, human perception is more sensitive to grayscale changes in smooth areas than in texture areas in a grayscale image. Second, human perception is sensitive to relative luminance rather than absolute one. Designing the cost function for distortion measurement for data embedding must take these two characteristics into consideration.

For the first consideration, assume that a pixel P with grayscale value g is to be used to embed message data. Let MAX denote the maximum grayscale value, and MIN the minimum, in a 3×3 block with P as the center, which we call the *neighborhood* of P. Then, the maximum *between-pixel grayscale range* in this block is $\Delta = \mathrm{MAX}-\mathrm{MIN}$. To avoid a significant change of smoothness with respect to the neighborhood of P, the new grayscale value g' resulting from data embedding should be restricted in a certain range, which is taken to be $g \pm \Delta/2$ in this study. Then, we define a *maximum number D* of *data-embeddable bits* at P as

$$D = \lfloor log_2(\Delta/2) \rfloor = \lfloor (\log_2 \Delta) - 1 \rfloor = \lfloor \log_2(\mathrm{MAX} - \mathrm{MIN}) - 1 \rfloor. \quad (1)$$

For the second consideration, let f denote the luminance of a pixel P with grayscale value g where $1 \leqslant f \leqslant 100$. According to the Fechner law [16], the relative luminance property perceived by the HVS may be expressed as a contrast value c computed by $c = 50 \times \log_{10} f$ where $0 \leqslant c \leqslant 100$. Moreover, according to the Weber law [16], the maximum allowable change Δc of the contrast value c, according to the principle of "just noticeable difference (JND)" about the pixel's luminance change, is about 2. That is, if the luminance of a pixel is changed too much so that Δc is larger than 2, the change will be noticeable to the HVS. Accordingly, we can compute in another way a maximum number of data-embeddable bits in the 8 bits of a pixel's grayscale value, as described next.

First, we compute the maximum luminance change $(\Delta f)_{max}$ in accordance with the maximum allowable contrast change $(\Delta c)_{max} = 2$. With c being the contrast of pixel P, let c_{max} denote the maximum possible contrast value. Then, we have

$$2 = (\Delta c)_{max} = c_{max} - c = 50 \times \log_{10} f_{max} - 50 \times \log_{10} f = 50 \times \log_{10} \frac{f_{max}}{f},$$

which can be reduced to be

$$\frac{f_{max}}{f} = 10^{(2/50)} = 10^{0.04}.$$

So, the maximum allowable luminance change can be expressed as

$$(\Delta f)_{max} = f_{max} - f = \left(\frac{f_{max}}{f} - 1\right) f = (10^{0.04} - 1) \times f \approx 0.0965 \times f.$$

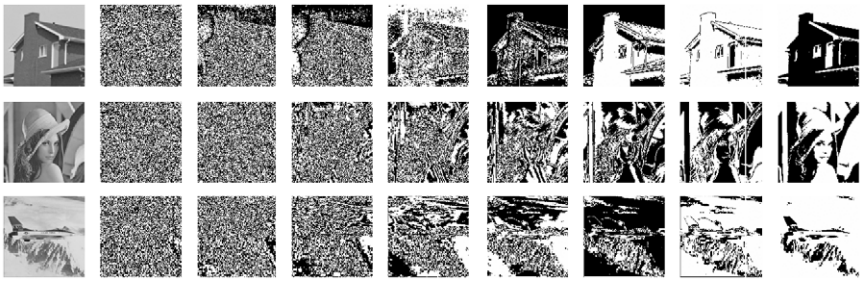

Fig. 1. Three grayscale images and their 8 corresponding bit planes (from left to right, original images, bp_0, bp_1, $bp_2,\ldots$, and bp_7, respectively).

And so we may impose the following constraint to the value of f:

$$(\Delta f)_{max}/f \leqslant 0.0965. \quad (2)$$

On the other hand, in a monochrome image the luminance f in the range of [1 100] is represented by the grayscale value g in the range [0 255], such that g may be computed by the linear mapping $g=(f-1)\times(255/99)\approx 2.576(f-1)$, or equivalently, the mapping $f\approx 0.3882g+1$. Hence, from Constraint (2), we can, after some derivations, get the following new constraint for grayscale changes:

$$0.0965 \geqslant (\Delta f)_{max}/f = (\Delta g)_{max}/(g+2.576), \quad (3)$$

where $(\Delta g)_{max}$, corresponding to $(\Delta f)_{max}$, denotes the maximum grayscale change in the pixel's neighborhood. That is, if Constraint (3) is set for data embedding, the changes of grayscales in the stego-image will not be detectable by human eyes.

