Pergamon

Comput. & Graphics, Vol. 21, No. 6, pp. 757–767, 1997

Printed in Great Britain
0097-8493/97 $17.00+0.00

PII: S0097-8493(97)00055-1

Technical Section

EDGE DETECTION AND EDGE-PRESERVED COMPRESSION FOR ERROR-DIFFUSED IMAGES

WEI-YU HAN and JA-CHEN LIN†

Department of Computer and Information Science, National Chiao Tung University, 1001 Ta Hsueh Road, Hsinchu 30050, Taiwan, ROC
e-mail: jclin@cis.nctu.edu.tw

Abstract—In this paper, a new approach to edge detection and image compression of bilevel error-diffused images is proposed. The proposed approach is directly applied to the error-diffused images without any inverse halftoning technique. The main idea behind the proposed edge detection method is to compute the consistency value of each pixel of the error-diffused image, and the computed values are then clustered into two classes to obtain the desired edges. Here, the consistency value is a function of the mass center and geometric center of the window; more precisely, it represents the possibility that the area covered by the window can be uniform. As for compression, each error-diffused input image is compressed by dividing the image into non-overlapping blocks with size 8×8, then each highly-consistent block is encoded by its average illumination, and each lowly-consistent block is transmitted directly using the original bitmap. The threshold to distinguish these two kinds of blocks is automatically calculated based on the compression rate required by the users. The complexity of our compression technique is low, and this fast method can be used in real-time application. © 1997 Elsevier Science Ltd. All rights reserved

1. INTRODUCTION

A halftone image is a kind of binary image which visually preserves the gray aspect and image detail of the original gray level image. Halftone images are widely applied to the printing of industrial and electronic documents with pictures, especially when the printers (or facsimiles) available are two-level. Halftone images may be categorized roughly into two classes, namely, 1) clustered-dot halftone (*e.g.* clustered ordered-dither images), and 2) dispersed-dot halftone (*e.g.* error-diffused images). Excellent reviews of various halftoning techniques can be found in Refs. [1–3].

As is well-known, edge detection plays an important role in many image processing applications, such as image compression and feature extraction. Most of the existing edge detection methods, such as Roberts, Sobel, Prewitt, Laplacian [4], Valley [5] and some thresholding methods [6], perform reasonably well for gray-level images, but tend to fail for halftone images. Therefore, the first goal of this paper is to find a new edge detection method which can be applied directly to halftone images (the second goal is to use this new method to compress error-diffused [7] images).

In industry, the halftoning technique has been a build-in binary quantizer of FAX machines [8] used to convert a gray image to its binary version so that the receiver can get a gray-imitated binary image (*e.g.* error-diffused image). Recently, the facsimile function has been integrated with personal computers, commonly known as PC-FAX [9]. In offices, PC-FAX can be used on-line to store the received facsimile date, and process them off-line later (such as printing, display, archives and re-distribution to other stations in a client–server mode). It is often necessary to compress the received facsimile data which are error-diffused images so that the space required for archives can be reduced (when there are many facsimiles keep on coming in) and the transmission time used later for the re-distribution of the received error-diffused images can also be cut down. A variety of compression techniques [10–15] for halftone images have been suggested. Some of them do the compression on the gray-level domain; more precisely, these approaches require the inverse halftoning technique to convert the halftone image from its binary version to a (pseudo) gray-level version. However, these approaches (especially when they are applied to the dispersed-dot halftone) will lose detail information because the inverse halftoning techniques have the effect of blurring edges more or less (whereas human visual perception is sensitive to edge). Besides, compression with inverse halftoning techniques also will slow down the processing speed. We therefore try in this paper to preserve the edge information of the original error-diffused [7] image by applying the compression method directly to the binary domain (error-diffused images) without any inverse halftoning technique.

The remainder of the paper is organized as follows: a review of error diffusion algorithms is given in Section 2, whereas the proposed edge detection method for halftone images is presented

† Author for correspondence.

in Section 3. The proposed edge-preserved compression technique of error-diffused images is then described in Section 4. Experimental results and conclusions are given in Sections 5 and 6, respectively.

2. A REVIEW OF ERROR DIFFUSION (ED)

ED is a popular halftoning technique for generating high-quality halftone images. In an ED algorithm, each pixel of the input image is compared with a fixed threshold value; if the current pixel's gray value is greater than the threshold value, the output is assigned as one, otherwise, a zero is placed. The error between the input and output of the current pixel is propagated to the unprocessed neighboring pixels (the propagation ratios among these neighbours are determined by a weight matrix). Figure 1(a) depicts the block diagram and the associated parameters of a typical ED, with the weight matrix used in Fig. 1(a) being the well-known weight matrix [see Fig. 1(b)] proposed by Floyd and Steinberg. Over the years, several modifications of the ED algorithm introduced by Floyd and Steinberg [7] had been proposed. Among the reported modifications, artifacts reduction [16–22] and printability improvement [23, 24] were the most important contributions. The commonly-seen approaches for artifacts reduction are: (1) alteration of the weight matrix [16, 17], (2) changing the raster order [3, 18], (3) modification of the threshold [3, 19], and (4) optimization [20–22]. On the other hand, there are two major approaches to improving the printability of error-diffused images: the fist is called the printer-model-based approach which incorporates printer model with error diffusion algorithm [23], and the second approach is to form clusters in the output image [24]. Since there are so many versions of ED, and almost all of them are derived from Ref. [7], to avoid confusion, whenever we mention error-diffused images in this paper, we always mean that the images are halftoned by the method proposed by Floyd and Steinberg [7].

3. THE PROPOSED EDGE DETECTION METHOD

We first discuss in this section how edges are represented in the halftone images. In gray-level images, edges can be treated as a boundary between two regions having different gray values. On the other hand, it is well-known that all halftoning algorithms imitate gray values by varying dot patterns (some distribution of the black and white dots); therefore, an edge of halftone images is formed by two adjacent regions having different dot patterns. The distinction can be seen in Fig. 2. A gray-value image is shown in Fig. 2(a), and its error-diffused version is shown in Fig. 2(b).

