Hindawi Publishing Corporation
EURASIP Journal on Advances in Signal Processing
Volume 2010, Article ID 430235, 13 pages
doi:10.1155/2010/430235

Research Article

A New Inverse Halftoning Method Using Reversible Data Hiding for Halftone Images

Jia-Hong Lee,[1] Mei-Yi Wu,[2] and Hong-Jie Wu[1]

[1] *Department of Information Management, National Kaohsiung First University of Science and Technology, Kaohsiung 811, Taiwan*
[2] *Department of Information Management, Chang Jung Christian University, Tainan 711, Taiwan*

Correspondence should be addressed to Mei-Yi Wu, barbara@mail.cjcu.edu.tw

Received 26 December 2009; Revised 28 July 2010; Accepted 5 October 2010

Academic Editor: Alex Kot

A new inverse halftoning algorithm based on reversible data hiding techniques for halftone images is proposed in this paper. The proposed scheme has the advantages of two commonly used methods, the lookup table (LUT) and Gaussian filtering methods. We embed a part of important LUT templates into a halftone image and restore the lossless image after these templates have been extracted. Then a hybrid method is performed to reconstruct a grayscale image from the halftone image. In the image reconstruction process, the halftone image is scanned pixel by pixel. If the scanned pattern surrounding a pixel appeared in the LUT templates, a gray value is directly predicted using the LUT value; otherwise, it is predicted using Gaussian filtering. Experimental results show that the reconstructed grayscale images using the proposed scheme own better quality than both the LUT and Gaussian filtering methods.

1. Introduction

Inverse halftoning is a process which transforms halftone images into grayscale images. Halftone images are binary images that provide a rendition of grayscale images and consists of “0” and “1”. It has been widely used in the publishing applications, such as newspapers, e-documents, and magazines. In halftoning process, it needs to use a kernel to carry out the conversion, and the common kernel is like Floyd-Steinberg kernel and is difficult to recover a continuous-tone image through halftone manipulation, conversion, compression, and so forth. In the past few years, many efficient inverse halftoning algorithms have been proposed, but there is no way to construct a perfect gray image from the given halftone image. There exist several inverse halftoning methods, including kernel estimation [1], wavelet [2], filtering [3, 4], and set theoretic approaches [5]. Most of these methods can obtain good reconstruction image quality but require relatively high computational complexity.

The halftoning and inverse halftoning processes can be regarded as the encoding and decoding processes of vector quantization. Therefore, the codebook design methods can be applied to build the inverse halftoning lookup tables [6, 7]. The content of a table entry is the centroid of the input samples that are mapped to this entry. The results are optimal in the sense of minimizing the MSE for a given halftone method. Although the table lookup method has the advantages of good reconstructed quality and fast speed, it faces the empty cell problem in which no or very few training samples are mapped to a specific halftone pattern.

In this paper, a reversible data hiding scheme for halftone images is proposed to embed specified information to improve the LUT-based inverse halftoning method. We embed a part of important LUT templates into a halftone image and generate a stego image in the data embedding process. Then, we can restore the halftone image without any distortion from the stego image after these templates have been extracted. Finally, we can obtain higher quality reconstructed images than the traditional LUT method by performing the proposed hybrid method with the extracted templates and the halftone image. The rest of the paper is organized as follows. Section 2 introduces related works about inverse halftoning methods and reversible data hiding methods for binary images. Section 3 presents the proposed reversible data hiding method for halftone images and the proposed hybrid method for inverse halftoning. Section 4 shows the experimental results and discussions, and the final section summarizes this paper.

2. Related Works

In this section, we introduce the methods which are related to inverse halftoning techniques including LUT-based and the Gaussian filtering methods. In addition, recently used reversible data hiding techniques for binary images are also introduced.

2.1. LUT-Based Method. The LUT-based method includes two procedures: the LUT buildup and the LUT-based inverse halftoning (LIH) procedures. The LUT buildup procedure is to build the LUT information by scanning selected grayscale images and their corresponding halftone images with a 3×3 and 4×4 template. Figure 1 shows the 3×3 and 4×4 templates with symbol X denoting the estimated pixel, respectively. Template is used as a sliding window to build up the LUT. In the LIH procedure, a grayscale image can be reconstructed from the given halftone image using the LUT information. The LUT information contains a set of pairs of binary pattern and its corresponding estimated gray value. Assume that there are L training image pair sets and $\{(O_i, H_i) \mid 1 \le i \le L\}$ represents the ith pair, where O_i denotes the ith original image and H_i the corresponding halftone image of O_i. The LUT buildup and LIH procedures using 4×4 template F are described as follows.

Procedure LUT Buildup

Step 1. Let arrays LUT[ ] and N[ ] be zero as the initial values. LUT[ ] is used to record the mapped gray value corresponding to a specific binary template with index I which appears in the input halftone image, and N[ ] is used to store the occurrence frequency of this specific binary template in the halftone image. Select L training grayscale images, and generate their corresponding halftone images, respectively.

Step 2. Select one image from the L training grayscale images. Scan the selected image and its corresponding halftone image in raster order with the template F. The index I for a pixel X can be calculated using (1), where k represents different locations on the template F. Since there are totally 16 locations on the template F, the value of I ranges from 0 to 65535. Then, the sum of the template occurrence frequencies and the sum of the gray values on the image pixel X can be computed as (2):

$$I = \sum_{k=0}^{15} 2^K P_k, \tag{1}$$

$$\begin{aligned} N[I] &= N[I] + 1, \\ \mathrm{LUT}[I] &= \mathrm{LUT}[I] + \mathrm{X}. \end{aligned} \tag{2}$$

Step 3. Repeat Step 2, until all L images are selected.

Step 4. The predicted gray value for a specified 4×4 pattern with indexed I can be computed as

$$\mathrm{LUT}[I] = \frac{\mathrm{LUT}[I]}{N[I]}. \tag{3}$$

Figure 2(a) shows an example to build LUT by performing Step 2 of LUT buildup algorithm.

Procedure LIH

Step 1. Perform the above-mentioned LUT buildup algorithm to build LUT.

Step 2. Scan a halftone image in raster order with template F, and compute the template index I using (1). The estimated gray value on pixel X can be obtained and denoted as $\mathrm{X} = \mathrm{LUT}[I]$.

Step 3. Output the estimated grayscale image.

Figure 2(b) shows an example to build LUT by performing of LIH algorithm.

Note that, some binary patterns in the input halftone image may not exist in the training images. In this situation, we will apply filters to estimate the mean gray pixel.

Though the LUT-based inverse halftoning method is easily implemented, there exists a disadvantage of this method, the constructed LUT information must be sent to the receiver.

