PERGAMON

Pattern Recognition 36 (2003) 2875–2881

PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY

www.elsevier.com/locate/patcog

A simple and high-hiding capacity method for hiding digit-by-digit data in images based on modulus function

Chih-Ching Thien, Ja-Chen Lin*

Department of Computer and Information Science, National Chiao Tung University, Hsinchu, 300, Taiwan, ROC

Received 5 August 2002; accepted 14 June 2003

Abstract

This study presents a simple method for high-hiding capacity. The basic concept uses a modulus operation. The proposed method has the following advantages: (1) the proposed method outperforms the simple LSB substitution method given the same range (0–$m-1$) of data digits in the embedded data; (2) the proposed method achieves good image vision quality without the need for post-processing; (3) the proposed method is almost as simple as the LSB method in both coding and decoding; (4) the smaller error advantage over the simple LSB substitution method can also be mathematically verified; (5) the error is smaller than $\lceil (m-1)/2 \rceil$ for almost every pixel; (6) the proposed method has high-hiding capacity (for example, the proposed method can hide a 256×256 or 256×512 image in a 512×512 host image).
© 2003 Pattern Recognition Society. Published by Elsevier Ltd. All rights reserved.

Keywords: Steganography; LSB substitution; Modulus operation; High-hiding capacity

1. Introduction

Concealing data in images is increasingly important and has numerous applications [1–4]. Although hiding data by replacing the least significant bits (LSB) of the gray level of each pixel is probably the easiest approach, this method might cause poor visual quality or waste storage space, as explained below.

The LSB approach usually assumes that the base of the numerical data being hidden is a whole power of 2. (For example, 1-bit LSB means the hidden number is treated as a binary number because $2^1 = 2$, 2-bit LSB means the hidden number is treated as a number of base $2^2 = 4$, etc.) When the base of the number is not a power of 2, a base transform can be used to switch the input number to its binary equivalent. Sometimes, this kind of transform is not applicable/suitable. For example, if the system is used on-line, and the incremental data that keep on coming in are decimal and of variable length (say, 87597257364, then 129, then 76887564923, then 9654988, then 45, and so on), then it is difficult to transform data to binary numbers for the on-line real-time hiding, because nothing can be done about the transform on receiving 8759725736 (at least not until the final digit 4 is received and the number is known to be 87597257364). Of course, the precise transform, $(87597257364)_{10} = (1 \cdots 10100)_2$, $(129)_{10} = (10000001)_2$, etc., can be skipped and instead the so-called "digit-by-digit" transform, $(9 = (1001)_2, \ldots, 0 = (0000)_2)$, can be used to transform the 11-digit decimal number 87597257364 into a binary sequence 1000–0111–$\cdots$–0100. However, using the 4-bit LSB substitution method to embed the 4-bit binary sequence, results in poor visual performance (as pointed out below and in Figs. 2(a) and 3(a)).

The LSB approach is limited mainly in that it often causes artificial edges in the areas where pixel values vary smoothly (see the false contours on the shoulder of the "Lena" in Figs. 2(a) and 3(a); Section 3 explains why false contours appear in the smooth area). These artificial edges seriously

* Corresponding author. Department of Computer and Information Science, National Tsing Hua University, 1001 Ta Hsueh Road, Hsinchu 30050, Taiwan. Tel.: +886-03-5715900; fax: +886-03-5721490.

E-mail address: jclin@cis.nctu.edu.tw (J.C. Lin).