Now, we discuss how many bits can be utilized for data embedding for each possible grayscale value g. If five bits of the pixel's grayscale are used for embedding message data, the maximum grayscale change at the pixel will be $(\Delta g)_{max}=2^5-1=31$. And according to Constraint (3), g must be larger than 319, which, however, is out of the grayscale range [0, 255]. This means that embedding five or more message bits into a pixel is impractical according to the principle of JND. As a result, bp_4, bp_5, bp_6, and bp_7 are not used for data embedding in this study. If four LSBs of g are changed, then $(\Delta g)_{max}=2^4-1=15$, and by Constraint (3) we get $g>153$. That is, when the constraint $g>153$ is satisfied, we can embed data into the four LSBs of g without causing a noticeable luminance change according to the principle of JND.

However, the binary value of 153 is 10011001_2. After the four LSBs of g are changed, the new value of g might become a value in the range of 10010000_2 through 10011000_2, which is smaller than 153, causing a violation of Constraint (3). Therefore, we must change the above constraint $g>153$ to be $g\geqslant 160$ where $160=10100000_2$ such that after any 4-bit data are embedded into the four LSBs of g, the resulting value g' of g will always be larger than 160, thus satisfying Constraint (3). In other words, to meet Constraint (3), only when a given pixel's grayscale g satisfies $g\geqslant 160$ can the four LSBs of g be replaced by 4-bit message data. Similarly, if three bits are changed, then $(\Delta g)_{max}=2^3-1=7$, and by Constraint (3) as well as a similar reasoning process, the constraint $g\geqslant 72$ should be satisfied, where $72=01001000_2$. If two bits are changed, the constraint $g\geqslant 32$ is required, where $32=00100000_2$. Finally, if one bit is changed, $g\geqslant 10$ is necessary, where $10=00001010_2$. In summary, we embed an appropriate number B of message bits in a pixel's grayscale g according to the following rule to satisfy the principle of JND:

if $g\geqslant 160$, then $B=4$;

if $g\geqslant 72$, then $B=3$;

if $g\geqslant 32$, then $B=2$;

if $g\geqslant 10$, then $B=1$;

otherwise, $B=0$. (4)

To combine the results of the above two considerations, it is not difficult to figure out that the maximum number of data-embeddable bits at a pixel should be taken to be $E=\min(D,B)$ where D and B are as specified in (1) and (4), respectively.

Let the grayscale value g of a pixel P in binary form be denoted as $g=(g_7\ g_6\ g_5\ g_4\ g_3\ g_2\ g_1\ g_0)_2$, and the *replacement cost* of g_i in the i-th bit plane be denoted as C_i where $0\leqslant i\leqslant 3$. According to the previous discussions, C_i is defined in this study as:

if $i\leqslant(E-1)$, then $C_i=8/2^{(E-1)-i}$; otherwise, $C_i=\infty$.

The above definition of cost function gives more penalties to replacements of more-significant-bits. In more detail, we have the following results:

if $E=4$, then $C_0=1$, $C_1=2$, $C_2=4$, $C_3=8$, and C_4 through $C_7=\infty$;

if $E=3$, then $C_0=2$, $C_1=4$, $C_2=8$, and C_3 through $C_7=\infty$;

if $E=2$, then $C_0=4$, $C_1=8$, and C_2 through $C_7=\infty$;

if $E=1$, then $C_0=8$, and C_1 through $C_7=\infty$;

if $E=0$, then C_0 through $C_7=\infty$.

4. Proposed horizontal data hiding method

The proposed method embeds two types of data into a cover image: *control data* and message data. The control data include the necessary information for use in the data recovery process. All data are embedded in bit planes bp_0 through bp_3 sequentially by the block pattern encoding method described next.

4.1. Block pattern encoding for data embedding

Every 2×2 block in a cover 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 1 a white one. A *block pattern encoding table* is constructed, 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. Data embedding is accomplished by changing the block bit values so that the corresponding code of the resulting block pattern becomes just some bits of the input message data to be embedded. A possible block pattern encoding table is shown in Table 1. Such a table is just one of the many possible ones which may be used for data hiding, and the proposed data embedding process chooses from them an *optimal one* for each specific input binary image.