2.2. Gaussian Filtering Method. Gaussian filtering is a smoothing algorithm for images. Equation (4) denotes a 2D Gaussian function, and σ is the standard deviation of a Gaussian distribution. In the implementation of inverse halftoning using Gaussian filtering, the binary pixel value 0 and 1 in the input halftone image will be regarded as 0 and 255, respectively. A weight mask with specified size and contents should be determined according to the use of Gaussian distribution with parameter σ. In the inverse halftoning process, the halftone image is scanned pixel by pixel in a raster order by moving the sliding mask. The output gray value on the corresponding central pixel of the mask is estimated via the summation of the weight value on the mask content multiplying by the binary values on the neighboring pixels. If the value of σ is larger, the resulting image will be more smoothing. In the inverse halftoning process using Gaussian filtering, the following equation of 2D Gaussian distribution is generally used to determine the mask:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}. \tag{4}$$

2.3. Reversible Data Hiding for Binary Images. Reversible data hiding can embed secret message in a reversible way. Relatively large amounts of secret data are embedded into a cover image so that the decoder can extract the hidden secret data and restore the original cover image without any distortion. Recently, a boundary-based PWLC method has been presented [8]. This method defines the same

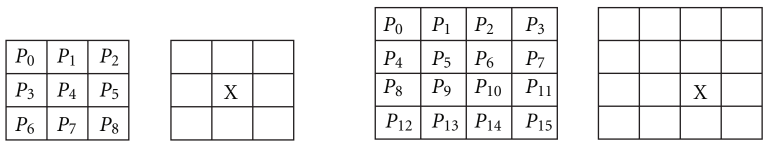
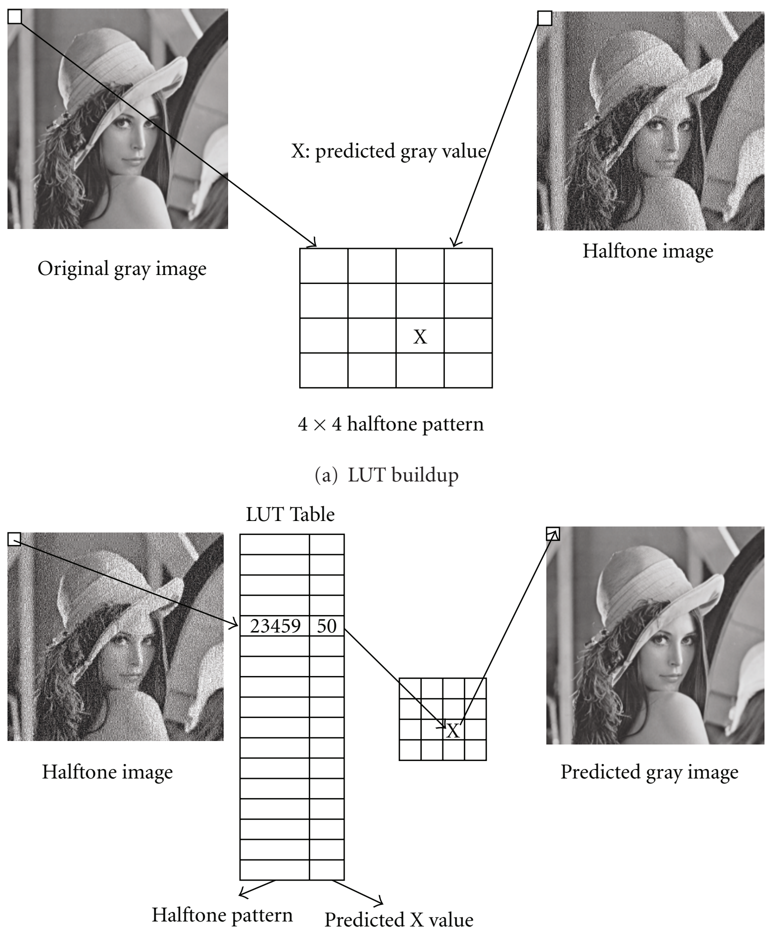


Figure 1: The 3 × 3 and 4 × 4 templates with symbol X denoting the estimated pixel, respectively.



(b) LUT-based inverse halftoning

Figure 2: LUT-based method includes two procedures (a) LUT Buildup (b) LUT-based inverse halftoning (LIH).

continuous 6 edge pixels as an embeddable block through searching for binary image edges. And then one can embed data in the pair of the third and fourth edge pixels. A reversible data hiding method for error-diffused halftone images is proposed [9]. This method employs statistics feature of pixel block patterns to embed data and utilizes the HVS characteristics to reduce the introduced visual distortion. The method is suitable for the applications, where the content accuracy of the original halftone image must be guaranteed, and it is easily extended to the field of halftone image authentication. However, these two methods have a drawback that the capacity of data hiding is still limited.

3. Proposed Method

The proposed inverse halftoning method based on reversible data hiding techniques can be divided into two phases: the embedding process and the extracting process. Figure 3(a) shows the diagram of the proposed method. In the embedding process, a grayscale image is transferred into a halftone image by error diffusion process. Then pattern selection is performed to determine the pattern pairs for the use in reversible data hiding. Meanwhile, a part of LUT templates are selected to keep high quality of recovery images in the reconstruction process. These templates along with the pattern pairs will be encoded in bit streams and embedded

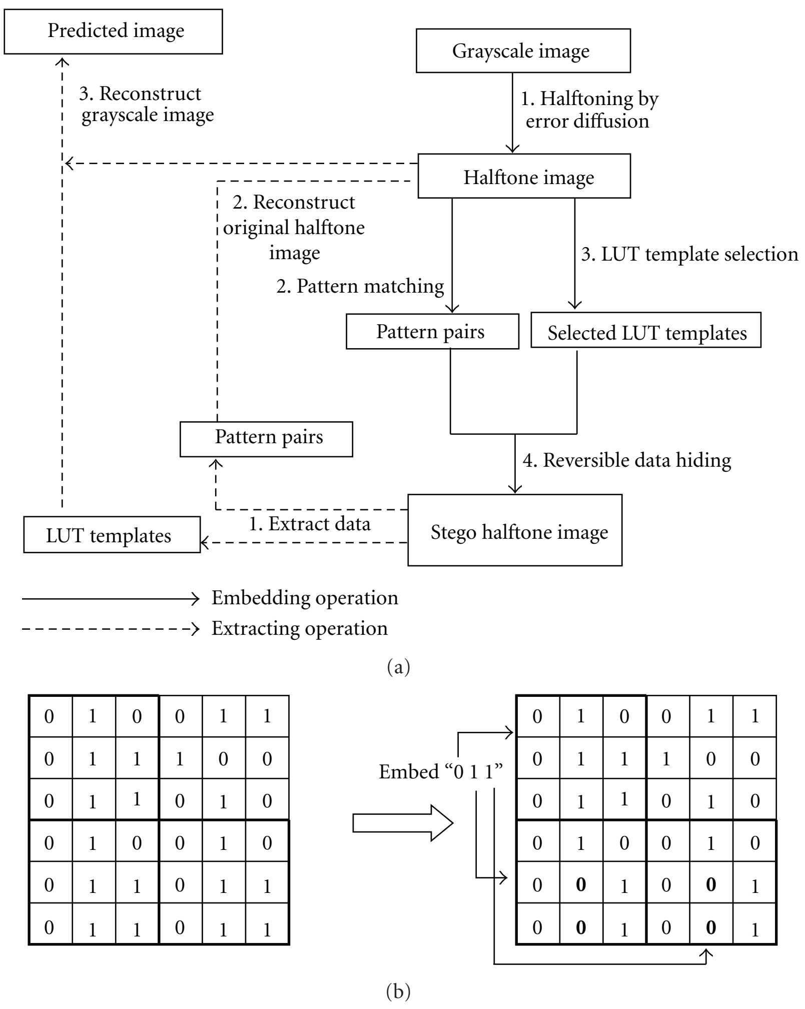


Figure 3: The embedding and extracting diagram and an example for the proposed method: (a) the embedding and extracting diagram of proposed method, (b) an example to illustrate the process of data embedding using pattern substitution.

