A Steganographic Method for Digital Images by Multi-Pixel Differencing

Cheng-Hsing Yang[1], Chi-Yao Weng[2]
Department of Computer Science, National Pingtung University of Education, Pingtung 9003, Taiwan, R.O.C
{chyang[1], bm094108[2]}@mail.npue.edu.tw

ABSTRACT

In order to provide large embedding capacity and to minimize distortion for the stego-image, a steganographic method using multi-pixel differencing is presented in this paper. We propose a method which is based on pixel-value differencing. We use neighboring pixels correlation to estimate the degree of smoothness or contrast of pixels. If pixels are located in edge area, then they may be embedded with larger capacity than that in smooth area. In order to improve the embedding capacity and reduce the distortion, multi-pixel differencing and pixel-value shifting are proposed in this paper. The experimental results show that our method provides a large embedding capacity without making noticeable distortion.

Key Words: Steganography, Multi-pixel differencing, Embedding capacity

1: INTRODUCTIONS

Nowadays, Internet develops so rapidly that it has become a common communication channel. In this open channel, transmitted data are exposed. Therefore, to find methods of transmitting data secretly becomes an important topic. Steganography is one of the most important approaches to perform this object [1].

Steganography can be simply distinguished into digital watermarking and data hiding. Digital watermark techniques are aimed at enhancing the power of copyright protection and authentication [2]. On the other hand, the main object of data hiding is to increase the hiding capacity and imperceptibility [3][4]. This paper focuses on data hiding.

Many steganographic techniques about embedding data into images have already been proposed. Some of them are based on the method of replacing the least-significant-bits (LSBs) of the pixels of the host images [5][6]. However, for the reason that embedding secret messages by replacing the LSBs directly is easily detected by human's eyes or programs [3][7], some embedding approaches considered the concept of human visual system [3][8][9].

Wu and Tsai proposed a "differencing" stgeanographic method that uses the difference between two pixels to determine how many secret bits should be embedded [8]. The difference value is then adjusted according to the embedded secret bits, and the difference between the original difference value and the new one is shared between the two pixels. Also, Chang and Tseng proposed a side match approach to embed secret messages [9]. The number of bits embedded into a pixel is decided by the difference between the pixel and its side pixels

This paper proposes an efficient steganographic method to hide data into gray-level images by multi -pixel differencing. It is based on the concept of human visual system, which points out that changing in smooth area is more sensitive than changing in edge area. We process a block with four neighboring pixels at a time. Each block forms three two-pixel groups and the difference between two pixels within a group is used to embed data. After the difference values created by the three groups are adjusted to embed data, a pixel-value shifting approach is used to optimize the image quality of this block. Experimental results show that our approach provides a larger embedding capacity and remains acceptable image quality.

The remainder of this paper is organized as follows. In Section 2, two recently embedding methods are reviewed. Our approach is described in Section 3. Experimental results are shown in Section 4. Finally, conclusions are presented in Section 5.

2: RELATED WORKS

In this section, we introduce Wu and Tsai's pixel-value differencing method and Chang and Tseng's side-match method.

2.1: WU AND TSAI'S DIFFERENCING MEHTOD

Wu and Tsai's differencing method embeds data using pixel-value differencing of the gray-level images. First, the host image is partitioned into non- overlapping consecutive two-pixel blocks by running through all the rows of the host image in a zigzag manner. Next, a difference value d is calculated from the two pixels of each block. By symmetry, only the possible absolute values d (0 through 255) are considered and they are classified into a number of contiguous ranges (called quantization), say R_i where $i = 1, 2, 3, \ldots, r$. If d falls in the range R_k, the width of R_k determines the number of bits to be embedded. Finally, d is replaced by a new difference value d', which is equivalent to the sum of the

embedded value and the lower bound of R_k. Moreover, an inverse calculation from d' is performed to yield the new gray values for the two pixels in the block. Note that it must avoid the resulting pixels falling off the boundaries of the range [0, 255]. Therefore, it needs to use the maximally possible value of d' to check whether these two pixels can be used to embed data.

The reverse of the embedding process is to extract the secret data from the stego-image. The number of secret bits embedded in a two-pixel block is determined by the range R_i, which the difference value between the two pixels belongs to. Then, the embedded bits in the block can be easily extracted by subtracting the lower bound of R_k from the difference value of the block.