Suppose that we want to embed one bit in a 2×2 block. The number of possible patterns in a 2×2 block are 16. This number is much larger than the required number of two to represent the two different message bits '0' and '1' in a block, so we may use *more than one* block pattern to represent a single message bit (0 or 1), allowing the possibility of choosing among the block patterns an *optimal one* to replace the original block in the data embedding process and thus reducing more distortion in the resulting block. On the other hand, we wish to embed more data in a block, not just a bit as just mentioned; and for this we use a block pattern to represent more than one bit. In short, we achieve both minimum-cost bit replacement and maximum-volume data embedding in the proposed method.

As an illustration, we may use either the block pattern $t_1=1011_2$ or the pattern $t_2=0111_2$ to represent the two-bit message value $s=01_2$. When we want to embed the message $s=01_2$ into a block B with value $v=1010_2$, we have *two alternative* block patterns $t_1=1011_2$ and $t_2=0111_2$ to choose to replace $v=1010_2$, instead of the conventional case of *just one*. And if we choose $t_1=1011_2$ to replace $v=1010_2$, then less distortion of just a 1-bit error (occurring at the LSB position) will result. Contrastively, if only one block pattern, say, $t_2=0111_2$ is available, then an error of three bits will result, causing more distortion in the resulting block. It is such an allowance of multiple choices for block pattern replacement that achieves more distortion reduction. By the way, the *bit error* is used here just for convenience of illustration; they in fact should be the *replacement costs* defined previously.

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

Type	Block pattern	Corresponding binary value	Encoded message data
0		1111	1
2		1110	00
4		1101	00
6		1011	01
8		0111	01
10		0011	011
12		0101	011
14		1010	010
1		0000	0
3		0001	11
5		0010	11
7		0100	10
9		1000	10
11		0110	100
13		1001	101
15		1100	010

4.2. Data embedding in binary images

The proposed data embedding process in binary bit-plane images consists of four major steps with two folds of distortion minimization, as described in the following:

1. *Computing bit costs for data embedding:* We calculate the replacement cost value for each bit in the image according to the cost function defined in Section 3.
2. *Dividing the input image into blocks:* We first divide each bit plane bp_i, $i = 0, 1, 2, 3$, into non-overlap 2×2 blocks with every two neighboring blocks separated by a 1-pixel-wide line of pixels in between, as shown in Fig. 2. And next, we select the first n "embeddable" blocks and concatenate them sequentially, where n is the length of the message data string to be embedded. A block is said to be *embeddable* if the replacement cost of any bit of the block is not *infinite*.
3. *Using multiple block pattern encoding tables to achieve 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.

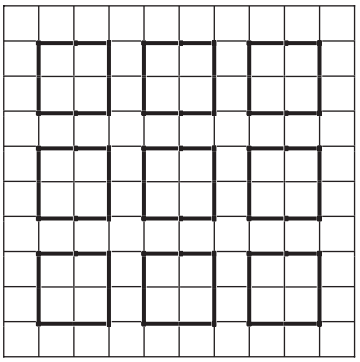

Fig. 2. Division of input image into 2×2 blocks with separating lines (grids with bold boundaries are 2×2 blocks for data embedding).

The reason is that a single block pattern encoding table will not be suitable for every input binary image; if an image is destroyed seriously after data embedding using a specific table like Table 1, it will be appropriate to use another table with other combinations of block patterns to encode the message data. Specifically, we exchange the encoded message data of certain types in Table 1 with those of the other types in the following way:

exchange message data "0" with message data "1";
exchange message data "00" with message data "01";
exchange message data "10" with message data "11";
exchange message data "010" with message data "011";
exchange message data "100" with message data "101";
exchange message data "00" and "01" with message data "10" and "11", respectively;
exchange message data "010" and "011" with message data "100" and "101", respectively.

By enumerating all possible cases in the above way, we can get 128 distinct tables (numbered from 0 to 127) for selection to minimize the distortion.

4. *Applying search techniques to achieve 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 the cost function proposed previously. This reduces the resulting distortion further in a global sense.