According to our experience, most gray-level edge detection methods (Sobel, Prewitt, Laplacian, Valley, *etc.*) are not suitable for halftone images. To convince the readers, we apply the Sobel [4] and Valley [5] methods, respectively, to the halftone image shown in Fig. 2(b), and the resulting edge images (both threshold values are selected by trial-and-error) are shown in Fig. 2(c–d). We can see that the results are not good. Therefore, the first goal of this paper is to find a new edge detection method which can be directly applied to halftone images (the second goal is to use this new method to compress error-diffused images [7]). From the previous paragraph, edge detection of halftone images can be considered as a problem of judging whether a given area contains more than one types of dot pattern. A feasible approach using the mass center of the given area is suggested here.

In order to let the readers know how we got the idea of using the mass center, we explain below the related observation. Physically, if a region is composed of several mass points regularly distributed over the region, then the mass center of the region will be very close to the geometric center of the region. On the other hand, we know that a uniform region of halftone image contains only one type of dot pattern; moreover, if we let each pixel be a mass point with the mass being its black (or white) value, *i.e.* 0 (or 1), the mass center will also be close to the geometric center. Therefore, we thought maybe we could use the distance between the mass center and the geometric center of a given region to judge whether the region is uniform or contains an edge. The proposed edge detector (or consistency evaluator) is a window-based operator that computes the mass center (MC) and other related information. We first introduce the term 'consistency' to represent the possibility that a specified neighbourhood (of an arbitrary pixel $(\bar{x}, \bar{y})$) contains only one type of dot pattern. More precisely, let

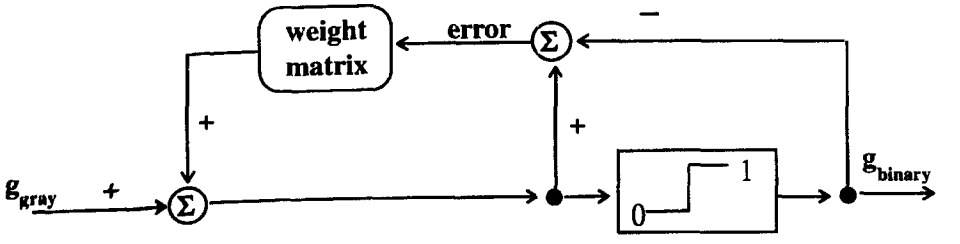

Fig. 1. (a) The typical error diffusion algorithm. (b) The weight matrix proposed by Floyd and Steinberg [7] ('x' is the location of the current pixel).

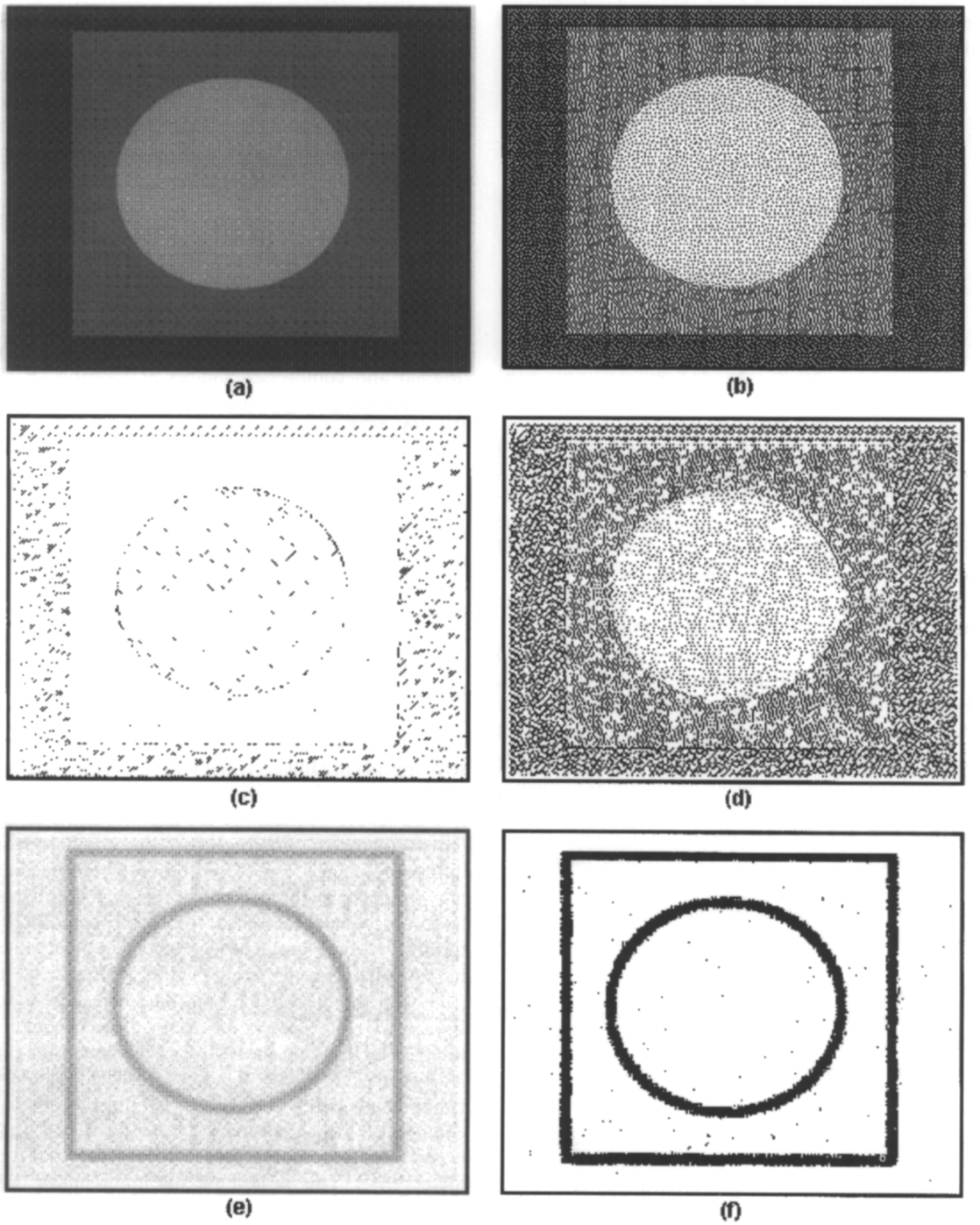

Fig. 2. An illustration of edge detection for halftone image. (a) The gray-level image; (b) the error-diffused version of (a). Note that (b), instead of (a), is used as the input image for (c)–(f). (c) The edge image obtained by applying the Sobel method [4] to (b); (d) the edge image obtained by applying the Valley method [5] to (b); (e) the consistency image after applying Equation (1) to (b); (f) the edge image generated by applying our method to (b).