Nomenclature

y_i	gray value of the ith pixel of the original (host) image
$\hat{y}_i$	y_i is replaced by $\hat{y}_i$ after embedding data digit x_i
$\hat{y}_{iLSB}$	the ith gray value resulting from applying the LSB substitution method
m	the digits used in the data are $\{0, 1, \dots, m-1\}$
x_i	the ith embedded data digit (hence, $x_i \in \{0, 1, \dots, m-1\}$)
k	number of substitution bits in a pixel (for the LSB method)
d_i	the value $(\hat{y}_{iLSB} - y_i)$
d_i'	the value $(\hat{y}_i - y_i)$

damage the visual quality of the stego images because they look abnormal. Notably, even if the smooth area in which artificial edges appear is small, these artificial edges are still highly visible because human vision is more sensitive to the modification of smoothly varying areas in an image than to the modification of textured areas. Consequently, hackers are likely to suspect that information is hidden within an image that is treated in this way, compromising the security of the hidden information. Unfortunately, most ordinary images (host images) contain several areas in which pixel values smoothly vary, and hence the LSB approach becomes inapplicable. This study attempts to design a high-hiding-capacity method (e.g. hiding a 512×256 image in a 512×512 host image) which has the advantages of the LSB method (i.e. simplicity), can be used for real-time (digit-by-digit) processing, and most importantly, achieves better visual quality than the LSB approach (not only avoiding the artificial edges mentioned above, but also reducing the distortion at each pixel of the host image after data embedding).

To achieve this goal, this study develops a method based on a modulus operation that is a modification of the LSB method. To clarify the basis of the proposed method, the well-known simple k-bit LSB substitution method is described first. Suppose that people wish to embed the ith digit x_i $(0 \leqslant x_i < m; m = 2^k)$ of the data into a pixel with a gray value of y_i $(0 \leqslant y_i \leqslant 255)$ in the host image. Assume that

$$y_i = m \times t_i + b_i,$$

where $b_i = y_i \bmod m$, $t_i = \lfloor y_i/m \rfloor$, and so both t_i and b_i are non-negative integers. The resulting gray value $\hat{y}_{iLSB}$ replacing y_i is

$$\hat{y}_{iLSB} = (y_i - b_i) + x_i. \tag{1}$$

In later days, the embedded data digit x_i can be extracted from $\hat{y}_{iLSB}$ by

$$x_i = \hat{y}_{iLSB} \bmod m. \tag{2}$$

The proposed method is based on observation of the above equations and the property of modulus operation. The resulting gray value $\hat{y}_i$ generated by the proposed method is $\hat{y}_i = y_i - b_i + x_i + lm$ where l is an integer suitably chosen from $\{0, 1, -1\}$, as explained in Section 2.

2. Proposed method and some related mathematical properties

2.1. Proposed method

Assume that m kinds of symbols are used in the data, and hence the digital system (i.e. the digits being used) representing the embedded data is $\{0, 1, 2, \dots, m-1\}$. Moreover, suppose that the goal is to embed the ith digit x_i $(0 \leqslant x_i < m)$ of the data into the pixel with a gray value of y_i $(0 \leqslant y_i \leqslant 255)$ in the host image. First,

$$b_i = y_i \bmod m, \tag{3}$$

$$d_i = x_i - b_i \tag{4}$$

are evaluated. Then d_i' is assessed using the following rule:

$$d_i' = \begin{cases} d_i & \text{if } \left(-\left\lfloor \dfrac{m-1}{2} \right\rfloor\right) \leqslant d_i \leqslant \left\lceil \dfrac{m-1}{2} \right\rceil, \\ d_i + m & \text{if } (-m+1) \leqslant d_i < \left(-\left\lfloor \dfrac{m-1}{2} \right\rfloor\right), \\ d_i - m & \text{if } \left\lceil \dfrac{m-1}{2} \right\rceil < d_i < m. \end{cases} \tag{5}$$

(Notably $-\lfloor (m-1)/2 \rfloor \leqslant d_i' \leqslant \lceil (m-1)/2 \rceil$, as stated in Property 1 later.) Finally, use

$$\hat{y}_i = y_i + d_i' \tag{6a}$$

as the resulting gray value to replace y_i. In a few cases, the $\hat{y}_i$ in Eq. (6a) may fall outside the valid gray value range, that is, $\hat{y}_i > 255$ or $\hat{y}_i < 0$. Should this occur, $\hat{y}_i$ is reset using the following equation:

$$\hat{y}_i = \begin{cases} y_i + d_i' + m & \text{if } y_i + d_i' < 0, \\ y_i + d_i' - m & \text{if } y_i + d_i' > 255. \end{cases} \tag{6b}$$