4.3. Search for optimal solutions

The *search cost* proposed for use in the search technique is the *total replacement cost*, computed as the summation of the replacement costs of all the bit changes in the replaced blocks. In Table 1, block patterns can be used to encode one, two, or three message bits. Accordingly, when we embed a binary message value v, we may choose to embed one, two, or three initial bits of v into a block. We can then calculate the cost for each case, and replace the selected block with the block pattern corresponding to the minimum cost. This process, quick for embedding message data sequentially and deterministically, is, however, just a *greedy search* algorithm and in general *does not* yield an optimal solution. An optimal algorithm based on *dynamic programming* is proposed next.

4.4. Dynamic programming for data embedding

In the proposed dynamic programming algorithm (abbreviated as DPA hereafter), *edit distances* are defined for search cost minimization. Assume that the input message data to be embedded are in the form of an n-bit string S_1 with $S_1[i]$ denoting its i-th bit. Also, let n 2×2 embeddable blocks be selected as a list in advance for data

embedding and 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 $k = 1, 2$ and $i, j = 1, 2, \ldots, n$.

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 S_2 is defined, according to the previous discussions, as the minimum total replacement cost to transform S_2 into S_1 by editing operations according to a certain block pattern encoding table. Let C be an $n{\times}n$ *cost matrix* with its element $C[j, i]$ denoting the minimum total replacement cost to transform a substring $S_2[j{\sim}m]$ of S_2 into a substring $S_1[i{\sim}n]$ of S_1, where $m \leqslant n$. Then $C[1,1]$ is the minimum total replacement cost to transform $S_2[1{\sim}m]$ into $S_1[1{\sim}n]$ (i.e., to transform the substring of S_2 into the entire string of S_1), where $1 \leqslant m \leqslant n$. Also, let RC be a cost function with each of its element RC(j, i, ℓ) denoting the minimum replacement 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 with $\ell = 1$, 2, or 3. We define RC$(j, i, \ell) = \infty$ if $i+\ell > n+1$.

By the above definitions, the value $C[j, i]$ is recursively just the minimum of all the possible values of RC(j, i, ℓ)+$C[j+1, i+\ell]$, where $\ell = 1$, 2 or 3. Also, we define $C[j, i] = 0$ if $i > n$ or $j > n$. Then, according to dynamic programming, the *minimum search cost* and its corresponding solution may be computed by the following algorithm.

Algorithm 1. Computing minimum search cost for minimizing distortion by the DPA.

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

1 Set all $C[j, i]$ initially to be ∞ for all $i, j = 1, 2, \ldots, n$.
2 Starting from $i = n$ and $j = n$, for each pair of (j, i) with $i, j = 1, 2, \ldots, n$, perform the following steps.
  2.1 If $C[j, i]$ is equal to ∞, continue the next step (Step 2.2); else increment i and j to calculate the next $C[j, i]$.
  3.1 Take $C[j, i]$ to be the minimum of the three replacement costs, RC$(j, i, 1)$+$C[j+1, i+1]$, RC$(j, i, 2)$+$C[j+1, i+2]$, and RC$(j, i, 3)$+$C[j+1, i+3]$; and record the corresponding number of the processed initial bits (1, 2, or 3) of $S_1[i{\sim}n]$ in $N[j, i]$, and the corresponding type of the used block pattern of T in $I[j, i]$.

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

Algorithm 2. Data embedding using block pattern encoding tables and the DPA.

Input: (1) a grayscale image G; (2) a secret message data string S_1 with n bits; (3) a control message data string S_c with m bits, including a *table number* T_{opt} (specifying the block pattern encoding table used) with seven bits, followed by a value L_{opt} (specifying the number of selected blocks used) with $m - 7$ bits; and (4) 128 block pattern encoding tables.
Output: a stego-image G'.
Steps:
1. Compute the cost of each bit of G as mentioned previously.
2. Get a list B_m of m 2×2 embeddable blocks sequentially from the bit planes bp_0 through bp_3 of G in order for embedding the m bits of S_c. Following B_m, get also a list B_n of n 2×2 embeddable blocks sequentially for the n bits of S_1. Let B_m and B_n also include the position information of each selected block.
3. For each block pattern encoding table T among the input 128 ones, with S_1, B_n, and T as input, apply Algorithm 1 to calculate the cost matrix $C[j, i]$, the type matrix $I[j, i]$, and the segmentation matrix $N[j, i]$ for all $i, j = 1, 2, \ldots, n$.
4. Find the minimum C_{min} of the 128 values of $C[1,1]$, and set T_{opt} to be the table number of the corresponding block pattern encoding table used in computing C_{min}.
5. Use the block pattern encoding table T_{opt}, the type matrix I_{min}, and the segmentation matrix N_{min} corresponding to C_{min}, and the position information of each block in B_n, to embed the string S_1 into bp_0 through bp_3 of G to get an *initial* stego-image G_i.
6. Set the value L_{opt} to be the number of the blocks used for embedding S_1 in the last step.
7. Using S_c (including T_{opt} and L_{opt}), B_m, and $T = 1$ as input, apply Algorithm 1 to calculate the cost matrix $C[j, i]$, the type matrix $I[j, i]$, and the segmentation matrix $N[j, i]$ for all $i, j = 1, 2, \ldots, m$.
8. Use the block pattern encoding table, Table 1, the type matrix I and the segmentation matrix N in the last step, and the position information of each block in B_m, to embed the substring S_c into bp_0 through bp_3 of G_i to get the *final* stego-image G'.