$$W = \{(x, y) | x = \bar{x} \pm 0, \bar{x} \pm 1, ...; \quad y = \bar{y} \pm 0, \bar{y} \pm 1, ...\}$$

be a window containing $N \times N$ pixels. Note that the geometric center of W is $(\bar{x}, \bar{y})$. The consistency of the pixel $(\bar{x}, \bar{y})$ is then defined as

$$C(\bar{x}, \bar{y}) = 1 - Q[\|(x_{MC}, y_{MC}) - (\bar{x}, \bar{y})\|] \quad (1)$$

where

$$x_{MC} = \frac{\sum_{(x,y)\in W}(x \times f(x,y))}{M_0} \quad (2)$$

$$y_{MC} = \frac{\sum_{(x,y)\in W}(y \times f(x,y))}{M_0}, \quad (3)$$

$$M_0 = \sum_{(x,y)\in W} f(x,y) \quad (4)$$

Note that $f(x, y)$ is the bilevel value (0 denotes black pixel while 1 denotes white pixel) at (x, y), $\|\cdot\|$ is the Euclidean distance and Q is a normalization operator so that $Q[\,]$ always has a value staying in the range $0 \leqslant Q[\,] \leqslant 1$. Also note that if window W contains only one type of dot pattern, the consistency value will be close to 1. On the other hand, the consistency value will be much less than 1 when some edges exist in window W. With $(\bar{x}, \bar{y})$ ranging through all pixels of the input image, Equation (1) gives us a consistency image C. After that, we partition the 1-dimensional data set $\cup_{(\bar{x},\bar{y})\in \text{image}} C(\bar{x}, \bar{y})$ into two classes (high-consistency and low-consistency) using an automatic (parameter-free) two-class partition method, called the RWM method [25]. Finally, the edge image E is constructed using the following rule: *If $C(\bar{x}, \bar{y})$ is high, then assign $E(\bar{x}, \bar{y}) = 255$; else assign $E(\bar{x}, \bar{y}) = 0$.*

The proposed edge detection method for halftone images is summarized as follows:

Algorithm 1. *Edge detection.*

Input: a halftone image H and the window size of the consistency evaluator.

Output: the edge image E.

Steps:

(1) Use Equations (1)–(4) to obtain a consistency image $C = \cup_{(\bar{x},\bar{y})} C(\bar{x}, \bar{y})$ with $(\bar{x}, \bar{y})$ ranging through all pixels of the input image H.
(2) Partition the 1-dimensional data $\{C(\bar{x}, \bar{y}) | (\bar{x}, \bar{y})$ are pixels of $H\}$ into two classes: high-consistency vs low-consistency.
(3) Each pixel $(\bar{x}, \bar{y})$ whose $C(\bar{x}, \bar{y})$ is high (low) will be painted as a white (black, respectively) pixel. The resulting (bilevel) image is the desired edge image E.

Figure 2(e) shows the consistency image $C = \cup_{(\bar{x},\bar{y})} C(\bar{x}, \bar{y})$ after applying Equation (1) to the halftone image shown in Fig. 2(b), whereas Fig. 2(f) is the final binary edge image E obtained by bisecting the pixels of the image C into two classes: black (low consistency value) vs white (high consistency value).

4. DATA COMPRESSION OF ERROR-DIFFUSED IMAGES

This section deals with the compression of (binary) error-diffused [7] images (both the input and output are error-diffused images). In image compression, RMSE (root mean square error) is one of several commonly-seen measures used to gauge the image quality. The RMSE is defined as

$$\text{RMSE} = \left[\frac{1}{N \times M}\sum_{x=0}^{N-1}\sum_{y=0}^{M-1}(h(x,y) - \hat{h}(x,y))^2\right]^{1/2} \quad (5)$$

where $h(x, y)$ and $\hat{h}(x, y)$ are the binary values of the original and compressed images at location (x, y). Note that the binary values would be scaled from [0, 1] to [0, 255] when we use Equation (5).

As was pointed out in Section 1, some existing compression methods of halftone images are operated on gray-level domain; in other words, these methods contain three steps, namely, 1. converting the halftone images to gray-level images (by the so-called inverse halftoning technique), 2. applying an existing gray-level compression method to the intermediate gray-level image obtained by Step 1), and 3. re-halftoning. (The reason Step 3 was needed is that the printer (facsimile) is binary.) For example, the error-diffused compression method suggested by Ting and Riskin [13] is an example using this three-step approach. However, the results obtained by this kind of approach usually lose the detail information. For the reader's benefit, we use Fig. 3 to demonstrate the phenomenon. Figure 3(a) is the original error-diffused [7] image, Fig. 3(b1) shows the intermediate gray-level image (Step 1) after applying the inverse halftoning technique (see [13]) to (a). Figure 3(b2) is the intermediate gray-level image (Step 2) obtained by applying wavelet compression method [26] to (b1), with the bit rate 0.5 bpp (bit per pixel). As for Fig. 3(b3), it shows the error-diffused version (Step 3) of (b2) and its RMSE (deviation from Fig. 3(a)) is 155.17. In general, no matter which gray-level compression skill is applied to Fig. 3(b1) to obtain a compressed gray-level image $\tilde{f}$ [*e.g.* Fig. 3(b2)], the $\tilde{f}$ is too blurry to be used to produce a clear halftone image $H(\tilde{f})$. This is because the input image [Fig. 3(b1)] to the gray-level compression method (*e.g.* the wavelet method) is already blurry, and even if a lossless compression method is used to compress Fig. 3(b1), the resulting compressed image $\tilde{f}$ is still blurry. To let the readers know what will happen if the role of the wavelet compression is replaced by a lossless compression method, we directly halftoned the gray-level image shown in Fig. 3(b1) (without the application of any compression method) and got the error-diffused [7] image shown in Fig. 3(c). We can see that Fig. 3(c) is still blurry, and the RMSE