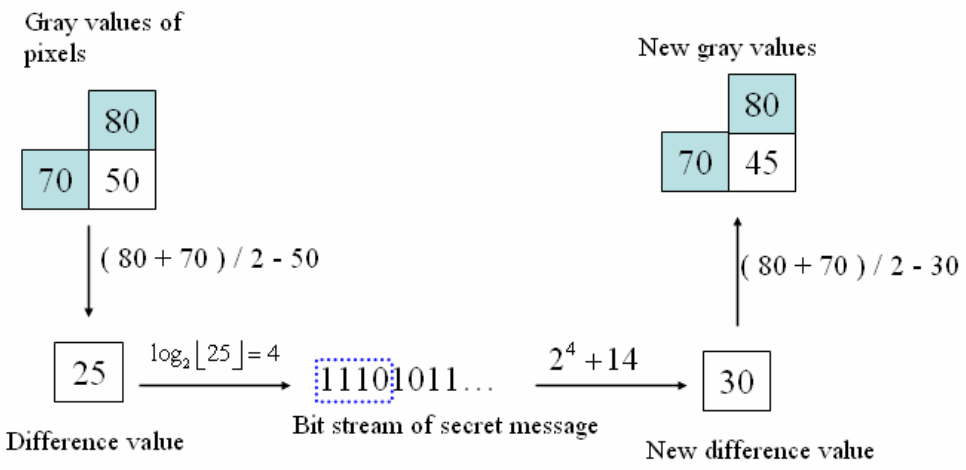

Fig. 1. An example of Chang and Tseng's embedding method.

2.2: CHANG AND TSENG'S SIDE-MATCH METHOD

In the two-sided side-match approach, P_X denotes the pixel, which the secret data are to be currently embedded in, and its upper and left pixels are denoted by P_U and P_L, respectively. A total of N bits can be embedded in P_X according to Eq. (1), where d is the difference value derived by subtracting P_X from the mean value of P_U and P_L. If d has values –1, 0, or 1, then one bit of secret data is embedded into the LSB of pixel P_X using the conventional LSB substitution method. The N bits of the embedded data are then converted to a decimal value b, and the new difference value d' is calculated using Eq. (2). Finally, the new pixel value P_X' can be calculated by subtracting d' from the mean value of the P_U and P_L.

$$N = \lfloor \log_2 |d| \rfloor \tag{1}$$

$$d' = \begin{cases} 2^N + b, & \text{if } d > 1 \\ -(2^N + b), & \text{if } d < 1 \end{cases} \tag{2}$$

Fig. 1 shows an example of Chang and Tseng's embedding method. In the figure, assume that P_U = 80, P_L = 70, and P_X = 50. The difference value d is 25 (i.e., (80 + 70) / 2 – 50 = 25), and the length of embedded bits is $\lfloor \log_2 |25| = 4 \rfloor$. Assume that the four secret bits are $(1110)_2$, whose decimal value is 14. Then new difference value is $2^4 + 14 = 30$, which is used to replace the original difference value 25. Finally, the new difference 30 is used to recalculate the pixel value, and the new pixel value of P_X is $P_X' = (80 – 70) / 2 – 30 = 45$.

The extracting phase is the reverse of embedding. First, the difference value d' is derived by subtracting P_X from the mean value of P_U and P_L. If d' has values –1, 0, or 1, only one bit of secret data is extracted from LSB of pixel P_X'; otherwise the number of bits N embedded in P_X' is computed using Eq. (3). The embedded value b can be calculated by Eq. (4) and recovered to a binary string with N bits.

$$N = \lfloor \log_2 |d'| \rfloor \tag{3}$$

$$b = \begin{cases} d' - 2^N, & \text{if } d > 1 \\ -d' - 2^N, & \text{if } d < 1 \end{cases} \tag{4}$$

3: THE PROPOSED SCHEME

From the concept of pixel-value differencing proposed by Wu and Tsai, two pixels are grouped for embedding secret data in each time. The drawback of their method is that only two pixels are considered at a time, therefore, it can not consider the features of edges sufficiently. Moreover, two-pixel groups are non-overlapping, and it will lower the embedding capacity. Side-match approach embeds data into each pixel by adjusting its gray value to fit the new difference value between the mean value of its side pixels and the gray value of itself for the reason that it would result in larger capacity. However the propagated error would be serious. In order to overcome above drawbacks, a new approach is proposed in this section. Our approach is based on block embedding. A block with four pixels is considered at a time, and more difference values within the block are used to embed secret data. The grouping,

embedding and extracting procedures of our approach are described in the following subsections.