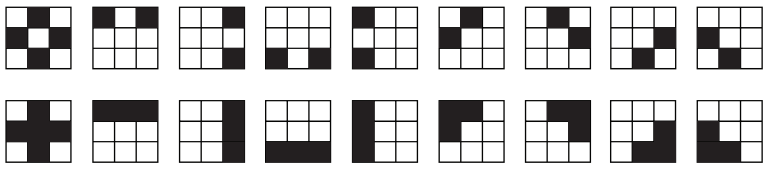


Figure 4: Bad human visual effects caused by pattern substitution during the data embedding process.

TC	PH^0	PL^0	…	PH^{TC-1}	PL^{TC-1}

Figure 5: The secret header of SH.

into the halftone image. The data embedding operation is performed based on pattern substitution. In the data extracting process, the pattern pairs and LUT templates are first extracted. The halftone image can be losslessly restored after the data extraction. Finally, we can reconstruct a good quality grayscale image from the halftone one with the aid of LUT templates. The proposed scheme has the advantages of two commonly used methods, the lookup table (LUT) and Gaussian filtering methods. We embed a part of important LUT templates into a halftone image and restore the lossless image after these templates had been extracted.

3.1. Data Hiding with Pattern Substitution for Halftone Images. The proposed method of reversible halftone data hiding technique uses pattern substitution method to embed and extract data into halftone images. The original image is partitioned into a set of nonoverlapping 3×3 blocks. There are totally 2^9 different patterns. Therefore, each pattern is uniquely associated with an integer in the range of 0 to 511. In most cases, many patterns never appear in an image.

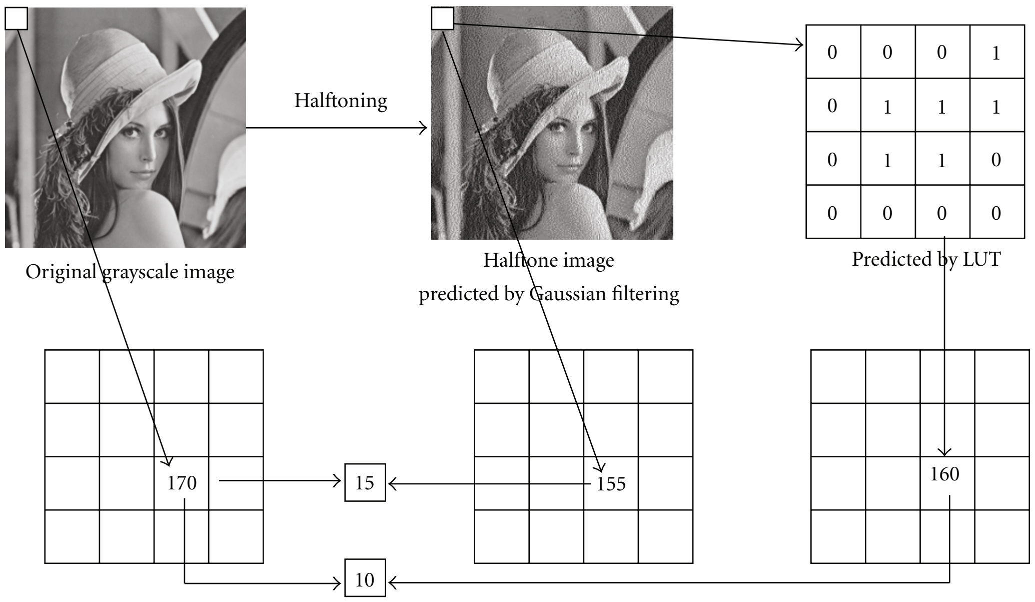


Figure 6: An example for the comparison of image quality loss using two different methods.

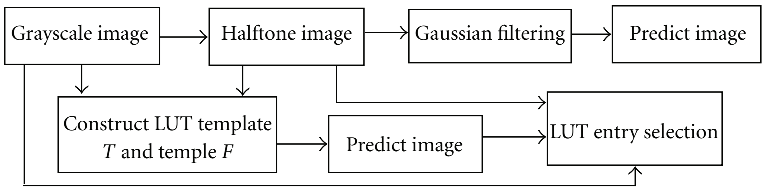


Figure 7: The flowchart of LUT entry selection.

In this study, all patterns are classified into two groups, *used* and *unused*. For each used pattern A, an unused pattern B, its content is the most closest to pattern A, will be selected to form a pair for data embedding. In the data embedding process, the original halftone image is partitioned into a group of 3×3 nonoverlapping patterns. Then, any pattern p on the halftone image with the same content of A will be selected to embed 1-bit data. If a data bit "0" is embedded on p, then the content of p remains as A. If a data bit "1" is embedded on p, then the content of p is updated as the content of pattern B. This scheme works because patterns A, B look similar. In data extraction process, the embedded message is obtained depending on the patterns A, B when the test image is scanned. For example, assume that the highest frequent pattern in the image is $\mathrm{PH} = 010, 011, 011$ and its corresponding unused pattern is $\mathrm{PL} = 010, 001, 001$. We can embed three secret data bits (e.g., 011) into the following 6×6 image block with the proposed pattern substitution method. The image is firstly divided into four nonoverlapped 3×3 patterns, and these patterns are scanned horizontally from top to buttom. If the content of the pattern PH is encountered, then check the bit value which is currently embedded. If the bit value is "0," then keep the content as PH. If the value is "1," then we replace PH with the PL pattern. This example of data embedding is shown in Figure 3(b).

To achieve a higher capacity of embedding data, more pattern pairs should be determined, whose steps can be presented as follows.

(1) Partition the original image into nonoverlapping 3×3 blocks.

(2) Compute the occurrence frequencies for all appeared patterns. Sort these used patterns decreasingly, and denote them as PH^i according to their occurrence frequencies. For instance, PH^0 is the pattern with the highest occurrence frequency.

(3) Find out all unused patterns. Assume that there are totally TC unused patterns; TC pairs of patterns (PH^i, PL^i) are selected to perform the data embedding operation, where the distance of pattern pair (PH^i, PL^i) owns the minimal distance. Based on the raster scan order, pattern PH and PL can be denoted as 9 elements PH_0, PH_1, $\mathrm{PH}_2,\ldots$, PH_8 and PL_0, PL_1, $\mathrm{PL}_2,\ldots$, PL_8, respectively. The calculation of pattern similarity for pattern PH and PL can be defined using the following distance equation:

$$\mathrm{Dist}(\mathrm{PH},\mathrm{PL}) = \sum_{j=0}^{8} \left\| \mathrm{PH}_j - \mathrm{PL}_j \right\|, \quad (5)$$

where j is the location in the 3×3 block.

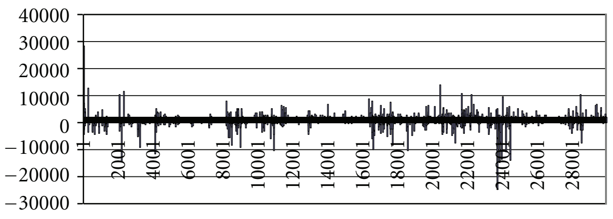


FIGURE 8: The sum of difference $B[1] \cdots B[30000]$ for Lena image.

DT	P	LT^0	LI^0	$\ldots$	LT^{P_1-1}	LI^{P_1-1}

FIGURE 9: LUT information header LH.

(4) Search all blocks in the original image. As long as we come across a pattern in the PH^i, if a bit "0" is embedded, the block remains as PH^i; otherwise, the block is updated as the pattern PL^i.

