Image Rendering for Digital Fax

Guotong Feng[a], Michael G. Fuchs[b] and Charles A. Bouman[a]

[a] Purdue University, West Lafayette, IN
[b] Hewlett-Packard Company, Boise, ID

ABSTRACT

Conventional halftoning methods such as error diffusion and ordered dithering are poorly suited to the compression of halftone images using the baseline fax compression schemes CCITT G3 and G4. This paper proposes an efficient and flexible solution for binary representation of mixed content documents using CCITT G3/G4 compression. The solution includes two variations which we refer to as FastFax and ReadableFax. FastFax performs edge detection and text detection by applying locally adaptive binary thresholding and combines the two detection results together. The FastFax algorithm produces an accurate representation of binary mixed document content with high compressibility using CCITT G3/G4 compression. ReadableFax is based on FastFax and applies clustered dot screening to background and halftone regions to enhance graphic content. Both methods provide accurate representation of image content while allowing for substantial compressibility, and provides a tradeoff between representation quality and bitrate.

Keywords: Halftoning, compression, facsimile, fax, segmentation, mixed raster content, documents

1. INTRODUCTION

Facsimile has become a very popular means of image data transmission since the 1980s. While color and grayscale facsimile are growing, a large number of black and white facsimile systems are still being widely used. In practice, grayscale and color documents are often converted to black and white images for fast transmission. In this case, efficient bi-level rendering techniques are required for the representation of grayscale or color documents containing a mixture of text, graphics and image content.

Conventional halftoning methods such as ordered dither[1] and error diffusion[2] can be used to create a binary representation of a raster document with graphic and image content. Such a raster document can be subsequently compressed using CCITT G3/G4 compression[3, 4] which are lossless bilevel coding schemes for fax transmission defined by the International Telegraph and Telephone Consultative Committee (CCITT). However, G3 and G4 are very poorly suited for the compression of halftone images because they were originally designed for black and white text and line art.[5] Therefore, G3 and G4 compression of halftoned documents results in large file sizes and subsequently very long fax transmission times. Conventional binary thresholding with a constant threshold is also a common technique for binary representation of black and white text. But for mixed content images, thresholding produces very poor visual quality in low contrast areas, for example, where color text and color background are mixed.

In this paper, we propose a method for converting a mixed content document to a binary format which preserves the content of the document but also allows for effective compression using the G3/G4 standards that are used in fax transmission. In particular, we present two variations of our method which we refer to as FastFax and ReadableFax. Both methods convert documents to binary forms which convey the essential content of the document, but can be effectively compressed for fax transmission. This is done by representing the document's text and graphic edges, but not quantitatively reproducing the document's continuous tones.

The FastFax method performs a binary segmentation that extracts text, edges, and graphic outlines. It first preprocesses the original grayscale image to remove noise in gray regions of the document. The result is then processed to obtain two binary masks corresponding to edges and text. The edge and text masks are combined together to form a final binary representation of the complete document. Importantly, this "cartoon"-like representation substantially reduces the entropy of the document so that CCITT G3 and G4 compression algorithms can achieve high compressibility. The FastFax algorithm is computationally efficient and produces a very small compressed file size. This makes it particularly well suited for the transmission of mixed content

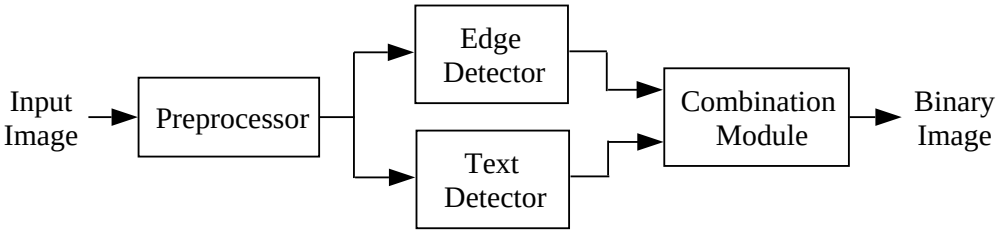


Figure 1. Block diagram of FastFax algorithm.

documents for quick review. ReadableFax is a graphic content enhanced algorithm based upon FastFax. The algorithm improves the representation of gray levels by replacing these regions with textures that can be effectively compressed using G3/G4 compression. Specifically, we use clustered dot screening to create these textures, but other halftoning methods could also be used. Thus the ReadableFax algorithm is also suitable for CCITT G3 and G4 (fax) compression, but with a somewhat lower compression ratio than is possible with FastFax.