If m happens to satisfy $2^k = m$ for some integer k, then Eq. (6b) can be reduced to

$$\hat{y}_i = y_i + d_i, \tag{6c}$$

the proof is omitted here to save space. The $\hat{y}_i$ obtained using Eq. (6c) is equivalent to the $\hat{y}_{iLSB}$ obtained in Eq. (1) because of Eq. (4). Notably Eq. (6b) is seldom used. For example, if the hidden data are decimal numbers, then 0–$m-1$ is 0–9, and thus $\lfloor (m-1)/2 \rfloor = 4$ and $\lceil (m-1)/2 \rceil = 5$; so Property 1 below ensures that Eq. (6a) is never overflow or underflow provided the original gray y_i of the host image satisfies $4 \leqslant y_i \leqslant 250$. (Many well-known images such as Lena, Jet, and Monkey, have no pixels outside the range 4–250.)

Extracting the data digit x_i from the image is very easy. Just calculate

$$x_i = \hat{y}_i \bmod m \tag{7}$$

just as with the LSB method (see Eq. (2)).

In fact, as long as the decoding formula is Eq. (7), the digit x_i could be encoded by $x_i + t_i m$ for any valid integer t_i, and the proposed method simply selects the best t_i, namely that which generates the gray value $x_i + t_i m$ closest to the original gray value y_i. The search for the best $x_i + t_i m$ in fact can be reduced to the search for the best $\hat{y}_i = y_i - b_i + x_i + lm$, where $l \in \{0, 1, -1\}$, so that $\hat{y}_i$, is closest to y_i. Eq. (5) helps to determine this l automatically, because Eqs. (4)–(6a) imply $\hat{y}_i = y_i + d'_i = y_i + d_i + lm = y_i - b_i + x_i + lm$ with l chosen from $\{0, 1, -1\}$. Conversely, Eq. (1) means that the LSB method does not choose l (its l is always 0), hence its error cannot be smaller than that using our method.

2.2. Some mathematical properties of the proposed method

Property 1 below gauges the distortion created by the proposed method. Property 2 then gives the distortion comparison at each pixel.

Property 1. In Eq. (6a), the distortion d'_i between the original gray value y_i and the resulting gray value $\hat{y}_i$ is in the range $-\lfloor (m-1)/2 \rfloor$–$\lceil (m-1)/2 \rceil$.

Property 2. If m happens to satisfy $2^k = m$ for some integer k, then at each pixel the gray value distortion of the k-bit LSB method is never smaller than the gray value distortion caused by the proposed method.

Property 1 is due to Eq. (5), whereas Property 2 results from the paragraph below Eq. (7). Moreover, Property 2 deals with the case when $m = 2^k$ for some integer k. For the general case when m does not necessarily equal any 2^k for any integer k, Properties 3 and 4 below can still be used to estimate the mean square error (MSE) caused by the proposed and LSB methods, respectively. The MSE is then compared between the two methods in both Properties 4′ and 4″. Lemma 1 is for proving Property 3 only.

Lemma 1. *In the general case (that is, embedding with Eq.* (6a)), *then for any given gray value* $y_i \in \{\lfloor (m-1)/2 \rfloor, \ldots, 255 - \lceil (m-1)/2 \rceil\}$, *the proposed method, denoted as* $F_{y_i} : x \to \hat{y} = F_{y_i}(x)$, *is a bijection function (one to one and onto) from the set* $A = \{0, 1, \ldots, m-1\}$ *onto the set* $C_{y_i} : \{y_i - \lfloor (m-1)/2 \rfloor, \ldots, y_i, \ldots, y_i + \lceil (m-1)/2 \rceil\}$. (*In notation,*