The maximum embedding capacity of the proposed data embedding method can be denoted as $= \sum_{i=0}^{\mathrm{TC}-1} \mathrm{Freq}[\mathrm{PH}^i]$, where $\mathrm{Freq}[\mathrm{PH}^i]$ represents the occurrence frequency of pattern PH^i in the image and TC is the number of selected pairs.

However, the image quality of stego image generated using the proposed method is not very well in the visual effect. To consider human visual effect, we should take notice about some situations which will cause "congregation" effect around the center, corners, or lines on the 3×3 block. These cases are displayed in Figure 4. To avoid these cases when a pattern replacement occurs, we apply the following equation to replace (5):

$$\mathrm{Dist}(\mathrm{PH},\mathrm{PL}) = \sum_{j=0}^{8} \left\| \mathrm{PH}_j - \mathrm{PL}_j \right\| + \sum_{j=0}^{8} \mathrm{weight}\left(\mathrm{PH}_j, \mathrm{PL}_j\right), \tag{6}$$

where

$$\mathrm{weight}\left(\mathrm{PH}_j, \mathrm{PL}_j\right) = \begin{cases} 1, & \text{if} \begin{cases} \mathrm{PH}_j = 1, \mathrm{PH}_{j-1} = \mathrm{PH}_{j+1} = \mathrm{PL}_j = 0, & j = 1,4,7, \\ \mathrm{PH}_j = 1, \mathrm{PH}_{j-3} = \mathrm{PH}_{j+3} = \mathrm{PL}_j = 0, & j = 3,4,5, \\ \mathrm{PH}_j = 1, \mathrm{PH}_{j+1} = \mathrm{PH}_{j+3} = \mathrm{PL}_j = 0, & j = 0, \\ \mathrm{PH}_j = 1, \mathrm{PH}_{j-1} = \mathrm{PH}_{j+4} = \mathrm{PL}_j = 0, & j = 2, \\ \mathrm{PH}_j = 1, \mathrm{PH}_{j-3} = \mathrm{PH}_{j+1} = \mathrm{PL}_j = 0, & j = 6, \\ \mathrm{PH}_j = 1, \mathrm{PH}_{j-3} = \mathrm{PH}_{j-1} = \mathrm{PL}_j = 0, & j = 8, \end{cases} \\ 0, & \text{otherwise}. \end{cases} \tag{7}$$

The matching pairs of PH^i and PL^i should be stored for the recovery and denoted as the Secret Header (SH) with size of $8 + \mathrm{TC} * 18$ bits. Therefore, we should embed the secret header SH to the cover halftone images. Figure 5 is the secret header SH, and the data hiding process for SH will be discussed in Section 3.3.

3.2. The Determination of Important LUT Templates. The proposed method is a kind of hybrid inverse halftoning method which has the advantages of Gaussian filtering and LUT methods. For a small image block which is the same size as the used template size in a halftone image, if the difference between the predicted value and the original real gray value using Gaussian filtering method is larger than the difference using LUT method with a specified template, it means that LUT method can obtain a better result than Gaussian filtering on the image block. Figure 6 shows an example for the comparison of image quality loss using these two methods.

But it does not guarantee that the LUT method with this template always works better in other image blocks than using Gaussian filter. So we should sum up the difference values for a specified template to all image blocks on the halftone image. If the sum of differences with LUT method is smaller than the sum with Gaussian filter, then this template is worth being recorded and embedded. This means that the LUT template can obtain a higher image quality than using Gaussian filtering method in the image recovery process. However, only a part of important templates which save larger quality loss are selected to embed since the embedding capacity is limited for a halftone image. In the grayscale image recovery process, we scan the halftone image by checking the templates. If the current template is one of the embedded templates, then LUT is used to predict the gray value; otherwise, Gaussian filtering method is applied to predict the value. Figure 7 displays the flowchart of the LUT entry selection.

implementation, 3×3 and 4×4 templates are considered for the LUT method. We introduce the operating procedure of using the proposed method with a 4×4 template as bellows, and the case with a 3×3 template is similar.

Procedure LUT Temple Selection with a Template of Size 4×4

Step 1. Perform the LUT buildup procedure, and proceed to train the input original grayscale and halftone images.

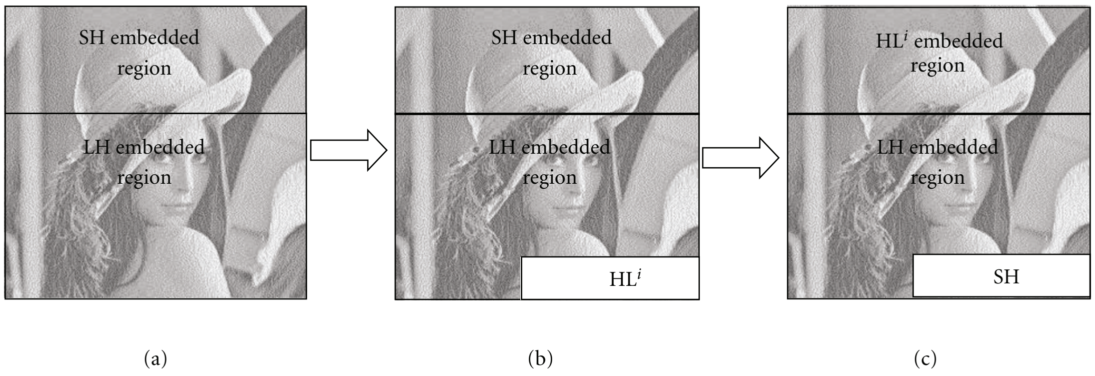


FIGURE 10: The flowchart of data embedding process: (a) embed SH and LH into the halftone image in horizontal scanning, and generate a stego image S; (b) extract data from the last row of stego image S, and denote it as HL_i; (c) HL_i is then embedded into the stego image S with pair-based method, and generate another stego image S'.

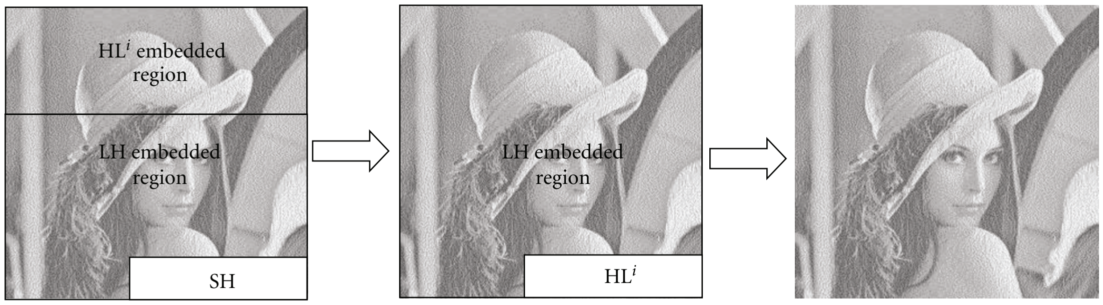


FIGURE 11: The flowchart of data extracting process.

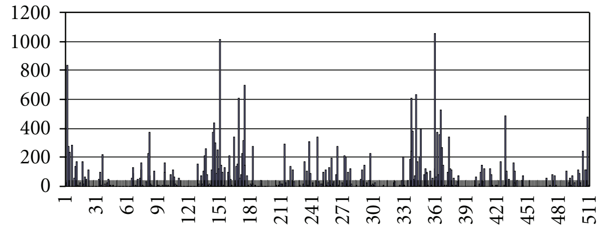


FIGURE 12: The pattern histogram of halftone image Lena.