Both FastFax and ReadableFax approaches employ a simple bi-level segmentation technique that can accurately extract text and edges from background in a mixed content document. Much work has been done in the area of segmenting text/graphics composite documents. Many segmentation methods have been proposed, such as variance based approaches,[6] edge based segmentation,[7] pattern matching combined with peak information,[8] and neural network based approaches.[9] A summary of proposed algorithms for the segmentation of text, screened halftone, and continuous tone content can be found in.[10] Our segmentation method differs from the others in its unique perspective of essential content representation.

This paper is organized as follows. Section 2 and Section 3 present the FastFax algorithm and ReadableFax algorithm, respectively. In Section 4, examples and results of the two algorithms are shown and compared with conventional techniques. Finally, Section 5 summarizes the conclusions of the paper.

2. FASTFAX ALGORITHM

Fig. 1 shows a block diagram of the proposed FastFax algorithm. The algorithm includes a preprocessor, an edge detector, a text detector, and a combination module.

Fig. 2 shows a block diagram of the preprocessor. The objective of the preprocessor is to eliminate undesired noise in gray regions of the document. Scanned documents typically contain high frequency noise from the halftone used to render the original hardcopy. The process of removing this halftone noise is known as "descreening". The preprocessor performs simple descreening by filtering the portions of the document that fall into the mid gray ranges that typically contain the majority of high frequency noise.

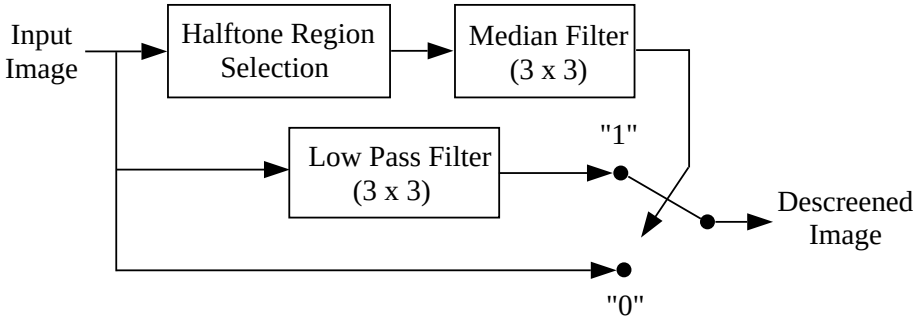


Figure 2. Block diagram of preprocessor.

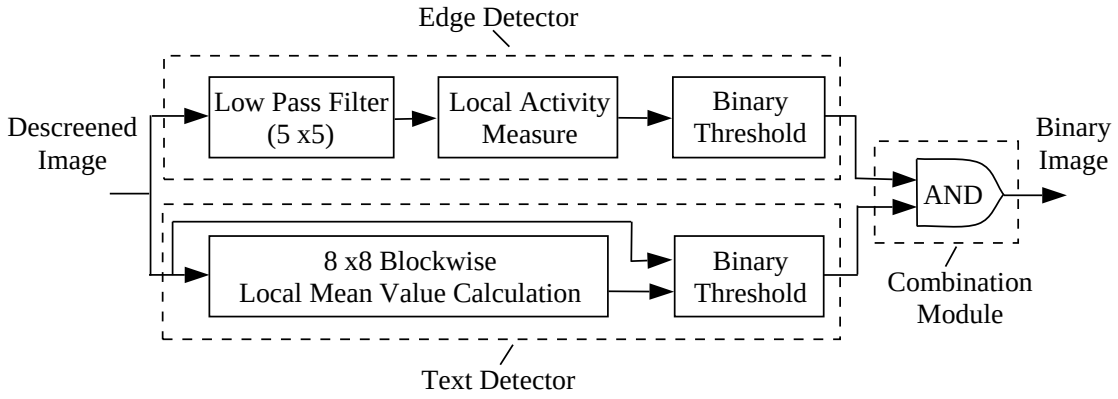


Figure 3. Block diagram of edge detector, text detector and combination module.