$$x \in A \xrightarrow{\hat{y} = F_{y_i}(x)} \hat{y} \in C_{y_i}.)$$

Proof. If x_1 and x_2 are both mapped to the same $\hat{y}$, then Eq. (7) implies that $x_1 = \hat{y} \bmod m$ and $x_2 = \hat{y} \bmod m$. Consequently, $x_1 = x_2$. The mapping thus is one to one. The "onto" part is proved below. Property 1 implies that $y_i - \lfloor (m-1)/2 \rfloor \leqslant \hat{y} \leqslant y_i + \lceil (m-1)/2 \rceil$, so the range of the mapping must be contained in the set C_{y_i}. Notably, both A and C_{y_i} have m numbers, and the mapping is already demonstrated to be one to one, so the mapping from A to C_{y_i} must be onto. □

Property 3. If the embedded data X is uniformly distributed, then the expected MSE in the host image resulting from the proposed method is approximately $(m^2 - 1)/12$ when m is an odd integer, and is approximately $(m^2 + 2)/12$ when m is an even integer.

Proof. Eq. (6b) is rarely used (see the paragraph between Eqs. (6c) and (7)), and thus only the general case (i.e. Eq. (6a)) is considered here. First, the expectation of the square error $E((\hat{Y} - y_i)^2)$ is checked for a single pixel y_i. Here, y_i denotes the original gray value of the host image, while $\hat{Y}$ represents a random variable denoting the gray value of the resulting pixel. Since X is uniformly distributed over the range 0–$m-1$ and the mapping between X and $\hat{Y}$ is one to one and onto (see Lemma 1), $\hat{Y}$ is also uniformly distributed (over the range $y_i - \lfloor (m-1)/2 \rfloor$–$y_i + \lceil (m-1)/2 \rceil$), see Property (1). Therefore, Property 1 (with the $\hat{y}_i$ there replaced by $\hat{Y}$) reads

$$E((d'_i)^2) = E((\hat{Y} - y_i)^2)$$

$$= \frac{\left(-\left\lfloor \frac{m-1}{2} \right\rfloor\right)^2 + \cdots + (-1)^2 + 0^2 + 1^2 + \cdots + \left\lceil \frac{m-1}{2} \right\rceil^2}{m}. \tag{8}$$

Case 1: When m is an odd integer. Because

$$-\left\lfloor \frac{m-1}{2} \right\rfloor \leqslant \hat{Y} - y_i = d'_i \leqslant \left\lceil \frac{m-1}{2} \right\rceil$$

(see Property 1) and the odd integer m implies $\lfloor (m-1)/2 \rfloor = (m-1)/2 = \lceil (m-1)/2 \rceil$, $-(m-1)/2 \leqslant d'_i \leqslant (m-1)/2$ can be derived. Therefore, d'_i is uniformly distributed over $-(m-1)/2$–$(m-1)/2$, and Eq. (8) then becomes

$$E((d'_i)^2)$$

$$= \frac{\left(-\frac{m-1}{2}\right)^2 + \left(-\frac{m-1}{2} + 1\right)^2 + \cdots + 0^2 + \cdots + \left(\frac{m-1}{2}\right)^2}{m}$$

$$= \frac{2\left(0^2 + 1^2 + \cdots + \left(\frac{m-1}{2}\right)^2\right)}{m}$$

$$= \frac{2\dfrac{\left(\left(\frac{m-1}{2}\right)\left(\frac{m+1}{2}\right)m\right)}{6}}{m} = \frac{m^2-1}{12}.$$

Case 2. When m is an even integer, the proof is similar to Case 1.

From Cases 1 and 2, it can be concluded that $E((\hat{Y}-y_i)^2)$ does not depend on pixel position i. Consequently, the expectation value of the square error is the same for each pixel. Accordingly, the expectation of MSE for the whole host image is identical to that of MSE for a single pixel. □

The following discusses the expectation value of the MSE for the simple k-bit LSB substitution method. Notably, when people try to use a digit-by-digit approach to embed data each digit of which is in the range 0–$m-1$, then $k=\lceil \log_2 m \rceil$ bits are used for substitution in the LSB substitution method.

