BIT RATE REDUCTION OF VECTOR REPRESENTATION OF BINARY IMAGES

Yuki YAMAMOTO, Kei KAWAMURA, Hiroshi WATANABE

Graduate School of Global Information and Telecommunication Studies, Waseda University
1011, Okuboyama, Nishitomida, Honjyo, Saitama, Japan

ABSTRACT

Vector representation of binary images has an advantage of keeping high image quality for arbitrary scaling as well as editing capability of an object. However, the vector representation suffers from low compression efficiency compared with JBIG. In this paper, we show the main cause reducing coding efficiency and propose two methods to improve it. The proposed methods can reduce the file size of a binary image up to about 30–40 percent.

Index Terms— Image coding, vectorization, binary image, noise reduction, image quality

1. INTRODUCTION

Images are normally representated by raster format. However, raster representation is not suitable for arbitrary scaling. When the resolution of an image is reduced, the relations between width and length of line segments may not be kept. In addition, moire occurs in an area that has halftone dots. When the image is expanded, jaggy occurs on curves or slash lines.

On the other hand, vector representation has an advantage of keeping high image quality for arbitrary scaling. When an extracted edge is vectorized, image quality can be maintained because of spatial scalability. For this reason, vector representation of binary images is effective for resolution conversion.

In recent years, a demand for digital content such as comics and novels has been increasing. Most comics are usually printed on paper as hard copy. Besides, they are provided as binary images. By digitizing such papers using a scanner, we can display them on screens of PCs or portable devices. However, scanned image does not maintain the original quality since addition of the noise is inevitable during the scanning process. Moreover, when it is vectorized, coding efficiency is decreased because of this noise.

In this paper, we propose two novel techniques to improve the coding efficiency of vector representation. In Section 2, we show the main cause of coding inefficiency of major binary coding techniques and explain the conventional method to resolve it. In Section 3, we describe the proposed method to improve the coding efficiency. Evaluation of simulation results is shown in Section 4, and the conclusion is given in Section 5.

Table 1. Representation of binary image coding

Representation	Coding
Raster	MH
	MR
	MMR
	JBIG
	JBIG2
Vector	EPS
	SVG
	Flash

Table 2. Experiment condition

Scanner	Canon Canoscan LIDE30
Source images	Weekly magazine
Paper size	B5
Scanning resolution	600dpi
Component/Bit depth	Gray scale/8bit
Binarization threshold	0.6 (0:white, 1:black)

2. CODING EFFICIENCY

2.1. Comparison of coding efficiency

First, we perform a preliminary experiment to compare efficiency of conventional coding methods. The binary coding methods and vector representation are listed in Table 1.

The experiment condition is shown in Table 2. Ten images are applied in this experiment. Binary images are converted into MH, MR, MMR, JBIG and JBIG2 as raster representations. As vector representation, binary images are converted into EPS by most efficient software Potrace [1]. Since EPS does not include entropy coding, the direct comparison is not fair. Thus, EPS is compressed by bzip2 [2].

The resulting file sizes are shown in Fig. 1. It is found that SVG and Flash have the same performance as EPS. Thus, their results are omitted in this paper. The horizontal axis indicates the image number, and the vertical axis indicates the file size. It turns out that the vector representation is not enough to achieve high compression ratio compared with the raster representation.

Raster representation is lossless compression, while vector representation is lossy compression. Thus, the distortion

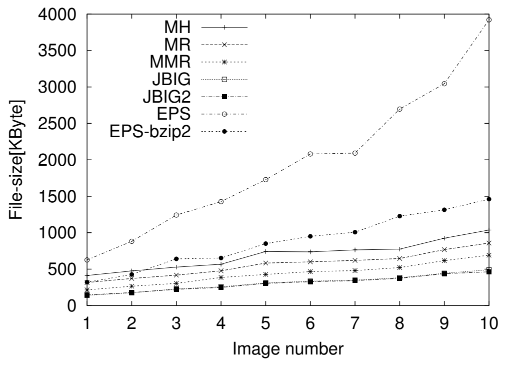

Fig. 1. File size of binary images

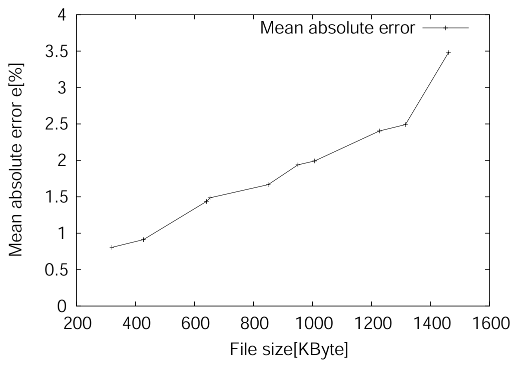

Fig. 2. mean absolute error

of the vector representation could be investigated by the mean absolute error. The mean absolute error "e" is defined as the ratio of sum of absolute value of difference between input image (A(x,y)) and modified image (A'(x,y)) by dividing the total number of pixels.