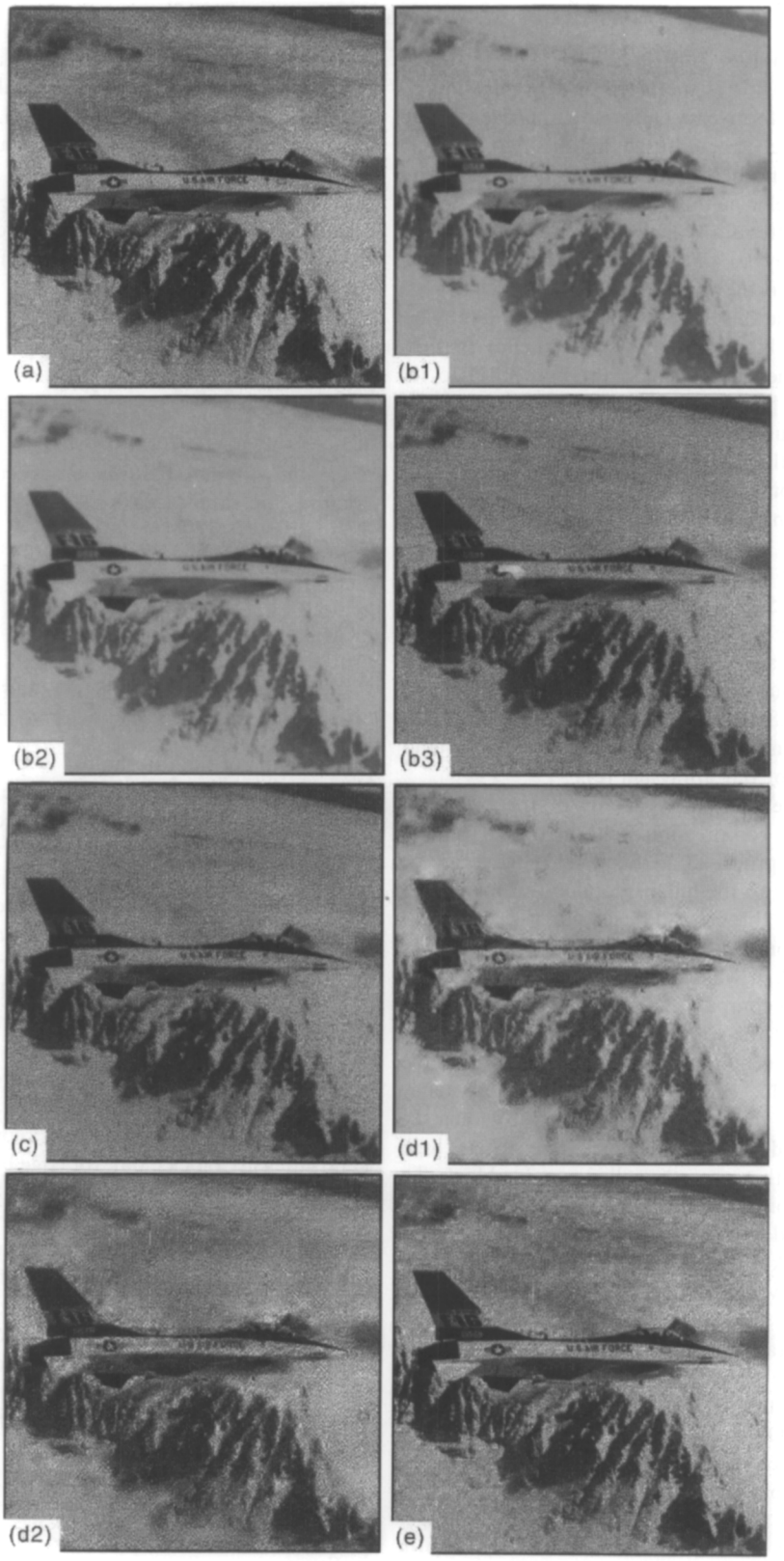

Fig. 3. An example showing that gray-level-domain compression methods have a blurring effect for compressing error-diffused images. (a) The original (input) error-diffused image; (b1) the intermediate gray-level image after applying the inverse halftoning technique [13] to (a); (b2) the intermediate gray-level image produced by applying the (0.5 bpp) wavelet compression method [26] to (b1); (b3) the re-halftoning (binary) image of (b2). (c) The re-halftoning (binary) image of (b1) without doing any compression. (d1) The intermediate gray-level image generated by applying the (0.5 bpp) wavelet compression method [26] directly to (a) (with black and white pixels of (a) interpreted as 0 and 255, respectively); (d2) the re-halftoning (binary) image of (d1). (e) The reconstructed (binary) image obtained by applying our method using the compression rate CR = 2 (bpp is 0.5) to (a) directly. The RMSEs (deviations from the binary input image (a)) are 155.17, 154.04, 154.91 and 115.06 for the binary images (b3), (c), (d2) and (e), respectively.

[deviation from Fig. 3(a)] is 154.04 [a little better than Fig. 3(b3)]. It is therefore evident that image compression with inverse halftoning approach is not suitable for the error-diffused images that contain fine details. Another interesting experiment is to apply the wavelet method [26] directly to compressing Fig. 3(a) (treats all black pixels as a gray-valued pixels with gray 0, and all white pixels with gray 255); the result is the 'gray-level' image shown in Fig. 3(d1). To enable this gray-level image (256 levels) be printed using a binary printer (*e.g.* facsimile), the ED technique is then used to convert Fig. 3(d1) to its binary version [shown in Fig. 3(d2)]. The bit rate is still 0.5 bpp, but the quality [Fig. 3(d2)] is completely unacceptable (the RMSE is 154.91). Finally, Fig. 3(e) shows the reconstructed (binary) image by applying our method (introduced below, using the compression rate CR = 2 (bpp is 0.5)) directly to the image Fig. 3(a). In Fig. 3(e), the RMSE [deviation from Fig. 3(a)] is 115.06, much smaller than that of Fig. 3(b3,c,d2). Our approach has an advantage not only in RMSE but also in computation time. In Fig. 3, given the input (a), the computation times are, respectively, 5.86 seconds for the approaches (b1)–(b3); 4.07 seconds for the approaches (d1)–(d2); and 1.54 seconds for our approach (e).