Denote the input raster image as $0 \leq g(m,n) \leq 1$ with 0 corresponding to white and 1 corresponding to black. The "Halftone Region Selection" determines regions of the image that are likely to contain substantial halftone noise. To do this, we compute

$$d(m,n) = u\left(T_h - |g(m,n) - 0.5|\right) \ , \tag{1}$$

where T_h is a preselected threshold value and the function $u(x)$ is equal to 1 if $x \geq 0$, and is equal to 0 if $x < 0$. So if the shading at pixel (m,n) is close to white ("0") or black ("1"), then $d(m,n)$ is equal to 0 (non-halftone). Otherwise, $d(m,n)$ is equal to 1 (halftone).

Next, a 3×3 median filter is applied to the binary mask d to form the output $\tilde{d}$. The operation can be written as

$$\tilde{d}(m,n) = u(\sum_{i=-1}^{1}\sum_{j=-1}^{1} d(m-i,n-j) - 5)\,, \tag{2}$$

This operation is important because black text on white background will still generate scanned gray values along text edges. These gray value pixels are initially detected as halftone regions, but the median filter removes these thin lines of detected pixels.

The value $\tilde{d}(m,n)$ is used to control a binary switch as shown in Fig. 2. If the pixel is from a non-halftone region, then it is processed without filtering. But, if the pixel is from a halftone region, then it is processed using a 3×3 low pass filter. More specifically,

$$f(m,n) = \tilde{d}(m,n)\left(\sum_{i=-1}^{1}\sum_{j=-1}^{1} h_1(i,j)g(m-i,n-j)\right) + (1-\tilde{d}(m,n))g(m,n)\,, \tag{3}$$

where $f(m,n)$ is the output and $h_1(i,j)$ is a simple low pass filter of 3×3 size. The binary switching is beneficial because it allows fine edge detail to be preserved while still removing undesirable halftone textures from the input image.

2.1. Edge detector

A block diagram of the edge detector is shown in Fig. 3 together with the text detector and combination module. The descreened document image is first smoothed by using a 5×5 Gaussian low pass filter with the standard deviation σ to remove noise.

$$\tilde{f}(m,n) = \sum_{i=-2}^{2}\sum_{j=-2}^{2} h_2(i,j)f(m-i,n-j) \ , \tag{4}$$

A local activity measure is then computed for the filtered output using the following formula.

$$e(m,n) = \sqrt{\frac{1}{8}\sum_{i=-1}^{1}\sum_{j=-1}^{1}\left|\tilde{f}(m,n) - \tilde{f}(m-i,n-j)\right|^2}, \tag{5}$$

A binary threshold block compares the value of $e(m,n)$ to a threshold, T_e, at each pixel and creates the binary edge detection output b_1.

$$b_1(m,n) = u(e(m,n) - T_e) \tag{6}$$

2.2. Text detector

The edge detector will detect the location of text edges, but it tends to distort and enlarge the shape of fine text structures. For this reason, a second text detector is required. The first step to text detection is the calculation of a local threshold shown in Fig. 3. A local threshold is computed for each 8×8 block in the document image. The threshold for the (k,l) block is denoted by $T(k,l)$. This threshold is computed using the following adaptation of the well known K-means clustering algorithm.

1. Compute the mean value (μ) of pixels in the block.
2. For $k = 0$ to 2
   (a) Group the pixels in the block into two clusters where C_{low} contains all pixels with value less or equal to μ, and C_{high} contains all pixels with value greater than μ.
   (b) If either cluster contains no pixels, then take μ as the local threshold value, $T(k,l)$, and terminate.
   (c) Compute the mean for each cluster. Let μ_{low} be the smaller mean value and μ_{high} the larger value.
   (d) Set $\mu \leftarrow (\mu_{low} + \mu_{high})/2$.
3. Compute the local threshold value $T(k,l)$ as:

$$T(k,l) = \alpha\mu_{low} + (1-\alpha)\mu_{high}\,, \tag{7}$$

   where α is a predefined value between 0 and 1.