Property 4. Let the digits used in the embedded data X be 0–$m-1$, and let $k=\lceil \log_2 m \rceil$. If the embedded data X and B $(=y \bmod (2^k))$ are both uniformly distributed, then the expectation value of the MSE resulting from the simple k-bit LSB substitution method is $(2\times 2^{2k}-3m\times 2^k+2m^2-1)/6$. (If $\log_2 m$ is an integer, then the expectation value is $(m^2-1)/6$.)

Proof. The expectation of the MSE is $E((\hat{Y}_{LSB}-Y)^2)$, where $\hat{Y}_{LSB}$ denotes the resulting value of the LSB substitution method corresponding to the original gray value Y. The following can be derived:

$$E((\hat{Y}_{LSB}-Y)^2)$$

$$=E((B-X)^2) \tag{9}$$

$$=E(B^2-2BX+X^2)$$

$$=E(B^2)+E(X^2)-2E(B)E(X) \tag{10}$$

$$=\frac{\sum_i b_i^2}{2^k}+\frac{\sum_i x_i^2}{m}-2\,\frac{\sum_i b_i}{2^k}\,\frac{\sum_i x_i}{m} \tag{11}$$

$$=\frac{((2^k-1)2^k(2\times 2^k-1))/6}{2^k}+\frac{((m-1)m(2m-1))/6}{m}-2\,\frac{((2^k-1)2^k)/2}{2^k}\,\frac{((m-1)m)/2}{m}$$

$$=\frac{(2^k-1)(2\times 2^k-1)}{6}+\frac{(m-1)(2m-1)}{6}-2\,\frac{(2^k-1)}{2}\,\frac{(m-1)}{2}$$

$$=\frac{2\times 2^{2k}-3m\times 2^k+2m^2-1}{6}$$

$$\left(=\frac{m^2-1}{6} \quad \text{while } \log_2 m \text{ is an integer}\right).$$

Eq. (9) follows from Eq. (1), while Eq. (10) follows from B being independent of X (for B is related to the image only, not to the data X). Moreover, Eq. (11) follows from that fact the possible values of b_i are all in the range 0–(2^k-1), while those of x_i are all in the range 0–$m-1$. □

Property 4′. Let $k=\lceil \log_2 m \rceil$. Assume both B $(=y \bmod (2^k))$ and the data X (with digits in the range 0–$m-1$) are uniformly distributed. Then the expected value of the MSE of the k-bit LSB method is never smaller than that of the proposed method.

Property 4″. Assume both B $(=y \bmod (2^k))$ and the embedded data X (with digits in the range 0–$m-1$) are uniformly distributed. If $m=2^k$ for some integer k, then the expectation value of MSE caused by the proposed method is $\frac{1}{2}+\frac{3}{2}(1/(m^2-1))$ of that caused by the simple LSB substitution method.

Properties 4′ and 4″ are both trivial results of Properties 3 and 4. In the special case that $m=2^k$ for some integer k, Property 4′ is also a direct result of Property 2.

3. Experimental results

This section presents and discusses the experimental results using the proposed method. The simple LSB substitution method and the GA-improved LSB substitution method [5] are also implemented to provide a comparison. $m=16$ (that is, four LSBs) is taken as an example. The images tested in the present experiment are all 8-bit images, as displayed in Fig. 1. Figs. 1(a)–(d) shows the data images of size 512 × 256 each; Figs. 1(e) and (f) display the host images of size 512 × 512 each. Each of Figs. 1(a)–(d) is embedded into each of Figs. 1(e) and (f) (with $m=16$), and the resulting PSNR is listed in Table 1. The corresponding theoretic PSNR (according to Properties 3 and 4) are also calculated and presented in the last row of Table 1. Table 1 indicates that the proposed method achieves higher PSNR than the simple or GA-improved LSB substitution [5] methods (around 2.83 or 1.40 dB better, respectively). Additionally, the theoretic PSNR of both the proposed method and the simple LSB substitution method (calculated from Properties 3 and 4, respectively) closely approximate the experimental results.