3.1: GROUPING OF A BLOCK

All of the pixels are 256 gray-values in the host image. All difference values are computed from every non-overlapping block with four neighboring pixels, say $p_{i,j}$, $p_{i,j+1}$, $p_{i+1,j+1}$, and $p_{i+1,j}$, of a given host image. The way of partitioning the host image into four-pixel blocks runs through all of rows in a raster scan.

In each block, pixels $p_{i,j}$, $p_{i,j+1}$, $p_{i+1,j+1}$, and $p_{i+1,j}$ are renamed as p_0, p_1, p_2, and p_3, and their corresponding gray values g_0, g_1, g_2, and g_3 satisfy the condition $g_0 \le g_1, g_2, g_3$. Note that p_0 is the pixel with the smallest gray value in the block. If there exist more than one pixels with the smallest gray value, p_0 is assigned to the pixel with the smallest gray value in the priority of $p_{i,j}$, $p_{i,j+1}$, $p_{i+1,j+1}$, and $p_{i+1,j}$. Also, p_1, p_2, and p_3 are assigned to the remainder pixels in the clockwise sequence, which begins at the pixel next to p_0 clockwise. An example is shown in Fig. 2. After the four pixels of a block are renamed, the block is used to form three two-pixel groups (g_0, g_1), (g_0, g_2), and (g_0, g_3), named as *group*1, *group*2, and *group*3, respectively. Three group differences d_1, d_2, and d_3 are computed by ($g_1 - g_0$), ($g_2 - g_0$), and ($g_3 - g_0$), respectively. All group differences are in the range from 0 to 255. We classify all possible values into a number of contiguous ranges, say R_i where i = 1, 2, 3, …, r. These ranges are assigned indices from 1 to r. The lower and upper bound values of R_i are denoted by l_i and u_i, where l_1 is 0 and u_r is 255. The width of R_i is ($u_i - l_i + 1$). Each range is taken to be a power of 2. This restriction of width facilitates embedding binary data. A difference value which falls in a range with index k is said to have index k. We embed some bits of the secret data into a block by changing the difference values of the three groups. Furthermore, a skill of pixel-value shifting is proposed to optimize the quality of the block. More details are described in the next subsection.

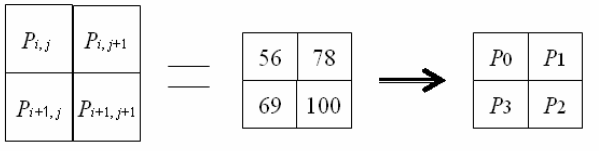

Fig. 2. An example of block whose pixels are renamed as p_0, p_1, p_2, and p_3.

3.2: EMBEDDING SCHEME

We want to embed secret bits of the bit steam into each block, which contains three two-pixel groups. The number of bits to be embedded in each block varies and is decided by the width of the range, which the difference values of the block belong to. Let a four-pixel block B have differences d_i (i = 1, 2, 3), which fall into the ranges with indices k_i (i = 1, 2, 3), respectively. Then, the number of bits, say n_i, to be embedded in each group is calculated by $n_i = \log_2(u_{k_i} - l_{k_i} + 1)$. We embed n_1, n_2, and n_3 bits into *group*1, *group*2, and *group*3, respectively. Let S_1, S_2, and S_3 be the next three sub-streams to be embedded, where $|S_1| = n_1$, $|S_2| = n_2$, $|S_3| = n_3$. Also, let b_1, b_2, and b_3 be the decimal values of S_1, S_2, and S_3, respectively. The new difference values of d_1, d_2, and d_3 are computed by