Next, the pixels in each block are thresholded using the computed local threshold value for the block to form the text detected binary output b_2.

$$b_2(m,n) = u(f(m,n) - T(\lfloor m/8 \rfloor, \lfloor n/8 \rfloor))\,, \tag{8}$$

2.3. Combination module

As shown in Fig. 3, a logical AND block combines the output from the edge detector and the text detector at every pixel to produce the binary output image of FastFax. Thus the output $b(m,n)$ at pixel (m,n) can be written as

$$b(m,n) = b_1(m,n) \cap b_2(m,n)\,, \tag{9}$$

where $b_1(m,n)$ and $b_2(m,n)$ are the output of the edge detector and text detector at pixel (m,n), respectively. The output image contains sharp text and graphic edges because the text and graphic outlines are accurately detected and separated from the background.

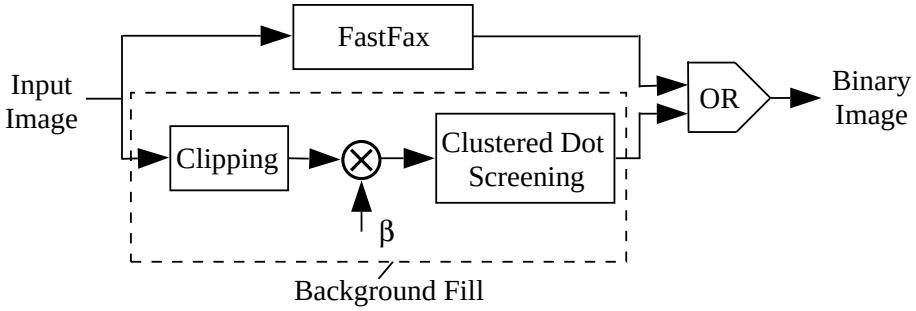


Figure 4. Block diagram of ReadableFax algorithm.

10	12	11	9	7	8
13	18	17	6	1	2
14	15	16	5	4	3
9	7	8	10	12	11
6	1	2	13	18	17
5	4	3	14	15	16

Figure 5. Threshold matrix of the clustered dot screen used in ReadableFax.

3. READABLEFAX ALGORITHM

The purpose of ReadableFax is to enhance the graphic content of FastFax by representing different gray levels with different textures while maintaining high compressibility using CCITT G3/G4. Fig. 4 shows how the FastFax result is combined with a clustered dot screened version of the document to produce an improved quality rendering with larger compressed file size.

The algorithm includes a FastFax module, a background fill module and a logical OR function. In the background fill module, clustered dot screening is applied to the entire input image, with the gray level of the input image properly clipped and scaled. The screening result is then combined with the FastFax result using the logical OR function. In essence, the clustered dot screening is turned on only in the background regions ("0") determined by FastFax, and turned off in the text/edge regions ("1") determined by FastFax. Thus the resulting binary output image is formed by a combination of the clustered dot screen result in background fill regions, and the FastFAx result in text and graphic outline regions.

First, the input image is clipped and scaled to produce the output $\tilde{g}(m,n)$.