Fig. 2 displays the images resulting from the above experiment. Figs. 2(a)–(c) display the three images resulting from embedding Fig. 1(a) into Fig. 1(e) (with $m=16$) using the three different methods. Fig. 2(a) shows the image resulting from the simple LSB substitution method, while Figs. 2(b) and (c) illustrate the images resulting from the GA-improved LSB substitution method and the proposed method, respectively. Clearly, the proposed method achieves better performance in terms of vision quality than the other two methods. Moreover, the false contour on the shoulder is

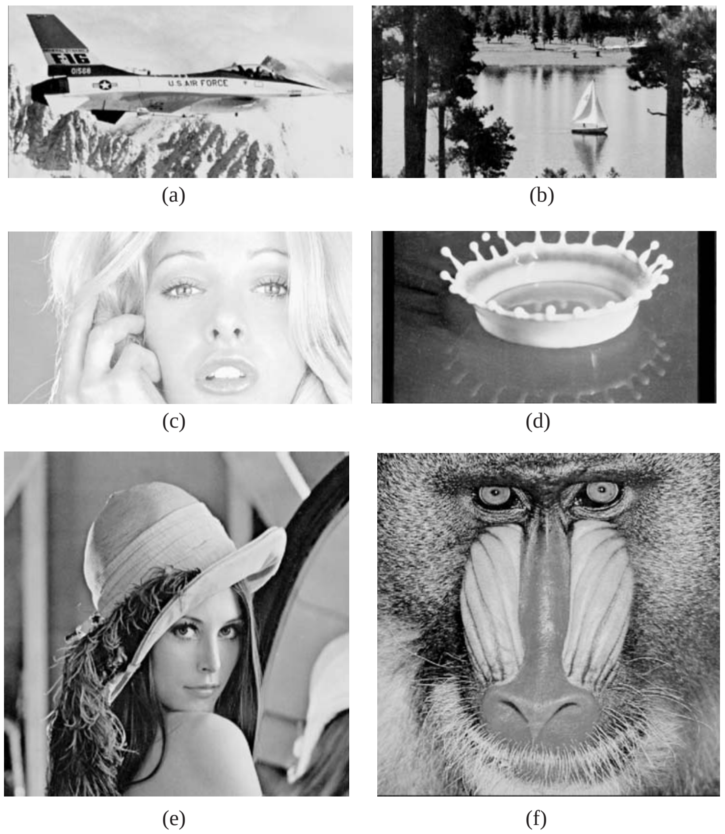

Fig. 1. The test images in the experiment: (a)–(d) are "data" images of size 512 × 256; (e) and (f) are "host" images of size 512 × 512.

Table 1
Experimental PSNR of the resulting images (with $m = 16$) generated using the simple and GA-improved LSB substitution methods, and the proposed method

	Host = Lena			Host = Baboon		
	The simple LSB method	The GA-improved LSB method	Our method	The simple LSB method	The GA-improved LSB method	Our method
Scene	32.28	34.08	34.79	32.31	33.99	34.82
Jet	32.02	34.16	34.76	32.08	34.28	34.81
Tiff	31.30	32.90	34.80	31.30	32.97	34.82
Milk	32.27	32.44	34.83	32.27	32.42	34.79
Theoretic value	31.85		34.81	31.85		34.81

Theoretic values (the expected values assuming uniform distribution) given by Properties 3 and 4 are also included for comparison.