Apply the LIH procedure to an input halftone image H, and generate a corresponding predicted image H'.

Step 2. Perform the inverse halftoning using Gaussian filtering for image H, and generate a predicted image G'.

Step 3. Compute the difference of the absolute values of G minus G' and G minus H' using the following equation:

$$D(x,y) = |(G(x,y) - G'(x,y))| - |(G(x,y) - H'(x,y))|. \tag{8}$$

If the D value is greater than zero, it means that the LUT method can obtain a better predicted value than Gaussian method on the pixel with location (x, y).

Step 4. Scan the image block by block with block size 4×4. For a processed block, if (x, y) is the predicted position and the image block on the halftone image is with index I, then the sum of difference $B[I]$ can be accumulated as

$$B[I] = B[I] + D(x,y). \tag{9}$$

$B[I]$ can be regarded as the quality improvement of the predicted image by replacing Gaussian filtering with the LUT method.

Step 5. Sort $B[\,]$ decreasingly, and generate the corresponding templates index $\mathrm{SI}[i]$, where $0 \le i \le 65535$. Since the embedding capacity is limited, we can only embed part of the top LUT templates into the halftone image. The total quality improvement is denoted as $\sum_{i=0}^{P-1} B[\mathrm{SI}[i]]$, where P represents the number of embedded templates.

Figure 8 shows an example of the sum of differences $B[I]$ for Lena image, where x-axis represents the pattern index I in the halftone image with 4×4 templates and y-axis represents the $B[I]$ value. If $B[I]$ is greater than zero, it means that the LUT-based method works better than Gaussian filtering method on the pixel value prediction under all image blocks with the same context of the template indexed by I.

Assume that P_1 is the number of parts of top LUT templates to be embedded into the halftone image using 3×3 templates, and P_2 is the number of parts of top LUT

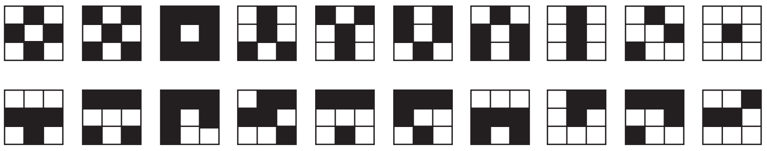


Figure 13: An example of PH^i (first row) and PL^i (2nd row) obtained from the Lena image.

templates to be embedded using 4×4 templates, respectively. P_1 and P_2 can be computed according to (10) and (11):

$$P_1 = \left\lceil \frac{\sum_{i=0}^{\mathrm{TC}-1} \mathrm{Freq}\left[\mathrm{PH}^i\right] - 8 - \mathrm{TC}\times 18 - 1 - 10}{17} \right\rceil, \quad (10)$$

$$P_2 = \left\lceil \frac{\sum_{i=0}^{\mathrm{TC}-1} \mathrm{Freq}\left[\mathrm{PH}^i\right] - 8 - \mathrm{TC}\times 18 - 1 - 10}{24} \right\rceil. \quad (11)$$

In (9), as mentioned above, parameter TC represents the number of pairs used in data embedding process. The maximum value of TC is 256 when 3×3 templates are applied, and it requires 8 bits to the storage. We also need $\mathrm{TC} \times 18$ bits to store the content of all matching pairs of templates and one bit to store the template type to discriminate the usage of 3×3 and 4×4 templates. If P represents the number of LUT templates which are embedded into the cover image, 10 bits are used to store P value in our experiments.

Finally, we can compare the two values of $\sum_{i=0}^{P_1-1} B[\mathrm{SI}[i]]$ and $\sum_{i=0}^{P_2-1} B[\mathrm{SI}[i]]$; if $\sum_{i=0}^{P_1-1} B[\mathrm{SI}[i]]$ is larger, it means that the quality improvement using 3×3 templates is better than the case of using 4×4 templates; otherwise, 4×4 templates are used in the data embedding process.

Assume that the template type is DT and the number of embedded LUT templates is P. The LUT information for each template should contain two parts, the template index LT^j and the predicted gray value LI^j, $0 \le j < P$. Figure 9 displays the LUT information and structure and is denoted as LH (LUT data header).

3.3. Overhead Information and Data Embedding. The overhead information includes two kinds of data; SH is the pattern pairs information (Figure 5) for data embedding, and LH is the important LUT template information (Figure 9) for improving the quality of recovery images. Since different images have different contents of SH and LH. We should embed the overhead information into the halftone image for image recovery. In the data embedding process, SH and LH are converted into a binary bit stream. Then SH is embedded into image and LH is embedded after the embedded SH. The embedding method is implemented by pattern matching approach according to the order of the image scanning, from top to bottom horizontally. If one pattern PH^i is encountered and the embedded bit value is currently 0, then the pattern PH^i is kept; otherwise, PH^i is replaced by PL^i. The output stego image in this stage is denoted as S.

Although the SH is embedded in the data embedding process, the receiver cannot extract data correctly from the stego image S since the receiver does not know the SH pair information before starting the pair-based extracting process. To overcome this problem, we define a region with the same size of SH length $8 + \mathrm{TC} * 18$ bits, and it is located in the last row of cover image. The pixel values on this region in the stego image are denoted as HL_i. Then we embed HL_i into stego image S using the proposed pattern substitution scheme again. In this stage, S is regarded as the cover image, and the pattern pair information PH is applied in the data embedding process. Finally, the bit stream of SH is then directly "paste" into the last row of S pixel by pixel, and a new stego image is generated and denoted as S'. Figure 10 shows the flowchart of data embedding; Figure 10(a) embeds SH and LH into the halftone image in horizontal scanning and generates a stego image S; Figure 10(b) extracts $8 + \mathrm{TC} * 18$ pixels from the last row of stego image S and is denoted as HL_i; in Figure 10(c), HL_i is then embedded into the stego image S with pair-based method and generated another stego image S'.

3.4. Data Extract and Recovering Grayscale Image. In data extracting process, we extract $8+\mathrm{TC} * 18$ bits of SH from the last raw of a stego image S' and get the pattern information of PH^i and PL^i from SH data. Then we start scanning the stego image from top to bottom. If PH^i is met, bit 0 is extracted; otherwise, bit 1 is extracted. The extracted length of bit stream is the same with SH's and denoted as HL_i, and HL_i is copied to the last raw of image to replace the content of SH. Then we can continue to scan image to extract the embedded secret header of LUT information as LH. All patterns of PL^i are replaced by PH^i in the extracting process to recover the original halftone image. Figure 11 displays the data extracting process and the original halftone image reconstruction.

Finally, we can reconstruct the grayscale images for the original halftone image using the proposed hybrid method. In the image recovery process, we first predict the grayscale image from the restored halftone image by Gaussian filtering method. Then we rescan the restored halftone image again to find the pattern with the same contents of the embedded LUT template LT_i. If this case is met, we will update the corresponding central pixel value in the predicted grayscale image with the value LI_i. Finally, a better quality of predicted grayscale image can be obtained.