$$e = \frac{\sum_{x,y} |A(x,y) - A'(x,y)|}{XY}, \tag{1}$$

where X, Y denote a width and height of $A(x,y)$. The relation between "e" and file size is shown in Fig. 2. The horizontal axis indicates the file size and the vertical axis indicates the mean absolute error "e". The mean absolute errors varies between 0.7-3.5(%) in proportion to the file size. We find that a little distortion is generated in vector representation.

2.2. Causes of coding inefficiency

In this clause, the causes of coding inefficiency of vector representation are investigated. The basic element of vector graphics is the path. Each path consists of one or more curved and/or straight path segments. Each segment has a passing point called an anchor point at either end. The simplest path consists of two passing points connected by a straight line. Thus, vector coding efficiency depends on these passing points.

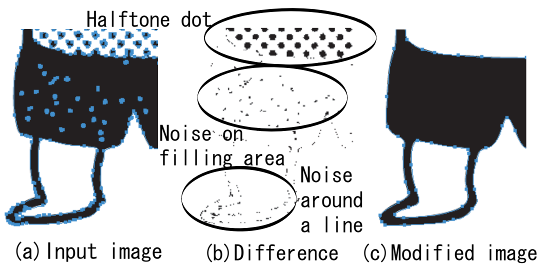

Fig. 3. Passing points of vector images

We have been studying that the causes of coding inefficiency are noise and halftone dots [3]. The passing points of input vector image are shown in Fig. 3(a). In addition, the causes of coding inefficiency is shown in Fig. 3(b). Comics generally contain many halftone dots. Thus, they are not suitable for vector representation because halftone dots have many passing points. Moreover, due to printed ink and rough paper, a lot of noise will be included in the image. Here, the noise is further classified into two categories.

- Noise on filling area that is daubed all over by black, and
- Noise around a line.

It has been found that the lines contain more noise of these two categories. Related results on coding inefficiency caused by redundant passing points owing to the noise around straight and curve lines have been presented in [4].

2.3. Gaussian filtering

For the purpose of improving the coding efficiency, the noise around the lines can be removed before vectorization using a Gaussian filter. A Gaussian filter can reduce the passing points by smoothing out the noise. Gaussian function is described by

$$\begin{aligned} A'(x,y) =& \frac{1}{N^2} \sum_i \sum_j exp(-\frac{((x+i)-x)^2}{2\sigma^2}) \\ & exp(-\frac{((y+j)-y)^2}{2\sigma^2}) A(x+i, y+j) \end{aligned} \tag{2}$$

$$\sigma = \text{standard deviation},$$
$$\frac{1}{N} = 0.398942280401.$$

Where value of N is calculated to normalize the total power of $A'(x,y)$ when the parameters i, j are set to [-15, 15]. As the value of σ increases, the image gets smoothed and passing points are reduced. Thus, the coding efficiency can be controlled by changing σ. The filtered image is shown in Fig. 3(c).

However, the detail of a figure is distorted with a large σ. Thus, Gaussian filtering is not the ideal operation for reduction of passing points.

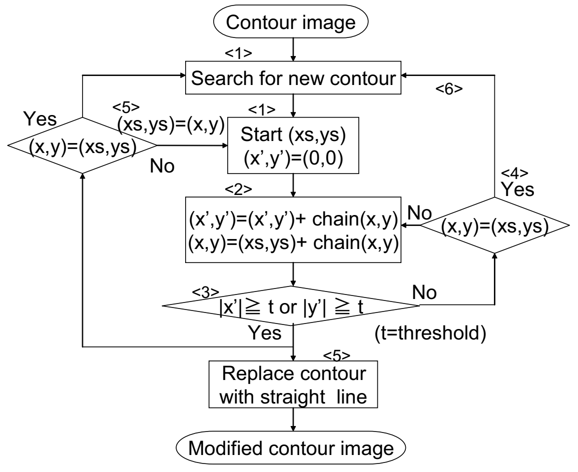

Fig. 4. Replacement algorithm

3. PROPOSED METHOD

We aim to remove the noise around straight and curve lines for vectorization. Here, two approaches are considered, (1) Contour Replacement with straight lines and (2) Adaptive Gaussian Filtering.