We illustrate below the details of the proposed edge-preserved compression method (based on the edge detector proposed in Section 3) that can be applied directly to the binary domain (error-diffused images). First, the error-diffused images are partitioned into non-overlapping blocks with size 8×8. For each block, a window that just covers the block is then used to compute the consistency value of the pixel located at the window center. This consistency value of the center pixel is then taken as the consistency value of the block. If the consistency value is greater than a specified threshold τ, then the block is called a uniform block due to its high consistency. Otherwise, it is called an edge block. To achieve the compression goal, each uniform block will be replaced by one of the 64 predefined constant-intensity blocks. The replacement is required to preserve the average illumination of the block, *i.e.* the average of the $8\times8=64$ intensity values of the block will be (almost) unchanged after this replacement. On the other hand, each edge block is transmitted (stored) directly without doing any replacement. Of course, one extra category bit is needed for each block to denote the class of the block. [The block size used in our method is determined by the following trade-off: (1) the larger the block size, the lower the bit rate of uniform blocks; (2) the larger the block size, the more visible the blocking effect existing in the reconstructed image; (3) to detect local detail, the block size had better be small, but too small a block size increases the ambiguity of distinguishing edge regions from uniform regions (*e.g.* 4×4 blocks have this ambiguity). By observations (1)–(3), we found empirically that the block size 8×8 is a compromise selection.]

The threshold τ can be calculated from the required compression rate. The details of computing the threshold τ are described below. Let P_1 and P_2 be the percentage of edge blocks and uniform blocks, respectively. Then $P_1+P_2=1$, and the number of bits (including the category bit) needed to represent a 8×8 block is $1+64$ or $1+6$, according to the category. In other words, $P_1(1+64)+P_2(1+6)$ bits per block, or equivalently,

$$\frac{P_1(1+64)+P_2(1+6)}{8\times8}=\frac{1}{64}+P_1+\frac{6}{64}P_2 \text{ bits per pixel (bpp)}$$

Since the original halftone image uses 1 bpp, the compression ratio (CR) is

$$\mathrm{CR}=\frac{1}{\frac{1}{64}+P_1+\frac{6}{64}P_2} \tag{6}$$

After some calculation, we obtain

$$P_1=\frac{64\times\left(\frac{1}{\mathrm{CR}}-\frac{7}{64}\right)}{58}=\frac{\left(\frac{64}{\mathrm{CR}}-7\right)}{58} \tag{7}$$

The value of the threshold τ is selected so that the percentage of the blocks which are edge blocks is below P_1. In other words, the value of threshold τ is selected as the P_1-tile of the histogram of all block consistency values. Note that this threshold τ guarantees the compressed images meet the compression rate requirement, in other words, this facility offers a selection for the compromise between compression rate and reconstructed image quality. Also note that, since $P_1\geq0$, Equation (7) implies $(64/\mathrm{CR})\geq7$. Therefore, $\mathrm{CR}\leq\frac{64}{7}\approx9$ is a natural limit.

On the receiving side, images are decompressed block by block. Each uniform block is reconstructed using one of the 64 predefined representative constant-intensity blocks; but, for edge blocks, each pixel is assigned 0 or 255 according to the $8\times8=64$ related binary values (0 or 1) received. After the decompression, images are halftoned again by the error diffusion algorithm [7]. This post-processing of re-halftoning is necessary to get a binary image of good quality.

Two algorithms are given below to summarize the proposed image compression method.

Algorithm 2. *Encoding part of the compression method.*

Input: an error-diffused input image and the required compression rate.

Output: code for each block.

Steps:

(1) Partition the input image into non-overlapping blocks with size 8×8 pixels.

(2) For each block, use a window that just covers the block to compute the consistency value of the pixel located at the window center. The obtained consistency value is then defined as the consistency value of the block.