4. Experimental Results

Four 512×512 error-diffused halftone images, "Lena," "Pepper," "Airplane," and "Baboon," are selected to test the performance of the proposed method. These halftone images are obtained by performing Floyd-Steinberg error diffusion

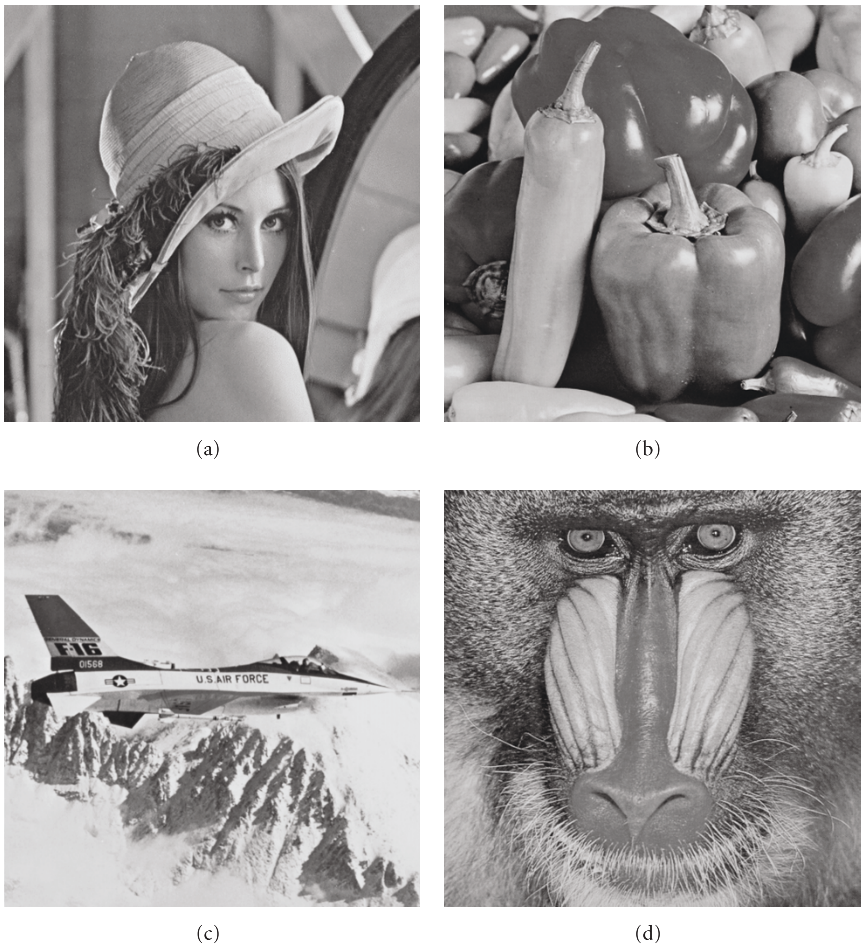


Figure 14: Four images for experiments: (a) Lena, (b) Pepper, (c) Airplane, and (d) Baboon.

filtering on the 8-bit gray-level images. Figure 12 shows the pattern histogram of the halftone image Lena with x-axis indicating the pattern index ranging from 0 to 511 and y-axis indicating the occurrence frequency of each pattern index. The highest peak among the histograms is with value 1058, and the number of zero value is totally 134. Figure 13 displays the top ten matching patterns which own the highest occurrence frequency in the halftone image Lena. In addition, we have also applied the proposed method on other images including Pepper, Baboon, and Airplane. Figure 14 shows the original grayscale images. Figures 15(a), 15(d), 15(g), and 15(j) are the generated halftone images from the images in Figure 14, respectively. Figures 15(b), 15(e), 15(h), and 15(k) are the generated stego images with 2072, 2329, 3086, and 2603 bits of data embedded, respectively. Figures 15(c), 15(f), 15(i), and 15(l) show the generated stego images with maximum capacity, respectively.

Figure 16 shows the difference between the generated stego images with and without applying the weight adjusting operation. Obviously, the stego image with weight adjusting operation owns better perceptive quality than the other one since the operation can reduce the possibility of forming black spots.

Table 1 shows the used templates, the number of selected pairs and embedding capacities using the proposed method for different images. The embedding capacities for image Lena, Pepper, and Airplane are all about 25000 bits. But the capacity for image Baboon is only 9544 bits. This is because the number of zero patterns is smaller than the other three images. It means that there is more different "texture" patterns existing in the Baboon halftone image. Due to the smaller embedding capacity, only few templates information are selected to embed into the halftone image. In the case for Baboon image, 3×3 template is used to be replaced by 4×4 template to obtain a better quality of reconstructed image.

To evaluate the performance of the proposed method, different inverse halftoning methods are used for comparison. They include the Gaussian filtering with parameter $\sigma = 1.41$, traditional LUT and Edge-based LUT methods with ten images for training. Table 2 shows the PSNR values for the recovery images using these different methods. Experimental results show that the reconstructed grayscale images using the proposed scheme own better quality than Gaussian filtering, traditional LUT, and Edge-based LUT methods. Figure 17 shows the experimental results for image Lena using different methods. The reconstructed