3.1. Contour Replacement

The contour with bunch of straight lines are replaced by Freeman's chain code [6]. This algorithm is shown in Fig. 4. The procedure is as follows.

1. Decide the starting point on the contour line.
2. Add the value of each coordinate using chain code along the contour.
3. When the sum of each coordinate value reaches a threshold, go to step 5.
4. When the both coordinates return to a starting point without reaching threshold, go to step 6.
5. Replace a contour line with a straight one, and set an ending point as the next starting point, go to step 2.
6. Do not replace a contour line with a straight one, go to step 1.

The procedure of contour replacement by straight lines is shown in Fig. 5(a) when the threshold is 5 pixels. The Bresenhum's algorithm [7] is used for the replacement of contour lines. In Fig. 5, one square indicates 1 pixel and one arrow indicates 1 chain code. Contour lines (Fig. 5(b)) are converted into straight lines (Fig. 5(c)) by this operation.

3.2. Adaptive Gaussian Filtering

We propose adaptive Gaussian filtering by selecting its standard deviation corresponding to the complexity of the image. There are many ways to define the complexity [8]. The number of pixels "N" contained inside a circle of radius 15 pixels

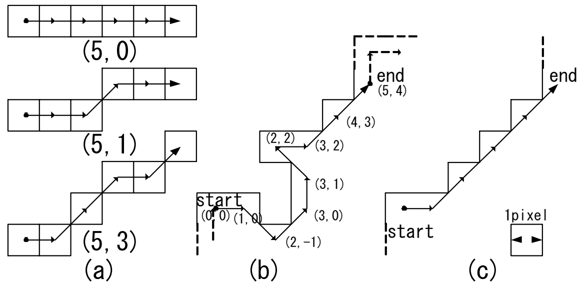

Fig. 5. Contour replacement

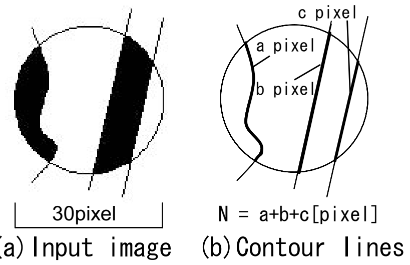

Fig. 6. Definition of complexity

(Fig. 6) is used as a control parameter. By changing the σ of the Gaussian filter according to N, we can realize efficient filtering to reduce the noise in images.

4. EXPERIMENT

4.1. Evaluation

Two proposed methods are evaluated by a file size and the number of contour lines. Since vector representation is lossy compression, the distortion between input image and vectorized one always occurs when lines are approximated. Thus, the evaluation method using the difference is not effective for the proposed method. Here, the number of contour lines of images is introduced as a quality assessment.

It has been shown that the number of contour lines indicates image quality and is proportional to the file size [5]. Since a shape of the image is expressed by the contour line, the image is distorted when the number of contour lines decreases. Thus, the distortion of the images can be evaluated by the number of contour lines. The relation between file size and the number of contour line is applied as an alternative method of the rate-distortion curve.

4.2. Contour Replacement

The experimental result, when the threshold of each axis is set to 5, is shown in Table 3. The proposed method (contour replacement) reduces the file size than the conventional method (Gaussian filter with σ=1). Further, in case of (Gaussian filtering (σ = 1) & Contour Replacement) , the file size can be

Table 3. Evaluation of the proposed method

Type	Operation	File-size [KByte]	Number of contour line
Conventional method	No operation	114.2	1298
	Gaussian filter(σ=1)	86.7	1249
	Gaussian filter(σ=2)	74.0	1219
	Gaussian filter(σ=3)	68.8	1192
	JBIG	98.8	1696
Proposed method	Contour Replacement	84.5	1269
	Gaussian filter(σ=1) &Contour Replacement	75.9	1224
	Adaptive Gaussian Filter	69.2	1226