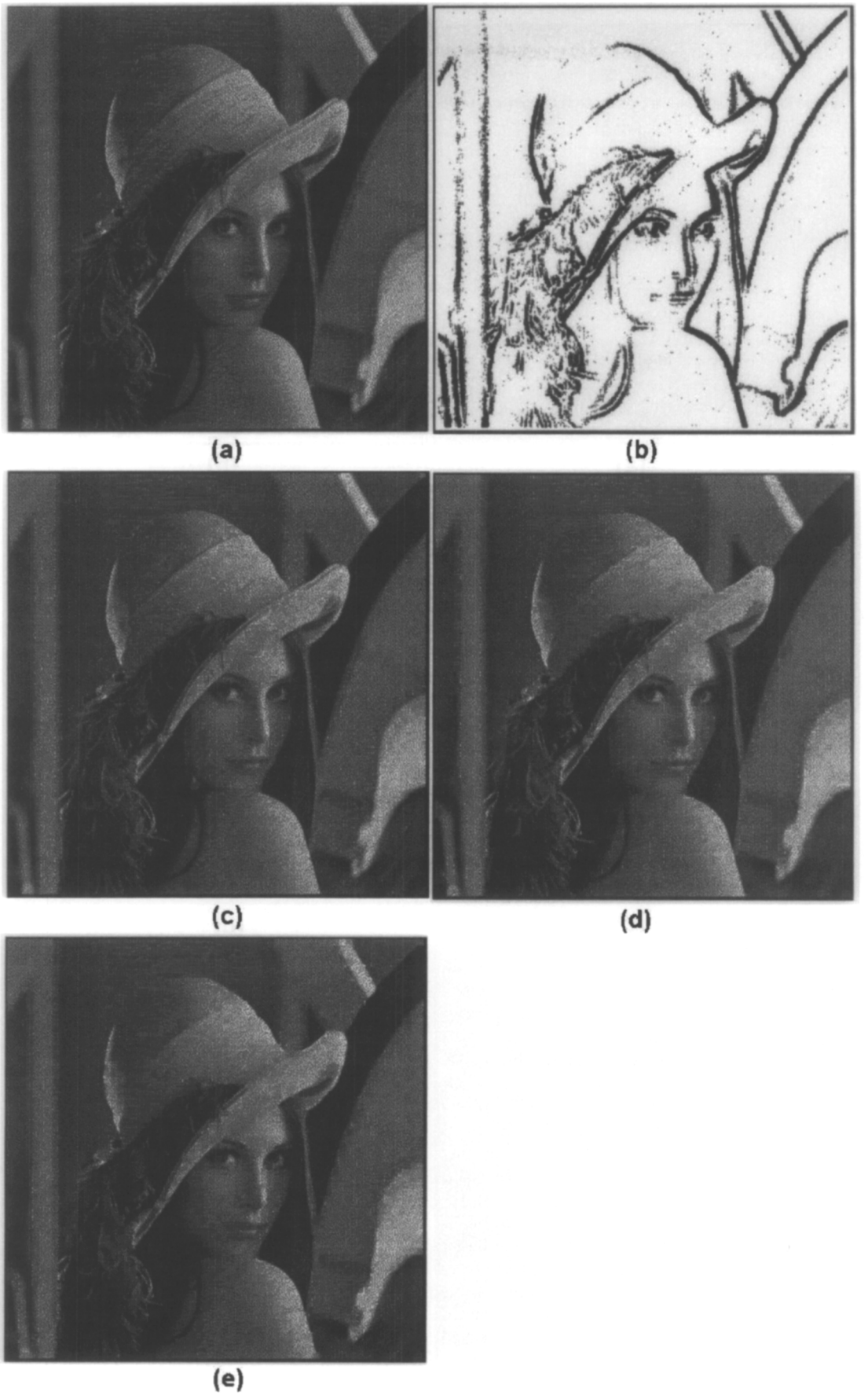

Fig. 4. The resulting images generated by using the proposed edge detection and compression methods to the image 'Lena'. (a) The original (input) error-diffused image, (b) the detected edge image for (a). The reconstructed images are: (c) CR = 2, (d) CR = 3, (e) CR = 4. Note that (c)–(e) are all error-diffused (binary) images instead of gray-level (256-level) images.

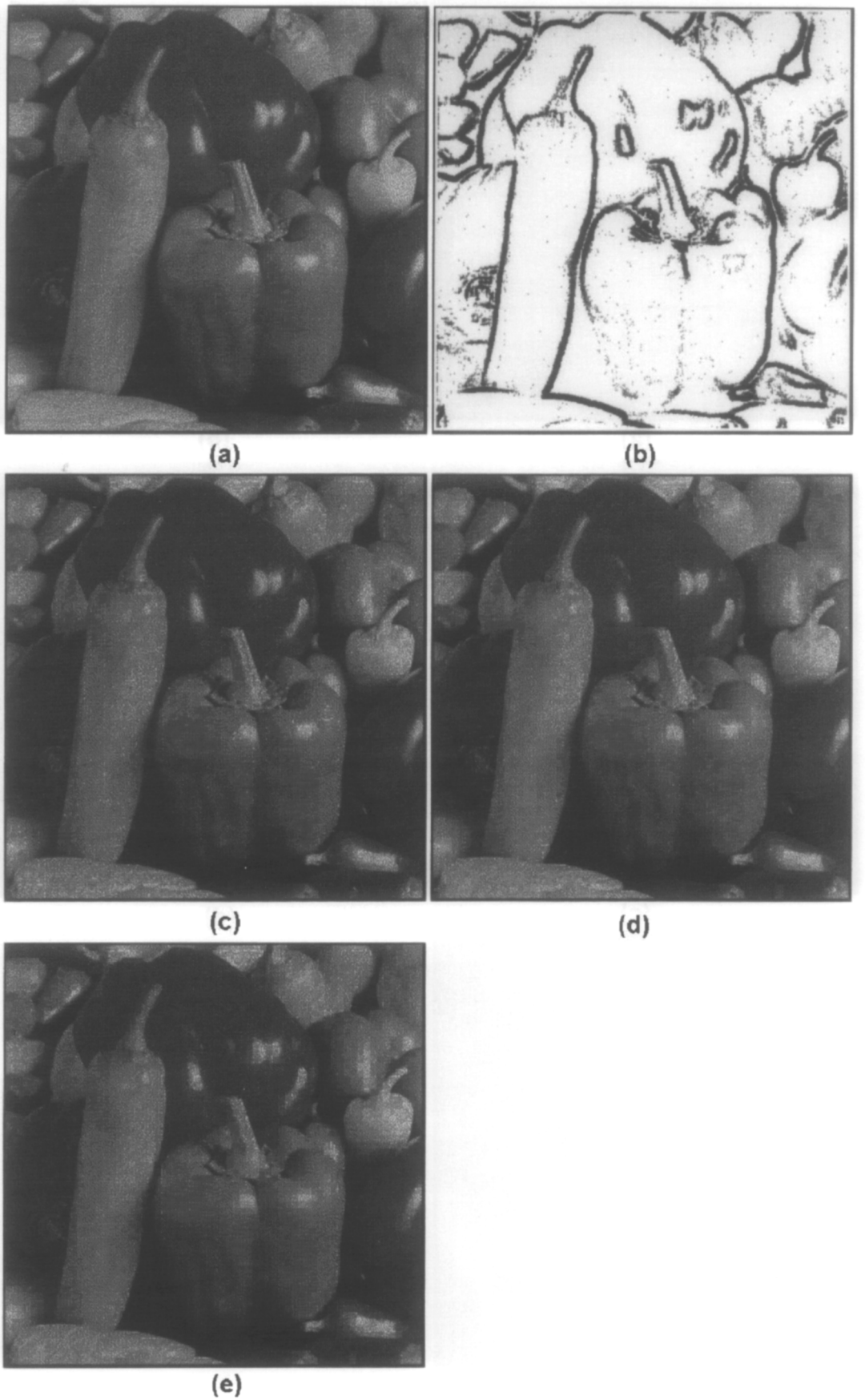

Fig. 5. The resulting images generated by using the proposed edge detection and compression methods to the image 'Pepper'. (a) The original (input) error-diffused image, (b) the detected edge image for (a). The reconstructed images are: (c) CR = 2, (d) CR = 3, (e) CR = 4. Note that (c)–(e) are all error-diffused (binary) images instead of gray-level (256-level) images.