5. Proposed data recovery process

The goal of data recovery is to extract the embedded message data from a stego-image, as described in the following algorithm.

Algorithm 3. Message data recovery.

Input: a stego-image G' including a message bit stream S.
Output: the message bit stream S.
Steps:
1. Calculate the cost of every bit of G' as mentioned previously.
2. Get m 2×2 embeddable blocks sequentially from bp_0 through bp_3 of G' as a list L_m.
3. For each 2×2 block P of L_m, compute the binary value v corresponding to the block pattern, and decode v by looking v up in the block pattern encoding table 1 to get the corresponding encoded message data bits as the data recovery result of P.
4. Concatenate the initial m data bits extracted in the last step into a sequence as a desired control message data S_c.
5. Get the initial 7 data bits of S_c as T_{opt}, and the remaining $m - 7$ data bits of S_c as L_{opt}, which specify, respectively: (1) the optimal block pattern encoding table T_{opt} used in data embedding; and (2) the number of 2×2 blocks of G' used in embedding S_c in the bp_0 through bp_3 of G'.
6. Also, get L_{opt} 2×2 selected blocks sequentially from bp_0 through bp_3 of G' as a list L.
7. For each 2×2 block P of L, compute the binary value v corresponding to the block pattern, and decode v by looking v up in the block pattern encoding table T_{opt} to get the corresponding encoded message data bits as the data recovery result of P.
8. Concatenate all the data bits extracted in the last step into a sequence as the desired message bit stream S and exit.

For security consideration, we encrypt further the control message by a secret key before the data embedding process, and embed the result into bp_0 through bp_3 at bit positions randomly generated with a distinct secret key as well as a random number generator. The reverse process can be easily performed to get the original

control message. The same method is also applied to the message data to get a higher degree of data protection.

6. Experimental results

Figs. 3 and 4 illustrate some experimental results of applying the proposed method with randomly generated bit streams as the message data. The stego-image "House" of size 256×256 with a high PSNR value of 52.89 dB obtained by embedding 16440 bits (about 2 kB) message data using the DPA and the optimal block pattern encoding table among the 128 ones is shown at the right side of Fig. 3 The cover image is shown in the left side of Fig. 3 for comparison. The result shows that the proposed method can be applied to embed message data in a grayscale image and obtain a good-quality stego-image without noticeable artifacts in smooth regions.

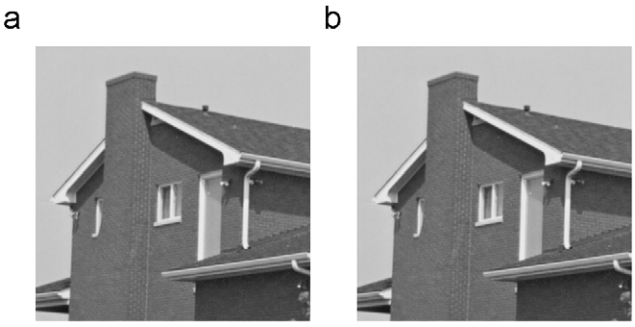

Fig. 3. A cover image "House" with the size of 256×256 and its stego-image with 16440-bit message data embedded. (a) The cover image. (b) The stego-image.