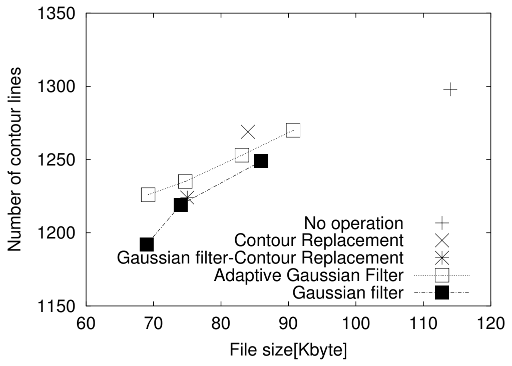

Fig. 7. Evaluation of the proposed method

reduced to 75.9 KB. In Fig. 7, the horizontal axis indicates the image number, and the vertical axis indicates the file size. As compared with the Gaussian filter, the proposed method provides a better result. A part of the input image is shown in Fig. 8(a). The shape of the image is distorted in Fig. 8(b) by the conventional method. On the other hand, the noise around lines is reduced without deterioration in quality by combining conventional Gaussian filtering and the proposed method (Fig. 8(c)).

4.3. Adaptive Gaussian Filtering

In this experiment, we apply the following values of parameter σ.

$$\sigma = \begin{cases} 3 & if \quad 0 \le N < 45 \\ 2 & if \quad 45 \le N < 60 \\ 1 & if \quad 60 \le N \end{cases} \qquad (3)$$

Coding efficiency of adaptive Gaussian filtering is the same as that of Gaussian filtering with σ=3 in Table 3. The proposed method can improve the file size up to about 40 percent. The noise around the straight and curve lines is removed and details of the figure are maintained as shown in Fig. 8(d).

In Fig. 7, it is supposed that two curves of proposed methods cross each other. It is found that the contour replacement provides better performance when the compressed file size is larger than 75 Kbytes.

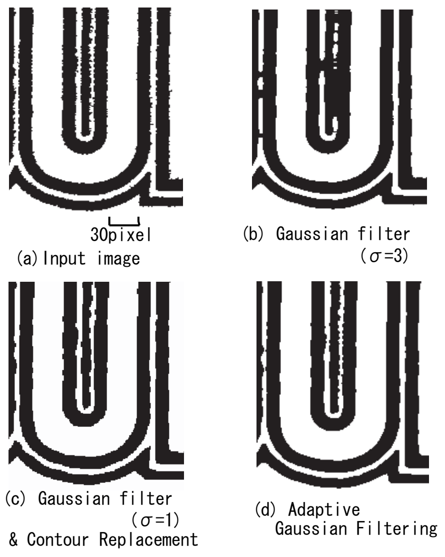
30pixel
(a) Input image
(b) Gaussian filter (σ=3)
(c) Gaussian filter (σ=1) & Contour Replacement
(d) Adaptive Gaussian Filtering

Fig. 8. Modified images

5. CONCLUSION

In this paper, we proposed two methods to improve the coding efficiency in vectorization of binary images. These were the contour replacement and adaptive Gaussian filtering. These methods were evaluated by the relation of a file size and the number of contour lines. Both of them provided better results than the conventional approach.

ACKOWLEDGMENT

We thank the Okawa Foundation for partial support of this research.

6. REFERENCES

[1] Peter. Selinger, “Potrace, ” http://potrace.sourceforge.net/

[2] J. Seward, “bzip2, version 1.0.2, ” http://sources.redhat.com/bzip2/

[3] K. Kawamura, H. Watanabe, H. Tominaga, “Vector Representation of Binary Images Containing Halftone Dots, ” 2004 IEEE International Conference on Multimedia and Expo, TP2-1, Jun. 2004.

[4] Y. Yamamoto, K. Kawamura, H. Watanabe, H. Tominaga, “A Study on Rate Reduction in vector representation of binary images, ” PCSJ2004, P-2.04, Nov. 2004.

[5] Y. Yamamoto, K. Kawamura, H. Watanabe, H. Tominaga, “A Study on Evaluation of Quality Metrics for Vectorized Binary Images,” AVM-48-4, Mar. 2005.

[6] H. Freeman, “Computer Processing of Line-Drawing Images, ” Computing Surveys, vol.6, no.1, pp.57-97, 1974.

[7] J. E. Bresenham, “Algorithm for Computer Control of a Digital Plotter, ” IBM SystemsJ., Vol.4, No.1, pp.25-30, 1965.

[8] H. Aoyama, M. Kawagoe, “Segmentation of Planar Curves Based on Complexity. Extraction of Complex area and Its Interpretation. ” Compurter vision on IPSJ, 93-CV-82, No.25, pp.81-86, Mar. 1993.