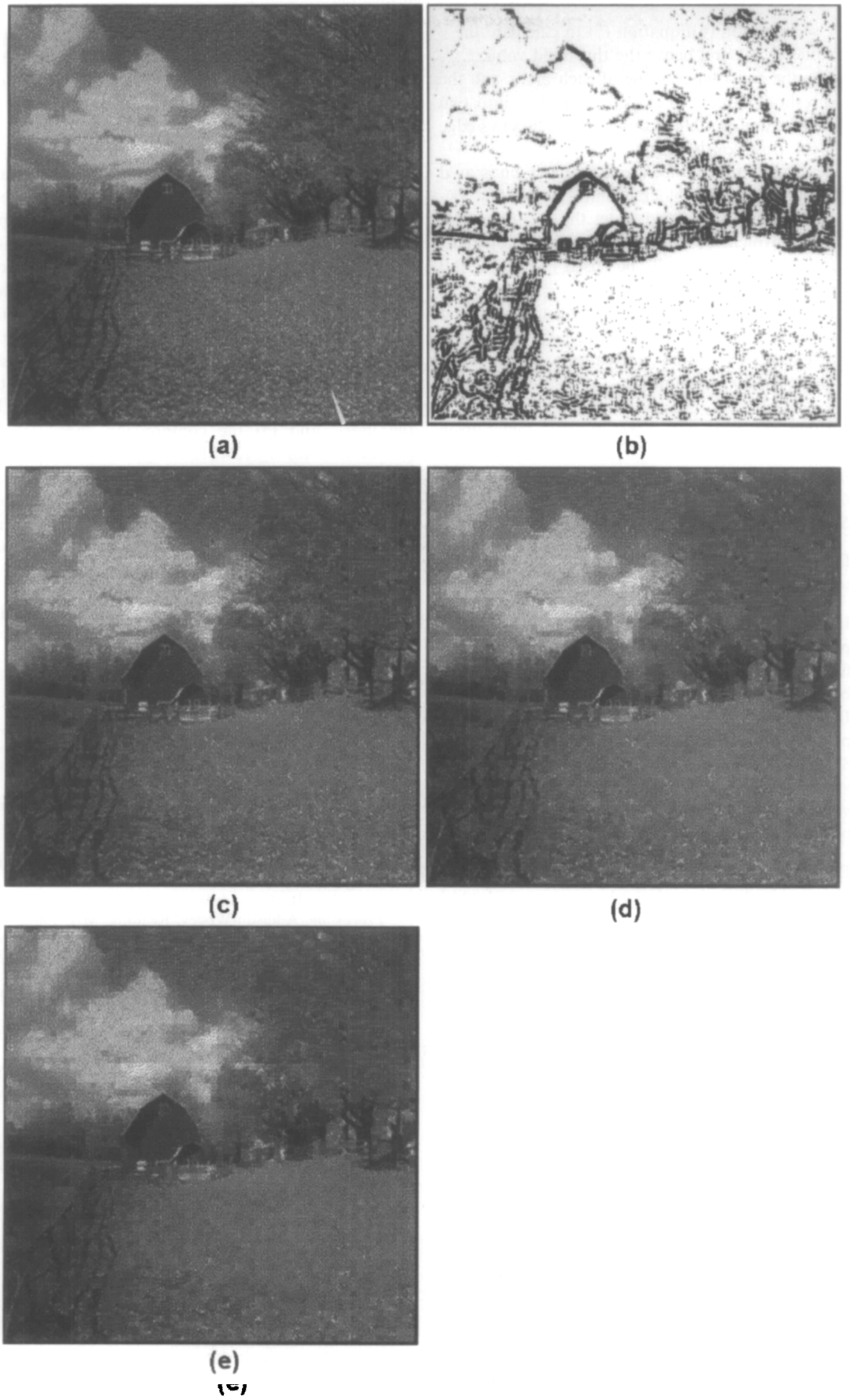

Fig. 6. The resulting images generated by using the proposed edge detection and compression methods to the image 'Farm'. (a) The original (input) error-diffused image, (b) the detected edge image for (a). The reconstructed images are: (c) CR = 2, (d) CR = 3, (e) CR = 4. Note that (c)–(e) are all error-diffused (binary) images instead of gray-level (256-level) images.

(3) Use Equation (6)Equation (7) to calculate the value of P_1, and hence the threshold value τ.
(4) Use the consistency value of each block and the threshold value τ to classify all blocks into two classes: uniform block (high-consistency) vs edge block (low-consistency).
(5) For each uniform block, one category-bit is used to denote that the block is uniform. Besides, a 6-bit code is also needed to let the decoding port know which of the 2^6 predefined constant-intensity blocks will be used to (roughly) reproduce this block.
(6) For each edge block, after sending a category-bit to denote that the block is an edge block, the original $8\times8=64$-bit dot-pattern of the edge block is also transmitted.

Algorithm 3. *Decoding part of the compression method.*

Input: the code of each block.
Output: reconstructed error-diffused image.
Steps:

- Decompress the image block by block according to the following rules:
- If the category-bit indicates that the block being processed is uniform, then use the next 6 bits to find one of the 64 predefined constant-intensity blocks.
- If the category-bit indicates that the block being processed is an edge block, then use the next $8\times8=64$ bits (0 means a black pixel whose gray value is 0, and 1 means a white pixel whose gray value is 255) to reconstruct the 64 pixels of the block.
- Error-diffuse the partially-reconstructed image obtained by Step 1 so that a binary image of good quality can be obtained.

Note that the proposed compression technique reduces the coding complexity significantly by means of operating directly on the binary error-diffused images [7]. Without any special hardware support, it can still be performed very fast for real-time applications. (The proposed compression method had been implemented on a 80486 PC using C language. It was found that the compression of a 512×512 error-diffused image [7] took only 1.5 s.)