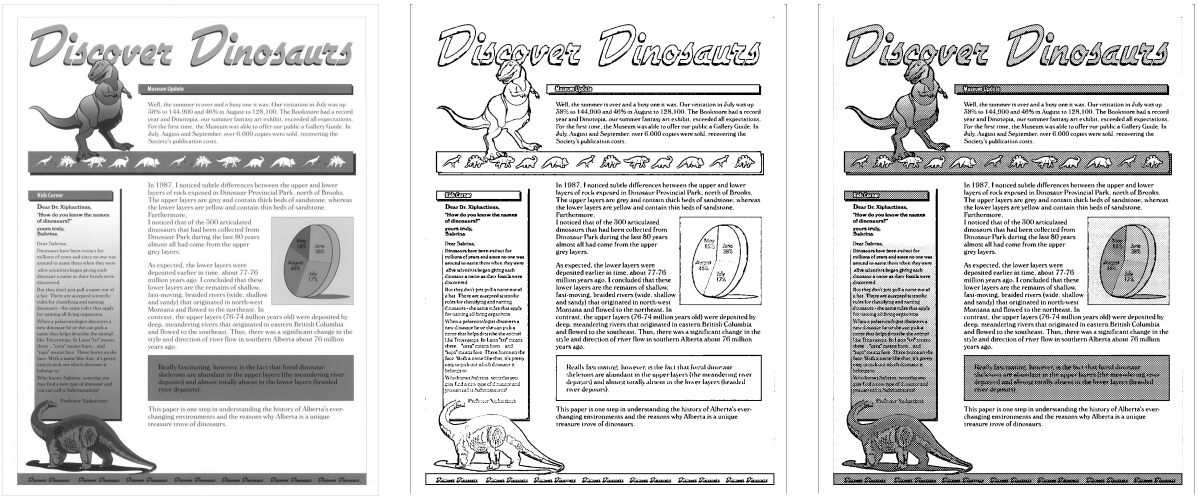
$$\tilde{g}(m,n) = \begin{cases} 0 & 0 \le g(m,n) < T_c \\ \beta \frac{g(m,n)-T_c}{1-T_c} & T_c \le g(m,n) \le 1 \end{cases}, \tag{10}$$

This clipping and scaling operation reduces noise and increases the contrast between the text/graphic content of the FastFax ouput, and the background fill of the clustered dot screen.

After clipping and scaling, a 3×6 $45°$ clustered dot screen is used to halftone the image $\tilde{g}(m,n)$. The threshold matrix of the screen is shown in Fig. 5. The purpose of using clustered dot screening is to maintain high compressibility with decent graphic content quality. An important advantage of clustered dot screening over error diffusion is that it is more effectively compressed by G3 and G4 compression schemes.

4. EXPERIMENTAL RESULTS

We used a set of 7 test images to evaluate the performance of the FastFax and ReadableFax algorithms, and to compare them against the alternative methods of simple thresholding, error diffusion, and clustered dot screening. All test images were scanned from hard copy at 300 dpi to form raster images of size 2448×3200. The raster



(a) (b) (c)

Figure 6. A full page example of FastFax and ReadableFax results: (a) original color image, scanned at 300 dpi with size of 2448 × 3200, (b) FastFax, G4 compressed file size 82 KBytes, and (c) ReadableFax, G4 compressed file size 322 KBytes.

images were then stored in JPEG format using a high quality factor to minimize distortion. All algorithms use a gray scale input formed by taking the Y component of a color image converted to the YC_rC_b color space.

The FastFax and ReadableFax algorithms used the parameters listed in Table 1. We choose these parameters because we found that they worked well for a wide variety of documents. Floyd Steinberg error diffusion was used with a serpentine scan pattern, and the binary thresholding algorithm used a fixed threshold of 127 for images scaled to a range of 0 to 255. The clusted dot screening algorithm used a 3×6 45° screen as shown in Fig. 5. The same screen was used in the ReadableFax results. All binary outputs were saved in TIFF format using CCITT G4 compression.

Fig. 6 shows a full page example of the FastFax and ReadableFax results. The Fastfax file size is 82 KBytes and the ReadableFax file size is 322 KBytes, resulting in bit rates of 0.083 bpp and 0.32 bpp respectively. The G4 compressed file size of FastFax is generally much smaller than that of ReadableFax, but the additional bit rate of ReadableFax allows for enhanced document quality in the rendition of background fills. In practice, these two methods can be chosen by users as alternative options, depending on their preferences for short transmission time or image content fidelity.

Fig. 7 shows a comparision of a small portion of the binary image resulting from the five methods, and Table 2 lists the compressed file sizes resulting from the application of each of the methods to each of the seven test images. The constant binary thresholding method produces a poor quality binary representation with major content loss in some halftone regions, especially those containing color text, and it usually results in a larger compressed file size than does the FastFax algorithm. Floyd Steinberg error diffusion produces very good overall image quality, but the compressed file size is by far the largest of the methods. Alternatively, clustered dot screening provides poor rendering of text edges and details, but generates a much smaller compressed file size than Floyd Steinberg error diffusion.

Table 1. Summary of parameters used in FastFax and ReadableFax.