obvious in Figs. 2(a) and (b), but not in Fig. (c). The artificial edges caused by the simple LSB method are explained easily below. In the area where pixel values smoothly vary (e.g. a small portion of the shoulder of Lena), the simple LSB encoding method converts the area into an area with "uniformly distributed pixels" when the hidden data are uniformly distributed. The conversion thus changes the average gray value of the area. For example, if the gray values of the pixels in the smooth area are in the range $\{i \times 2^k, 1 + i \times 2^k, \ldots, 2^k - 1 + i \times 2^k\}$ and originally have

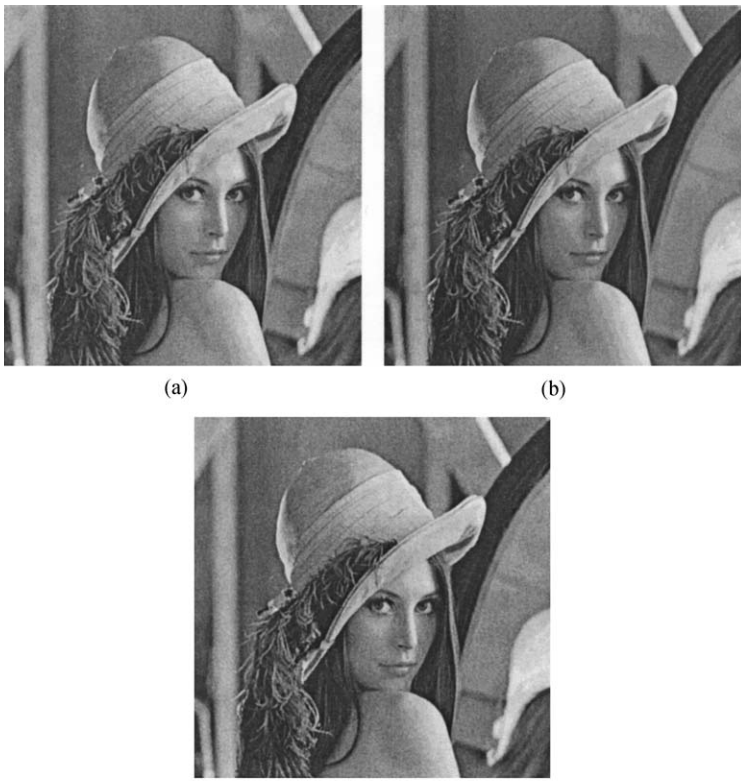

Fig. 2. Three resulting images with $m = 16$: (a)–(c) denote the resulting images by embedding Fig. 1(a) into Fig. 1(e) using three different methods; (a) is the image using the simple LSB substitution method, while (b) and (c) are those using the GA-improved LSB substitution method and the proposed method, respectively.

average gray value $\bar{y}$, then after the conversion, the average gray value of the area becomes $(i + \frac{1}{2}) \times 2^k$, which is independent of $\bar{y}$, and the distortion becomes big. However, since the method described here always selected the closest valid value (to the host value) when encoding data, it (almost) preserves the average gray $\bar{y}$ of the encoding area. (The preservation can be easily proved by calculating the expectation $E(d')$ (similar to Eq. (8), but without squaring each term), and obtaining the result that $E(d') = 0$ (i.e. no change) if m is odd; or $E(d') = \frac{1}{2}$ if m is even.) Therefore, no artificial edges exist.

The final experiment embedded randomly generated data into the Lena. The randomly generated data contain 512×512 digits, each of which is randomly picked from the integers in the range 0–9. (Some scientific data or important data may resemble the random numbers in a certain range, for example, important phone numbers may be embedded in an image, where each digit is in the range 0–9.) Fig. 3 shows the resulting images. In Fig. 3, (a) is obtained using the simple LSB method, while (b) is obtained using the present method. The vision quality using the proposed method obviously is superior to that using the simple LSB method.

4. Concluding remarks

This work proposes a simple embedding method based on the modulus operation. As noted in the abstract and shown in the figures, the method is obviously better than the simple LSB substitution method. On the other hand, compared with the GA-improved LSB method [5], the proposed method has better PSNR (see Table 1), better image quality (see Fig. 2), and most of all, is considerably simpler and faster (with a coding time approximately $\frac{1}{2}$ of that with the GA-improved LSB method).