Fig. 4(b) illustrates three stego-images "House", "Lena" and "Jet," and their 8 corresponding bit planes. For comparison, the cover images in Fig. 1 are repeated in Fig. 4(a) here. These 128×128 stego-images were obtained by embedding 1000 bytes of message data. The PSNR values are 42.31, 45.65 and 44.88 dB, respectively. Compared with the cover images in Fig. 1 and their 8 corresponding bit planes, it can be seen that the stego-images retain most significant textures.

Table 2 summarizes the statistical data of the stego-image "Lena" using the DPA and the optimal encoding table with the message bit stream generated randomly. When the amount of the embedded data is smaller than 600B, the PSNR values in Table 2 are all larger than 50 dB. And the differences in the stego-images cannot be noticed by human eyes.

The proposed DPA method takes long computation time to obtain the optimal solution when the volume of the message data is large. If time is a major concern, then the greedy search method mentioned previously may be used. As a comparison, we list in Table 3 the run times spent by the proposed methods (DPA and greedy search) and two others on a PC with a 3.4 G Pentium 4 CPU for some grayscale images with two typical image sizes and three input message lengths. One of the two other methods is the simplest "1-LSB" which embeds message data in the LSB of each pixel. The other

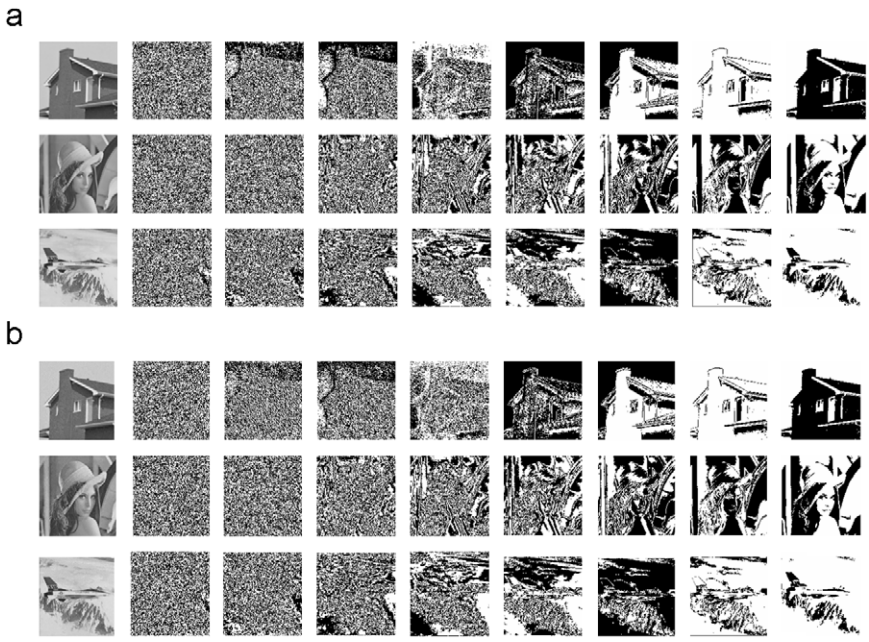

Fig. 4. Experimental results of three images. (a) The original images and their corresponding bit planes (repeated from Fig. 1). (b) The resulting three stego-images and their corresponding bit planes (from left to right, bp_0, bp_1, $bp_2, \ldots$, and bp_7, respectively).

Table 2
Statistics of stego-images yielded by DPA using optimal encoding table.

Stego-image	Message data length (bytes)	PSNR (dB)	Table number	No. of used blocks	Cost value	Embedded bit number per block
Lena (128×128)	100	55.48	4	428	1023	1.869
	200	55.27	8	779	1774	2.054
	400	55.38	57	1636	3243	1.956
	600	51.91	57	2297	5680	2.09
	800	49.28	57	3101	8355	2.064
	1000	45.65	8	3826	11699	2.091
	1200	43.06	57	4295	16248	2.235

is "Hide4PGP" whose program was downloaded from the website http://www.heinz-repp.onlinehome.de/Hide4PGP.htm. As can been see from the table, the DPA takes about a minute to embed a message of 200 bytes and more than 15 minutes to embed 1200-byte data, while all the other three methods takes little times to accomplish the works. Therefore, the DPA currently can only be used for non-real-time applications with the need of distortion reduction, though the greedy search method may be used as a suboptimal substitute of it.