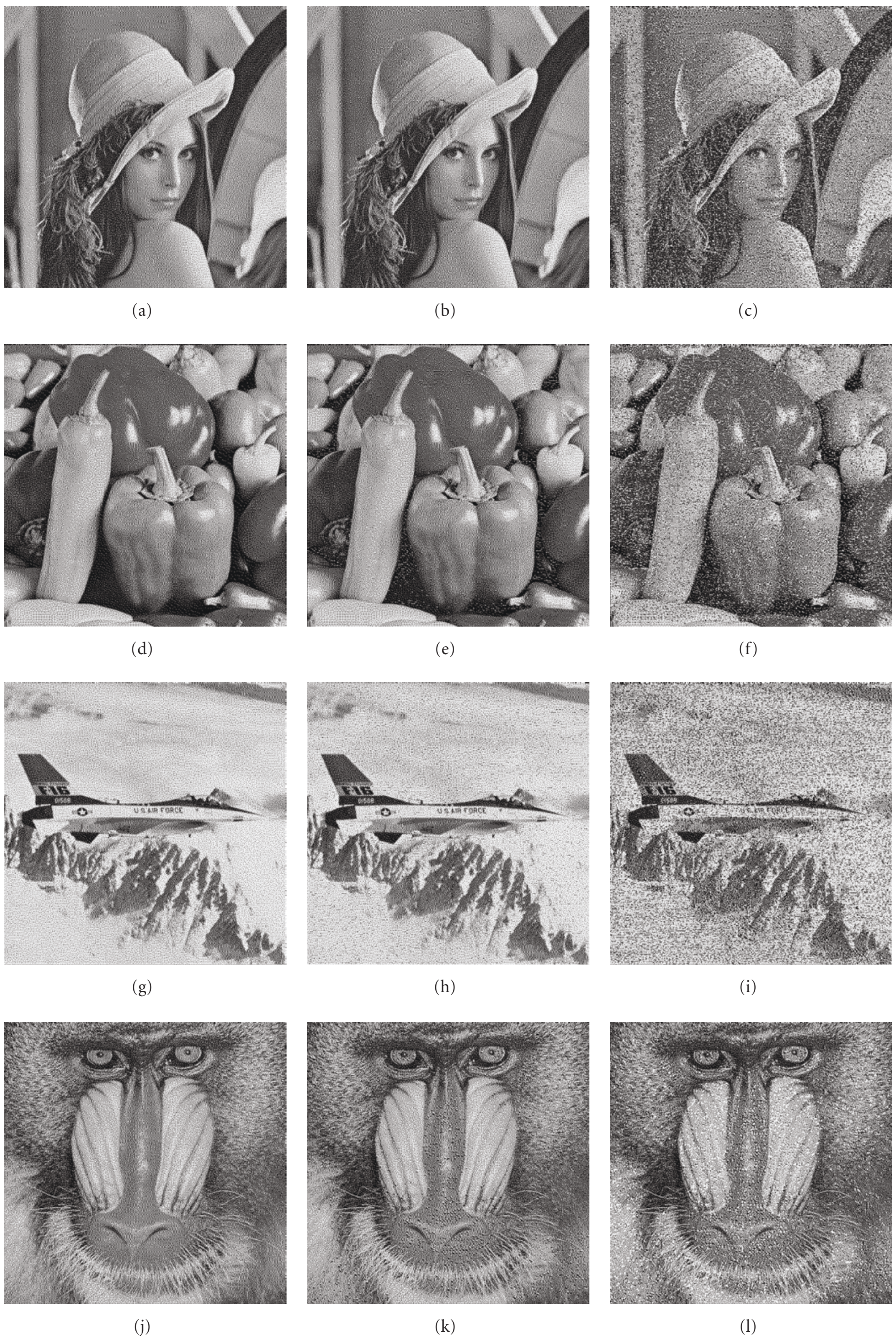


FIGURE 15: The stego images of data hiding using the proposed method: (a), (d), (g), and (j) the halftone image generated by error diffusion; (b) and (h) the stego images of embedding two pairs of templates; (e) and (k) the stego images of embedding three pairs of templates; (c), (f), (i), and (l) the stego image of hiding all pairs of templates.

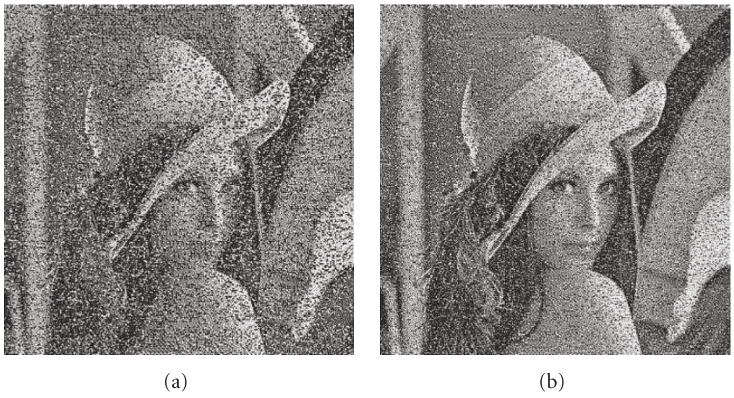


Figure 16: The generated stego images with and without applying the weight adjusting operation: (a) without the weight adjusting and (b) with the weight adjusting.

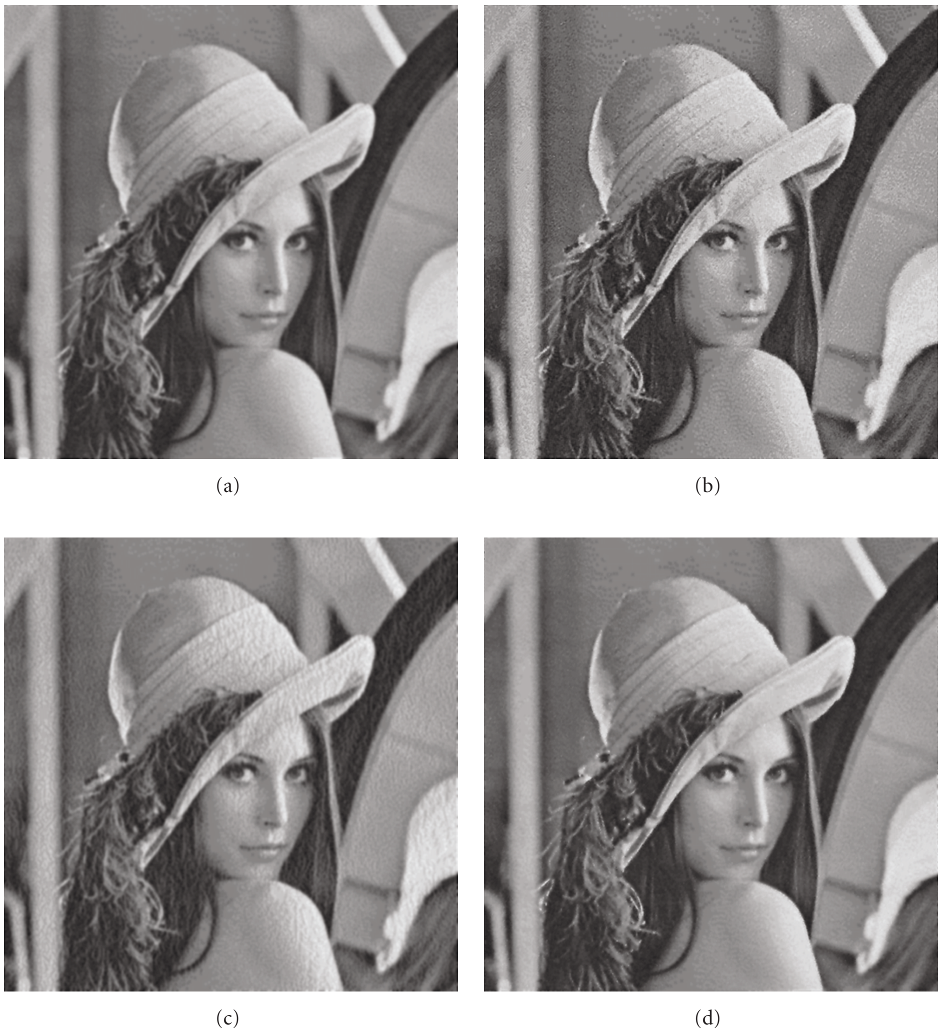


Figure 17: The inverse halftoning results using different methods: (a) the reconstructed image using Gaussian filtering; (b) the reconstructed image using LUT method; (c) the reconstructed image using ELUT method; (d) the reconstructed image using the proposed method.