Parameter	T_h	σ	T_e	α	T_c	β
Value	0.314	1.5	0.0196	0.4	0.0588	0.75

In most cases, the FastFax method produced the smallest file size of all the methods. In addition, the FastFax method did a good job of preserving the document's text and graphic content. Notice that the text is quite readable and the edge detail gives the user a clear understanding of the graphic content in the document. The ReadableFax result preserves the accurate text edges and graphic content of the FastFax method, but also enhances the rendering of fill areas. However, it also generates compressed file sizes that are are about 4 times larger than the FastFax method. From another perspective, the ReadableFax file size is generally somewhat smaller than the clustered dot rendering, and has much higher text and graphics quality.

Table 2. Compressed image sizes (KBytes) using CCITT G4 with different halftoning methods.

Test Image	Binary Thresholding	Floyd Steinberg Error Diffusion	Clustered Dot Screening	FastFax	ReadableFax
"dinosaur"	187	923	369	82	322
"toystore"	129	917	349	82	297
"NorthPage1"	104	1,193	411	89	368
"NorthPage2"	148	1,524	579	150	468
"NorthPage3"	95	557	321	62	112
"NorthPage4"	76	701	261	84	208
"lotto"	11	451	254	59	69

5. CONCLUSION

We have proposed two methods for generating binary representations of mixed content documents that can be efficiently compressed using the CCITT G3/G4 fax compression standards. Both methods work by extracting text, edge, and graphic detail from complex documents. The FastFax method produces a binary representation of the document content which can be effectively compressed for fax transmission, and the ReadableFax method enhances FastFax by using binary patterns to represent fill regions of the document. Both methods have a simple and efficient implementation, and experimental results indicate their effectiveness on a variety of scanned documents.

ACKNOWLEDGMENTS

This work was supported by Hewlett-Packard.

REFERENCES

1. R. Ulichney, *Digital Halftoning*, MIT Press, Cambridge, MA, 1987.
2. R. W. Floyd and L. Steinberg, "An adaptive algorithm for spatial greyscale," *Journal of the Society for Information Display* **17**(2), pp. 75–77, 1976.
3. CCITT Recommendation T.4, "Standardization of Group 3 facsimile apparatus for document transmission," **VII-Fascicle VII.3**, pp. 21–47.
4. CCITT Recommendation T.6, "Facsimile coding schemes and coding control functions for Group 4 facsimile apparatus," **VII-Fascicle VII.3**, pp. 48–57.
5. S. J. Urban, "Review of standards for electronic imaging for facsimile systems," *Journal of Electronic Imaging* **1**, pp. 5–21, Jan. 1992.
6. C. T. Chen, "Transform coding of digital images using variable block size DCT with adaptive thresholding and quantization," in *Proc. of SPIE - The International Society for Optical Engineering*, **1349**, pp. 43–54, 1990.
7. P. J. Bones, T. C. Griffin, and C. M. Carey-Smith, "Segmentation of document images," in *Proc. of SPIE - The International Society for Optical Engineering*, **1258**, pp. 66–78, 1990.

8. S. Ohuchi, K. Imao, and W. Yamada, "A segmentation method for composite text/graphics (haftone and continuous tone photographs) documents," *Systems and Computers in Japan* **24**(2), pp. 35–44, 1993.
9. A. Ikeda and Y. Shimodaira, "Segmentation of character and natural image documents with neural network model for facsimile equipments," in *Proc. of IEEE Instrumentation and Measurement Technology Conference*, **1**, pp. 149–152, (Hamamatsu, Japan), May 10-12 1994.
10. M. Yoshida, T. Takahashi, T. Semasa, and F. Ono, "Bi-level rendition of images containing text, screened halftone and continuous tone," in *Global Telecommunications Conference*, **1**, pp. 104–109, (Phoenix, AZ), Dec. 2-5 1991.