This work conducts some mathematical analysis, including:

(1) the range of the visual distortion (gray value shift) at each pixel of the host image (Property 1);
(2) the error "at each pixel" is never worse than the LSB substitution method when $m = 2^k$ for a certain k (Property 2);
(3) the expectation value of the MSE of the proposed method (Property 3);
(4) the expectation value of the MSE of the LSB substitution method (Property 4);

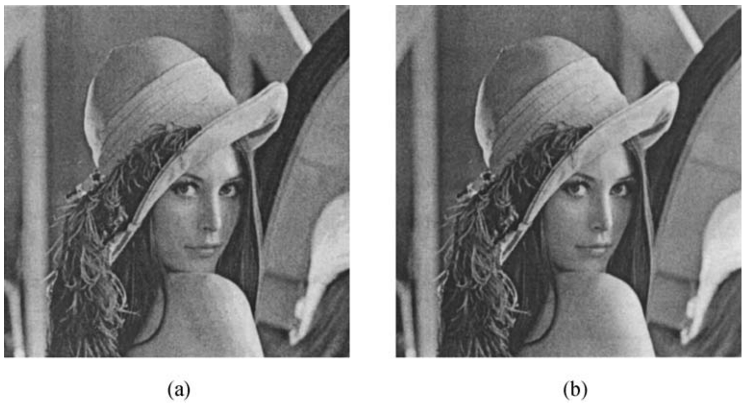

Fig. 3. Experimental results of embedding randomly generated decimal digits (512 $\times$ 512 digits and each digit is 0–9) in Lena: (a) is the resulting image obtained by the simple LSB method, (b) is the resulting image obtained by the proposed method. Notably, $m = 10$ here, and thus m is not a whole power of 2.

(5) the expectation value of our MSE will not exceed that of the LSB substitution method (Properties $4'$ and $4''$).

The proposed method is easy and fast. In fact, the extraction of the proposed method is the same as that of the simple LSB substitution method (see Eqs. (2) and (7)). Therefore, the proposed method can replace the simple LSB method in any application in which the simple LSB method was adopted [6–9].

Acknowledgements

This work was supported by the National Science Council, ROC, under Grant NSC 91-2213-E009-097. We thank the referee whose comments have greatly improved this paper.

References

[1] W. Bender, D. Gruhl, N. Morimoto, A. Lu, Techniques for data hiding, IBM Systems J. 35 (3–4) (1996) 313–316.
[2] W. Bender, F.J. Paiz, W. Butera, S. Pogreb, D. Gruhl, R. Hwang, Applications for data hiding, IBM Systems J. 39 (3–4) (2000) 547–568.
[3] E. Adelson, Digital signal encoding and decoding apparatus, US Patent, No. 4939515, 1990.
[4] L.F. Turner, Digital data security system, Patent IPN WO 89/08915, 1989.
[5] R.Z. Wang, C.F. Lin, J.C. Lin, Image hiding by optimal LSB substitution and genetic algorithm, Pattern Recognition 34 (3) (2001) 671–683.
[6] W.N. Lie, L.C. Chang, Data hiding in images with adaptive numbers of least significant bits based on the human visual system, International Conference on Image Processing, IEEE Part Vol. 1, Kobe, Japan, 1999, pp. 286–290.
[7] Y.C. Hou, P.M. Chen, Y.F. Chiao, Steganography: an efficient data hiding method, Conference on Intelligent Systems, Vol. 4, NC, USA, 1998, pp. 211–214.
[8] F.A.P. Petitcolas, R.J. Anderson, M.G. Kuhn, Information hiding—a survey, Proc. IEEE 87 (7) (1999) 1062–1078.
[9] J. Fridrich, M. Goljan, Images with self-correcting capabilities, International Conference on Image Processing, IEEE Part Vol. 3, Kobe, Japan, 1999, pp. 792–796.