5. MORE EXPERIMENTAL RESULTS

For the reader's benefit, more images have been tested. Figure 4(a) shows the original error-diffused image [7]. Figure 4(b) depicts the resulting edge image after applying the proposed edge detection method to the image shown in Fig. 4(a). Figure 4(c–e) were the reconstructed error-diffused images after applying Algorithm 2 and Algorithm 3 (the compression rate CR was 2, 3 and 4, respectively) to the image depicted in Fig. 4(a). Other tested images and their results are given in Figs 5and 6. It was found that the proposed compression technique preserved most of the detail information, especially when the reconstructed images were compressed under a low compression rate such as CR = 2.

6. CONCLUSIONS

A new approach for edge detection and image compression of error-diffused images [7] has been developed. The major contributions of this paper include: (1) both the edge detection and compression methods are directly operated on (binary) error-diffused images; (2) the proposed edge detection method can work with any halftone images; (3) the reconstructed images preserve the details of the original error-diffused images; (4) the proposed compression technique is equipped with an easy-to-use tool to achieve the compression rate required by the users; and (5) the compression speed is fast (about 1.5 s for a 512×512 image when a 80486 PC was used). The method is therefore suitable for real-time applications. As future work, the compression rate and image quality can both be improved further by means of using variable block sizes and vector quantization in the compression scheme.

Acknowledgements—This work was supported by the National Science Council, Republic of China, under grant NSC86-2213-E009-108.

REFERENCES

1. Jarvis, J. F., Judice, C. N. and Ninke, W. H., A survey of techniques for the display of continuous tone pictures on bilevel displays. *Computer Vision, Graphics, and Image Processing*, 1976, **5**, 13–40.
2. Stoffel, J. C. and Moreland, J. F., A survey of electronic techniques for pictorial reproduction. *IEEE Transactions on Communications*, 1981, **C-29**, 1898–1925.
3. Ulichney, R. A., *Digital Halftoning*. MIT Press, Cambridge, MA, 1987.
4. Gonzalez, R. C. and Woods, R. E., *Digital Image Processing*. Addison-Wesley, Reading, MA, 1992.
5. Pearson, D. E. and Robinson, J. A., Visual communication at very low data rates. *Proceedings of the IEEE*, 1985, **73**, 795–812.
6. Haddon, J. F., Generalized threshold selection for edge detection. *Pattern Recogition*, 1988, **21**, 195–203.
7. Floyd, R. W. and Steinberg, L., An adaptive algorithm for spatial gray scale. *Proc. SID*, 1976, **17**, 75–77.
8. Nakashima, K., Shinoda, S., Kojima, Y., Hori, Y., Nakamura, T. and Suemori, N., High-quality image processing architecture for facsimiles. *Journal of Electronic Imaging*, 1992, **1**(1), 61–67.
9. Urban, S. J., Review of standards for electronic imaging for facsimile systems. *Journal of Electronic Imaging*, 1992, **1**(1), 5–21.
10. Chao, Y., An investigation into the coding of halftone pictures, Ph.D. thesis, MIT, Cambridge, MA, 1982.
11. Hsuen, Y. C., Chern, M. G. and Chu, C. H., Image requantization by cardinality distribution. *Computers & Graphics*, 1991, **15**, 397–405.
12. Neuhoff, D. L. and Pappas, T. N., Perceptual coding of images for halftone display. *IEEE Transactions on Image Processing*, 1994, **3**, 1–13.
13. Ting, M. Y. and Riskin, E. A., Error-diffused image compression using a binary-to-gray-scale decoder and predictive pruned tree-structured vector quantization. *IEEE Transactions on Image Processing*, 1994, **3**, 854–858.

14. Forchhammer, S. and Jensen, K. S., Data compression of scanned halftone images. *IEEE Transactions on Communications*, 1994, **42,** 1881–1893.
15. Vander Kam, R. A. and Gray, R. M., Lossy compression of clustered-dot halftones. In *Proceedings ICIP-94*, Vol. 3. 1994, pp. 836–840.
16. Stucki, P., MECCA—a multiple-error correcting computation algorithm for bilevel image hardcopy reproduction. Research Report RZ1060, IBM Research Lab., Zurich, Switzerland, 1981.
17. Eschbach, R., Reduction of artifacts in error diffusion by means of input-dependent weights. *Journal of Electronic Imaging*, 1993, **2**(4), 352–358.
18. Witten, I. H. and Neal, M., Using Peano curves for bilevel display of continuous-tone images. *IEEE CG and E*, 1982, 47–52.
19. Sullivan, J., Miller, R. and Pios, G., Image halftoning using a visual model in error diffusion. *Journal of the Optical Society of America A*, 1993, **10,** 1714–1724.
20. Kolpatzik, B. W. and Bouman, C. A., Optimized error diffusion for image display. *Journal of Electronic Imaging*, 1992, **1,** 277–292.
21. Xie, Z. and Rodriguez, M., A bandwidth preservation approach to stochastic screening. In *Proc., IS&T's 3th Technical Symposium on Prepress, Proofing, and Printing*. 1993, pp. 113–116.
22. Wong, P. W., Adaptive error diffusion and its application in multiresolution rendering. *IEEE Transactions on Image Processing*, 1996, **5,** 1184–1196.
23. Pappas, T. N., Dong, C. K. and Neuhoff, D. L., Measurement of printer parameters for model-based halftoning. *Journal of Electronic Imaging*, 1993, **2**(3), 193–204.
24. Levien, R., Output dependent feedback in error diffusion halftoning. In *Proc., IS&T's 46th Annual Conf.* 1993, pp. 115–118.
25. Yang, C. Y. and Lin, J. C., Use of radius weighted mean to cluster two-class data. *Electronics Letters*, 1994, **30,** 757–759.
26. Said, A. and Pearlman, W. A., A new, fast, and efficient image code based on set partitioning in hierarchical trees. *IEEE Transactions on Circuits and Systems for Video Technology*, 1996, **6**(3), 243–250.