$$d_i' = l_{k_i} + b_i, \text{ where } i = 1, 2, 3. \quad (5)$$

To embed values b_1, b_2, and b_3 into *group*1, *group*2, and *group*3, respectively, difference value d_i is replaced by d_i', i = 1, 2, 3. Then, an inverse calculation is performed to yield the new gray values g_0', g_1', g_2', and g_3' using $g_0' = g_0$, $g_1' = g_0 + d_1$, $g_2' = g_0 + d_2$, and $g_3' = g_0 + d_3$. In order to increase the image quality, a skill of pixel-value shifting is proposed to minimize the sum of square errors between the original block and the embedded block. The pixel-value shifting is to shift the gray values of a group (g_i, g_j) concurrently, i.e., both g_i and g_j increase or decrease into g_i' and g_j' such that $g_i' - g_j' = g_i - g_j$ and $0 \le g_i', g_j' \le 255$. Therefore, an embedded block (g_0', g_1', g_2', g_3') is optimized by the pixel-value shifting such that $(g_0 - g_0')^2 + (g_1 - g_1')^2 + (g_2 - g_2')^2 + (g_3 - g_3')^2$ is minimized. With the skill of the pixel-value shifting, the block is abandoned only if one of d_i, i = 1, 2, 3, falls into the last range, which is the only range whose width is not a power of 2 in our approach. That is to say, if one of index k_i, i = 1, 2, 3, equals r, the block is not used to embed data. This check is called the falling-off-boundary checking.

In order to embed and extract the bit streams correctly, it is necessary to distinguish the pixels p_0, p_1, p_2, and p_3 in a block before embedding and after embedding. A simple strategy is to assign p_i, i = 0, 1, 2, 3, to fixed positions, for example, p_0 is assigned to $p_{i,j}$, p_1 is assigned to $p_{i,j+1}$, p_2 is assigned to $p_{i+1,j+1}$, and p_3 is assigned to $p_{i+1,j}$. But this strategy would be not efficient for embedding data. For the purpose that p_0 always has the smallest gray value in the block, the following strategy is used in our approach:

(1) If there exist just one pixel with the smallest gray value in a block, p_0 is assigned to the pixel with smallest gray value in the block.
(2) If there exist more than one pixel with the smallest gray value in a block, p_0 is assigned to the first pixel of those candidates in the sequence $p_{i,j}$, $p_{i,j+1}$, $p_{i+1,j+1}$, and $p_{i+1,j}$.
(3) Pixels p_1, p_2, and p_3 are assigned to the remainder pixels of the block in the clockwise sequence, which begins at the pixel next to p_0 clockwise.
(4) The width of the first range R_1 is 1, i.e., $l_1 = 0$ and $u_1 = 0$.

It is easy to show that p_0, p_1, p_2, and p_3 can be recognized before and after embedding. The reason is that the set of candidates, which have the smallest gray value in a block, is the same before and after embedding. Therefore, p_0 can be recognized, and so are p_1, p_2, and p_3. The embedding algorithm is shown as follows:

(1) Partition the host image into blocks of four neighboring pixels $p_{i,j}$, $p_{i,j+1}$, $p_{i+1,j+1}$, and $p_{i+1,j}$ in non-overlapping raster scan manner.

(2) For each block, the following steps are performed:

(a) Rename the pixels of the block as p_0, p_1, p_2, and p_3, whose corresponding gray values are g_0, g_1, g_2, and g_3.

(b) Create three groups *group*1 = (g_0, g_1), *group*2 = (g_0, g_2), and *group*3 = (g_0, g_3).

(c) Compute each group difference d_i, i = 1, 2, 3, and find out the range of index k_i, which the difference value d_i falls into.

(d) If one of k_i, i = 1, 2, 3, equals to r (the last range), abandon this block.

(e) By Eq. (5), calculate new differences d_1', d_2', and d_3'.

(f) Replace d_1, d_2, and d_3 with d_1', d_2', and d_3', respectively. Then, calculate the new pixel values g_0', g_1', g_2', and g_3' by the inverse calculation.

(g) Use the pixel-value shifting to shift the pixel values g_0', g_1', g_2', and g_3', such that the value of $(g_0-g_0')^2 + (g_1-g_1')^2 + (g_2-g_2')^2 + (g_3-g_3')^2$ is minimized.