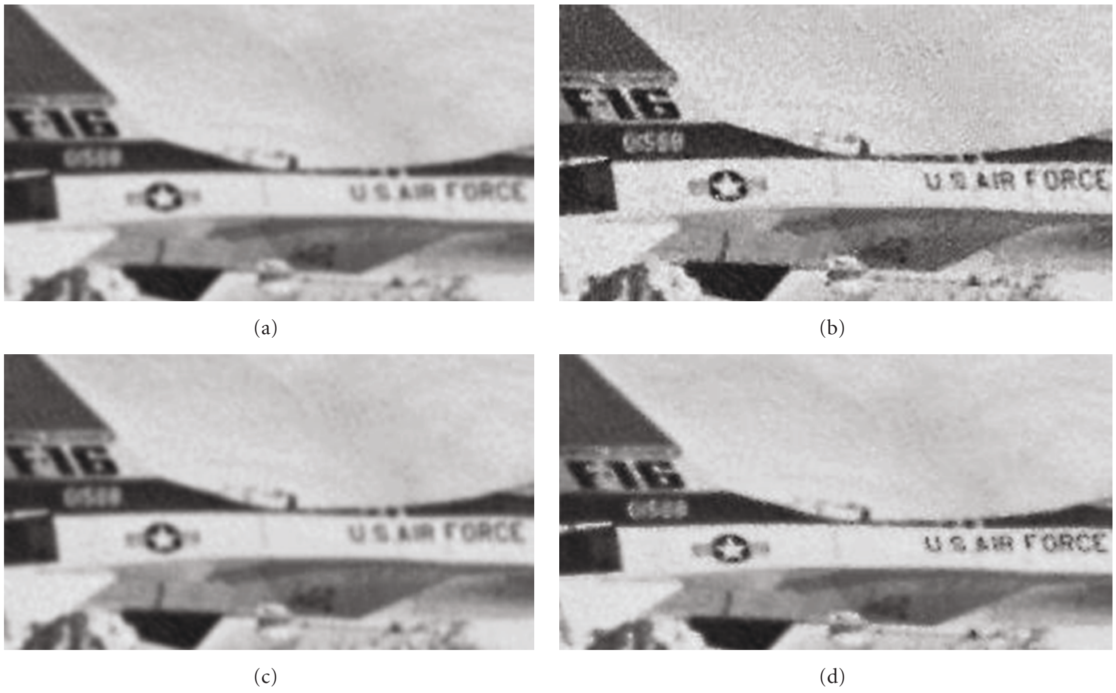


Figure 18: The inverse halftoning result of Airplane using different methods: (a) the reconstructed image using Gaussian filtering; (b) the reconstructed image using LUT method; (c) the reconstructed image using ELUT method; (d) the reconstructed image using the proposed method.

image using Gaussian filtering seems to be "blurring," and the reconstructed one using LUT and ELUT will show some noises on the image. In the experiments of LUT and ELUT, we used ten common images for training to obtain the corresponding LUT information. In the reconstruction process, if the pattern was not in the training LUT, we restored the predicted pixel by Gaussian filtering.

Figure 18 shows the results of image Airplane using different methods, and we extract a part of the reconstructed image to enlarge for comparison. We see that Gaussian filtering results in a visual pleasing but quite blurred image, while LUT and ELUT result in a noisy image. Table 2 displays the PSNR values of reconstructed images using different methods. The proposed method can increase 1.3 dB on PSNR than Gaussian filtering, 1.5 dB than LUT method, and 0.53 dB than ELUT method. It represents the high feasibility of the proposed method.

The used algorithms of LUT, ELUT, and Gaussian filtering in our experiments can be implemented in the raster fashion, the number of operations per pixel is all less than 500, and the time complexity of these methods can be denoted as $O(N^2)$ for the image of size $N \times N$. The proposed method can be actually regarded as a hybrid scheme of the LUT and Gaussian filtering methods, but it requires another image scan to extract the embedded important LUT information from the stego halftone image before performing the inverse halftoning. Therefore, the computational complexity of the proposed method is slightly higher than that of the three methods. But the total operation number per pixel using the proposed method is still less than 500. We compare the complexity for several different inverse halftoning methods, and the results are shown in Table 3. The computational complexity is estimated from algorithms given in the corresponding references. Low means a number fewer than 500 operations per pixel, median denotes 500–2000 operations per pixel, and high means more than 2000 operations required.

Table 1: The embedded capacity of LUT and maximum embedding capacity.

Images (embedded capacity)	Template	LUT pairs	Maximum capacity (bit)	PSNR (dB)
Lena (26317)	*4×4	995	26330	30.11
Lena (5146)	3×3	290	26330	29.06
Pepper (26575)	*4×4	1005	26585	30.11
Pepper (4995)	3×3	279	26585	30.15
Airplane (24985)	*4×4	974	24999	29.75
Airplane (6706)	3×3	386	24999	29.58
Baboon (9541)	4×4	380	9544	22.59
Baboon (8315)	*3×3	469	9544	23.99

* represents the selected temple for the proposed method.

5. Conclusions

A new inverse halftoning algorithm based on reversible data hiding techniques for halftone images is proposed in this

Table 2: The experiment result for four images using different methods.

Images (embedded capacity)	PSNR(dB)			
	Gaussian filtering* [10]	LUT [6]	ELUT [7]	The proposed method
Lena (26317)	29.40	27.53	28.54	30.11
Pepper (26575)	29.30	27.33	29.29	30.72
Airplane (24985)	28.29	26.94	28.05	29.75
Baboon (8315)	22.22	22.35	22.22	23.99

*Gaussian filter $\sigma = 1.41$.

Table 3: Complexity comparison of different inverse halftoning schemes.

Algorithms (ref.)	Complexity
MAP [11]	High
Wavelet [1]	Median
Gaussian filtering [10]	Low
LUT [6]	Low
ELUT [7]	Low
The proposed method	Low

paper. We embed a part of important LUT templates into a halftone image and restore the lossless image after these templates have been extracted. Then a hybrid method is performed to reconstruct a grayscale image from the halftone image. Experimental results show that the proposed scheme outperformed Gaussian filtering, LUT, and ELUT methods. The proposed method can be also modified by selecting different filtering methods for practical applications.

Acknowledgments

This work was supported by the National Science Council, Taiwan, under Grants 99-2220-E-327-001 and 99-2221-E-328-001.

References

[1] P. W. Wong, "Inverse halftoning and kernel estimation for error diffusion," *IEEE Transactions on Image Processing*, vol. 4, no. 4, pp. 486–498, 1995.

[2] Z. Xiong, M. T. Orchard, and K. Ramchandran, "Inverse halftoning using wavelets," in *Proceedings of the IEEE International Conference on Image Processing (ICIP '96)*, vol. 1, pp. 569–572, September 1996.

[3] Z. Fan, "Retrieval of images from digital halftones," in *Proceedings of the IEEE International Symposium on Circuits Systems*, pp. 313–316, 1992.

[4] T. D. Kite, N. Damera-Venkata, B. L. Evans, and A. C. Bovik, "A fast, high-quality inverse halftoning algorithm for error diffused halftones," *IEEE Transactions on Image Processing*, vol. 9, no. 9, pp. 1583–1592, 2000.

[5] P.-C. Chang, C.-S. Yu, and T.-H. Lee, "Hybrid LMS-MMSE inverse halftoning technique," *IEEE Transactions on Image Processing*, vol. 10, no. 1, pp. 95–103, 2001.

[6] M. Meşe and P. P. Vaidyanathan, "Look-up table (LUT) method for inverse halftoning," *IEEE Transactions on Image Processing*, vol. 10, no. 10, pp. 1566–1578, 2001.

[7] K.-L. Chung and S.-T. Wu, "Inverse halftoning algorithm using edge-based lookup table approach," *IEEE Transactions on Image Processing*, vol. 14, no. 10, pp. 1583–1589, 2005.

[8] C.-L. Tsai, H.-F. Chiang, K.-C. Fan, and C.-D. Chung, "Reversible data hiding and lossless reconstruction of binary images using pair-wise logical computation mechanism," *Pattern Recognition*, vol. 38, no. 11, pp. 1993–2006, 2005.

[9] J.-S. Pan, H. Luo, and Z.-M. Lu, "Look-up table based reversible data hiding for error diffused halftone images," *Informatica*, vol. 18, no. 4, pp. 615–628, 2007.

[10] S. Hein and A. Zakhor, "Halftone to continuous-tone conversion of error-diffusion coded images," *IEEE Transactions on Image Processing*, vol. 4, no. 2, pp. 208–216, 1995.

[11] R. L. Stevenson, "Inverse halftoning via MAP estimation," *IEEE Transactions on Image Processing*, vol. 6, no. 4, pp. 574–583, 1997.