We also conduct an additional comparison of image distortion caused by the above-mentioned four methods for the same set of images of Table 3. The result is shown in Table 4 from which we see clearly that the proposed DPA method yields the largest PSNR values for all the tested images, indicating its effectiveness for reducing image distortion.

Finally, we applied steganalysis to the four the methods using a software tool available at http://diit.sourceforge.net, which is an open-source implementation of RS analysis developed by Fridrich et al. [14]. We made a comparison of the analysis results shown in Table 5. Because the tool was designed for 24-bit color images with three color channels, we apply the proposed DPA and the greedy search method by embedding the message data evenly into image blocks of the three channels of R, G, and B. The third column in Table 5 specifies the detected message length of the cover image. This length value may be regarded as a "bias" of the RS detector, which supposedly should be zero because no message is hidden in the cover image. The last four columns specify the detected message lengths of the four methods, from whose contents we see that the DPA method is more robust against steganalysis than the greedy search method, and is not obviously so when compared with the other two methods.

Table 3
Comparison of run times for four methods for grayscale images (in unit of s).

Size	Stego-image	DPA	Greedy search	1-LSB	Hide4PGP
256×256	Lena256+200B	60	0.079	0.00016	0.0068
	Lena256+1200B	2112	0.437	0.00097	0.0072
	House256+200B	60	0.079	0.00016	0.0068
	House256+1000B	1473	0.366	0.00081	0.0071
	Jet256+200B	59	0.079	0.00016	0.0068
	Jet256+1200B	1082	0.439	0.00097	0.0072

Table 4
Comparison of PSNR values of the four methods for grayscale images (in unit of dB).

Size	Stego- image	DPA	Greedy search	1-LSB	Hide4PGP
256×256	Lena256+200B	55.40	54.59	51.11	51.15
	Lena256+1200B	55.19	54.59	51.14	51.12
	House256+200B	55.58	54.72	50.95	51.16
	House256+1000B	55.41	54.66	51.08	51.08
	Jet256+200B	54.86	54.55	51.03	51.03
	Jet256+1200B	55.28	54.60	51.10	51.24

7. Discussions and conclusions

A data hiding method for hiding messages into grayscale images with distortion reduction effects have been proposed. Two novel techniques for reducing distortions in stego-images have been adopted, one being an optimal dynamic programming algorithm, and the other the use of multiple block pattern encoding tables. First, a cost function has been proposed to estimate the weight of each bit in each pixel to be replaced according to an HVS model. Next, a horizontal data hiding scheme in which message data are embedded in a sequence of bit planes has also been proposed to decrease possible distortions in stego-images. Also, an optimal block pattern encoding table is chosen from 128 alternative ones for use in data embedding to minimize image distortion. The encoding tables are designed in such a way that up to three bits in a 2×2 image block can be embedded. Finally, the proposed method minimizes further the distortion using dynamic programming based on the proposed cost function.

The proposed dynamic programming algorithm has quadratic space and time complexities, and so takes long time to embed a long secret message. For applications with concerns of distortion reduction, the proposed method is good to use. If computation speedup is desired, the proposed greedy search algorithm may be applied. If high-speed processing is necessary, our method can be adapted to run on a parallel computer with each of the 128 block pattern encoding tables processed separately, and the dynamic programming steps parallelized.

At least two approaches may be adopted to make the proposed method more robust. First, multiple copies of a secret message may be embedded in the input image randomly with control by a key, so that an attack will not entirely destroy the secret information. And after the data are extracted by the proposed method, we may apply a voting scheme to recover the secret. The second approach is to try to place secret data in the more-significant-bits of the cover image, for example, in bp_2 and bp_3 in the proposed method, assuming that most attacks to BMP images are conducted to the LSBs. Because the information encoded in these bit-planes cannot be removed in most applications (otherwise, the image will be seriously distorted or destructed), hopefully this method will work in real applications.

Future works may be directed to extending the proposed method to process blocks larger than 2×2, resulting possibly in greater reduction of image distortion. It is also possible to embed multiple message data in a grayscale image for protecting the intellectual property right and authenticating multimedia data, to define more general cost functions for other HVS models, and to design better encoding tables to reduce image distortion further.