An illustration of the data embedding process is shown in Fig. 3. In the figure, the pixel values of a sample block are assumed to be ($p_{i,j}$, $p_{i,j+1}$, $p_{i+1,j+1}$, $p_{i+1,j}$) = (150, 140, 143, 145). First, we find out the smallest gray value 140 of pixel $p_{i,j+1}$. Therefore, (p_0, p_1, p_2, p_3) are assigned to ($p_{i,j+1}$, $p_{i+1,j+1}$, $p_{i+1,j}$, $p_{i,j}$). Then, we distinguish three groups *group*1 = (g_0, g_1) = (140, 143), *group*2 = (g_0, g_2) = (140, 145), and *group*3 = (g_0, g_3) = (140, 150). The difference values are d_1 = (143 – 140) = 3, d_2 = (145 – 140) = 5, and d_3 = (150 – 140) = 10, where d_1 and d_2 are in the range of 1 through 8 and d_3 is in the range of 9 through 16. Both widths of these two ranges are $8 = 2^3$, which means that each group can be embedded 3 bits of the secret data. Assume that the secret data are 1001101011. The bit stream embedded into *group*1 is 100, *group*2 is 110 and bit stream 101 is embedded into *group*3. In Eq. (5), we calculate the new differences d_1' = 1+ $(100)_2$ = 5, d_2' = 1 + $(110)_2$ = 7, and d_3' = 9 + $(101)_2$ = 14. Finally, new pixel values *group*1 = (g_0', g_1') = (140, 145), *group*2 = (g_0', g_2') = (140, 147), and *group*3 = (g_0', g_3') = (140, 154) are calculated by the inverse calculation of the embedding algorithm in Step 2(f). To enhance the image quality, we use the pixel-value shifting resulting in new pixel values (g_0', g_1', g_2', g_3') = (138, 143, 145, 152). Note that sum of square errors reduces from 24 to 8.

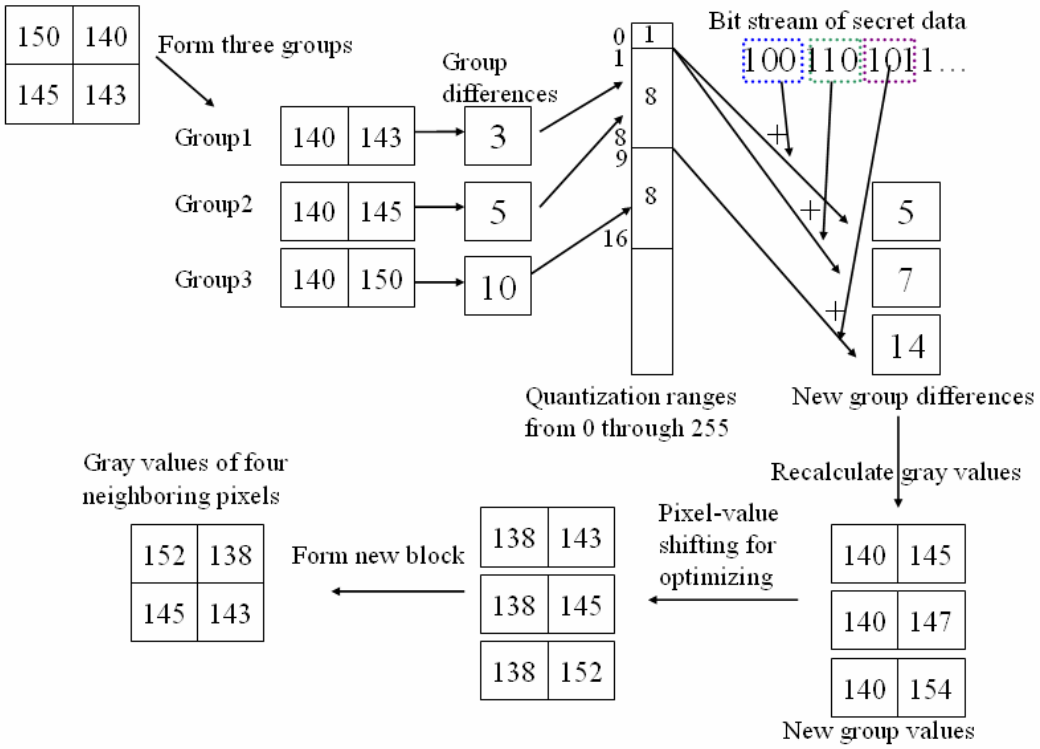

Fig. 3. An illustration of the data embedding process.