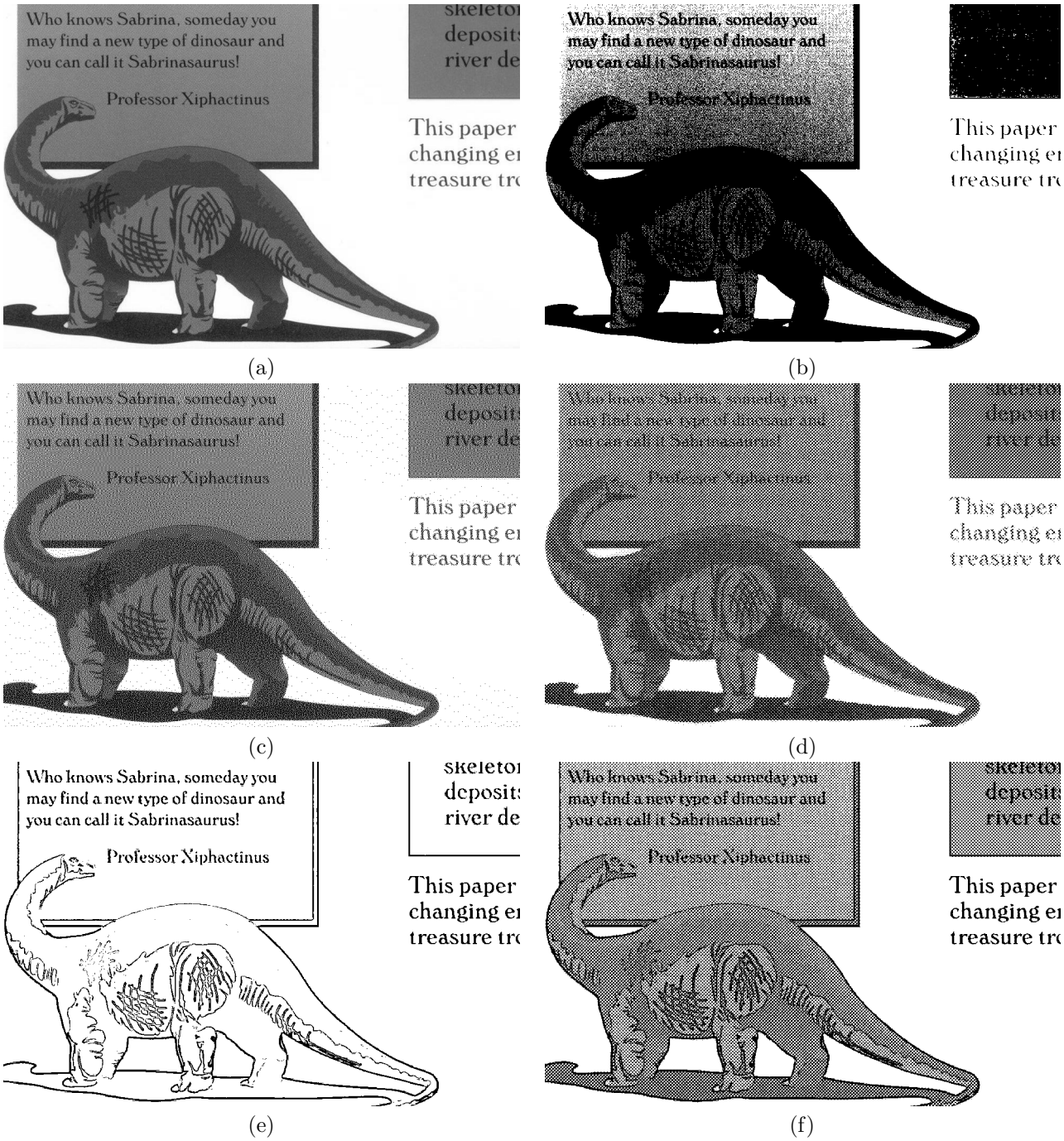


Figure 7. Binary images generated by FastFax, ReadableFax and conventional halftoning methods: (a) portion of original grayscale image as shown in Fig. 6, (b) binary thresholding with a constant threshold, G4 compressed size 187 KBytes, (c) Floyd Steinberg error diffusion, G4 compressed size 923 KBytes, (d) clustered dot screening, G4 compressed size 369 KBytes, (e) FastFax, G4 compressed size 82 KBytes, and (f) ReadableFax, G4 compressed size 322 KBytes.