References

[1] S. Katzenbeisser, F.A.P. Petitolas, Information Hiding Techniques for Steganography and Digital Watermarking, Artech House, Boston, USA, 2000.
[2] L.M. Marvel, J.C.G. Boncelet, C.T. Retter, Spread spectrum image steganography, IEEE Transactions on Image Processing 8 (8) (1999) 1075–1083.
[3] M. Swanson, M. Kobayashi, A. Tewfik, Multimedia data-embedding and watermarking technologies, in: Proceedings of the IEEE, vol. 86, 1998, pp. 1064–1088.

Table 5
Comparison of RS analysis results of the four methods for color images.

Size	Cover image C	Detected message length of C	Stego-image S	Detected message length of S yielded by DPA	Detected message length of S yielded by greedy search	Detected message length of S yielded by 1-LSB	Detected message length of S yielded by Hide4PGP
256×256	Lena256	379.58B	Lena256+200B	458.42B	497.65B	529.13B	510.38B
	Lena256	379.58B	Lena256+1200B	1174.06B	1378.88B	1737.28B	999.68B
	House256	508.05B	House256+200B	596.93B	626.86B	561.83B	552.40B
	House256	508.05B	House256+1000B	1330.27B	1646.94B	1607.94B	1310.35B
	Jet256	731.17B	Jet256+200B	751.60B	839.44B	819.19B	772.44B
	Jet256	731.17B	Jet256+1200B	1341.43B	1403.21B	1244.48B	1406.40B

[4] R.Z. Wang, C.F. Lin, J.C. Lin, Hiding data in images by optimal moderately-significant-bit replacement, IEEE Electronics Letters 36 (25) (2000) 2069–2070.
[5] C.C. Chang, J.Y. Hsiao, C.S. Chan, Finding optimal least-significant-bit substitution in image hiding by dynamic programming strategy, Pattern Recognition 36 (2003) 1583–1595.
[6] C.K. Chan, L.M. Cheng, Improved hiding data in images by optimal moderately-significant-bit replacement, IEEE Electronics Letters 37 (16) (2001) 1017–1018.
[7] C.K. Chan, L.M. Cheng, Hiding data in images by simple LSB substitution, Pattern Recognition 37 (2004) 469–474.
[8] C.C. Thien, J.C. Lin, A simple and high-hiding capacity method for hiding digit-by-digit data in images based on modulus function, Pattern Recognition 36 (2003) 2875–2881.
[9] Y.K. Lee, L.H. Chen, High capacity image steganographic model, in: IEE Proceedings on Vision, Image Signal Process, vol. 147, no. 3, June 2000.
[10] S.H. Liu, T.H. Chen, H.X. Yao, W. Gao, A variable depth LSB data hiding technique in images, in: Proceedings of third International Conference on Machine Learning and Cybernetics, Shanghai, PR China, 2004, pp. 3990–3994.
[11] D.C. Wu, W.H. Tsai, Spatial-domain image hiding using an image differencing, in: IEE Proceedings on Vision, Image, and Signal Processing, vol. 147, no. 1, 2000, pp. 29–37.
[12] W.N. Lie, L.C. Chang, Data hiding in images with adaptive numbers of least significant bits based on the human visual system, in: Proceedings of IEEE International Conference on Image Processing, Taipei, Taiwan, vol. 1, 1999, pp. 286–290.
[13] S. Lyu, H. Farid, Detecting hidden messages using higher-order statistics and support vector machines, in: Lecture Notes in Computer Science, vol. 2578, 2003, pp. 340–354.
[14] J. Fridrich, M. Goljan, R. Du, Detecting LSB steganography in color and gray-scale images, IEEE Multimedia 8 (4) (2001) 22–28.
[15] I.S. Lee, W.H. Tsai, Data hiding in binary images with distortion-minimizing capabilities by optimal block pattern coding and dynamic programming techniques, in: IEICE Transactions on Information and Systems, vol. E90-D, No. 8, 2007, pp. 1142–1150.
[16] A.K. Jain, Fundamentals of Digital Image Processing, Prentice-Hall, Singapore, 1989.

About the Author—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.

About the Author—WEN-HSIANG TSAI was born in Tainan, Taiwan, Republic of China (ROC) 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, USA in 1977, and the Ph.D. degree in Electrical Engineering (with major in Computer Engineering) from Purdue University, West Lafayette, Indiana, USA 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 to 1988. He has also been the Associate Director of the Microelectronics and Information System Research Center from 1984 to 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 to 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 ROC or USA 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 USA 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 USA, 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.