3.3: EXTRACTING SCHEME

The process of extracting the embedded data is similar to the embedding process with the same traversing order of blocks. First, we rename the pixels of a block as (p_0, p_1, p_2, p_3). If a block is embedded, all values of d_i (i = 1, 2, 3), k_i (i = 1, 2, 3), n_i (i = 1, 2, 3), *group*1, *group*2, and *group*3 are found out of this block. Note that the calculation of those values is the same to the embedding process, except that the block comes from

stego-images now. The bit-stream values b_1, b_2, and b_3, which are embedded in this block, are then extracted using the following equations:

$$b_i = d_i - l_{ki}, \text{ where } i = 1, 2, 3. \quad (6)$$

Each value b_i is transformed into a bit stream with n_i bits.

4: EXPERIMENTAL RESULTS

In the experiments, we use four host images Lena, Baboon, Peppers, and Toys with size 512 × 512. We employed the peak signal-to-noise ratio (PSNR) as a measure of the stego-image quality. Also, we used a random bit stream as the embedding data. Besides, we set the widths of 1, 8, 8, 16, 32, 64, and 127 in the experiments. The value of PSNRs and capacities are all the averages of the results created by 100 times executing of differently random bit streams.

Fig. 4 shows two of host images in our experiments. Fig. 5(a) shows the stego-images, and Fig. 5(b) and 5(c) show the corresponding enhanced difference images between the stego-images and the host images (with the differences of gray values being scaled eight times). Fig. 5(b) does not use the minimized pixel-value shifting, but Fig. 5(c) does. From the enhanced difference images, it can be seen that most of the distortions are found on the edges of the images. Table 1 shows the embedding capacities and PSNRs of our approach. "Before PSNR" are the PSNR values without using the minimized pixel-value shifting, and "Adjust PSNR" are the PSNR values with using this skill. Table 2 shows the results of comparisons of various methods in terms of PSNR and embedding capacity. The widths of ranges are 8, 8, 16, 32, 64, and 128 for Wu and Tsai's differencing method. Simple 2-bit LSB substitution approach have large capacities and best PSNR values, however, it can not pass the test of the programming detection [7]. All other approaches can pass the programming detection, and our approach has largest capacities in them. The PSNRs of ours are also accepted to human's eyes.

Table 1. The capacities and PSNRs for embedding random bit stream by our approach.

	Capacity	Before PSNR	Adjust PSNR
Lena	561,740	37.12	38.76
Baboon	691,735	30.53	32.79
Peppers	562,249	37.40	38.85
Toys	577,003	34.94	37.41

Table 2. The capacities and PSNRs for various cover images and various approaches.

Cover image	2-bit LSB			Differencing			Side-match			Our approach		
	Capacity	PSNR	Test	Capacity	PSNR	Test	Capacity	PSNR	Test	Capacity	PSNR	Test
Lena	524,288	44.23	NO	406,632	41.71	YES	398,227	40.62	YES	561,740	38.76	YES
Baboon	524,288	44.53	NO	437,806	39.14	YES	669,992	34.58	YES	691,735	32.79	YES
Peppers	524,288	44.49	NO	401,980	40.39	YES	390,664	40.67	YES	562,249	38.85	YES
Toys	524,288	44.45	NO	406,656	39.93	YES	426,536	37.30	YES	577,003	37.41	YES
Average	524,288	44.43		413,268	40.29		471,354	38.29		598,182	36.96	

(a) (b)

Fig. 4. The host images with size 512 × 512. (a) Lena, and (b) Baboon.

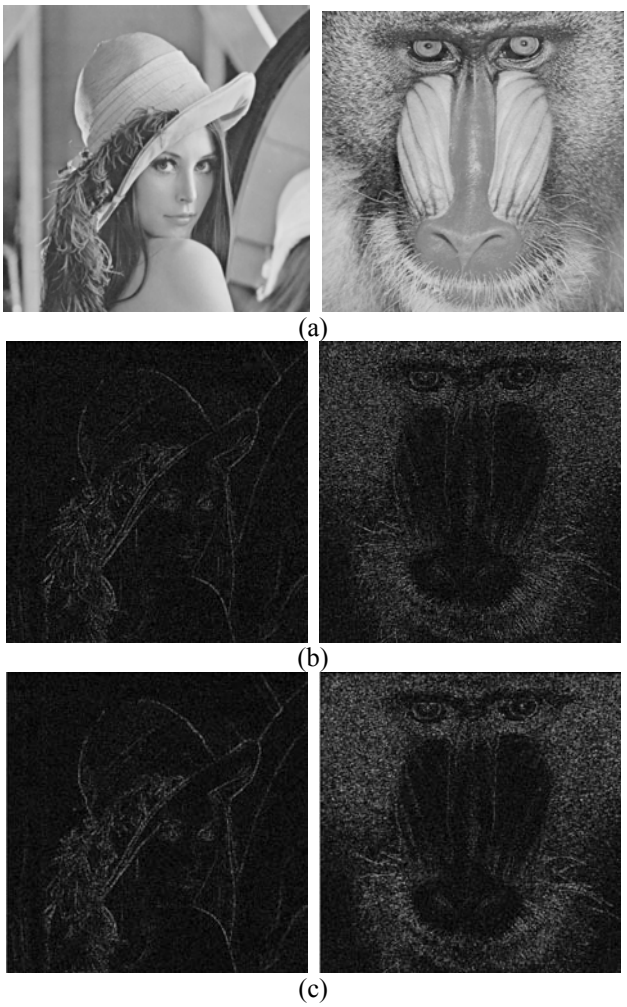

Fig. 5. The embedding results using the set of widths 1, 8, 8, 16, 32, 64, and 127. (a) The stego-images, (b) the enhanced difference images without using minimized pixel-value shifting, (c) the enhanced difference images with using minimized pixels-value shifting.

5: CONCLUSIONS

A new and efficient steganographic method for embedding secret messages into images without producing perceptible distortions has been proposed. Our approach extends the pixel-value differencing approach by processing a four-pixel block at a time. Moreover, more pixel-value differences are considered in our approach. In order to increase image quality, we also propose a skill of pixel-value shifting to optimize the quality of the stego-image. The experimental results show that our method has a larger capacity and can pass the detection of programs. Also, the PSNRs of ours are still acceptable to human's eyes. Our approach uses the concept of human visual system efficiently. Beside, there is no need referencing the host image to extract embedded data.

REFERENCES

[1]. Artz, D.: Digital Steganographic: Hiding Data within Data. IEEE Internet Comput., Vol. 5. (2001) 75–80.

[2]. Chang, C.C., Hwang, K.F., Hwang, M.S.: Robust Authentication Scheme for Protecting Copyrights of Images and Graphics, IEEE Proceedings on Vision, Image and Signal Processing Vol. 149, No. 1. (2002) 43-50.

[3]. Lee, Y.K., Chen, L.H.: High Capacity Image Steganography. IEE Proceedings on Vision Image, and Signal Processing, Vol. 147, No. 3. (2000) 288–294.

[4]. Chang, C.C., Lin, C.Y., Wang, Y.Z.: New image Steganographic Methods using run-length approach. Information Sciences, Vol. 176, No. 22. (2006) 3393-3408.

[5]. Wang, R.Z., Lin, C.F., Lin, J.C.: Image Hiding by Optimal LSB Substitution and Genetic Algorithm. Pattern Recognition, Vol. 34, No. 3. (2000) 671–683.

[6]. Chan, C.K., Chen, L.M.: Hiding Data in Images by Simple LSB Substitution. Pattern Recognition, Vol. 37, No. 3. (2004) 469–474.

[7]. Fridrich, J., Goljan, M., Du, R.: Reliable Detection of LSB Stegnography in Grayscale and Color Images. In: proc. ACM Workshop on Multimedia and Security. (2001) 27–30.

[8]. Wu, D.C., Tsai, W.H.: A Steganographic Method for Images by Pixel-Value Differencing. Pattern Recognition Letters, Vol. 24. (2003) 1613-1626.

[9]. Chang, C.C., Tseng, H.W.: A Steganographic Method for Digital Images Using Side Match. Pattern Recognition Letters, Vol. 25. (2004) 1431–1437.