MVA'94 IAPR Workshop on Machine Vision Applications Dec. 13-15, 1994, Kawasaki

A HIGH-SPEED CHARACTER CONTOUR-FILL METHOD USING AN EDGE-FLAG LIST

Kazuki NAKASHIMA†, Masashi KOGA†, Katsumi MARUKAWA†, Yoshihiro SHIMA†, and Yasuaki NAKANO††.

†Central Research Laboratory, Hitachi, Ltd. 1-280 Higashi-koigakubo, Kokubunji-shi, Tokyo 185, Japan
††Faculty of Engineering, Shinshu University. 500 Wakasato, Nagano-shi, Nagano 380, Japan

ABSTRACT

This paper describes a new, high-speed contour-fill method for alpha-numeric and Japanese Kanji characters developed to generate digital bitmap image (binary image) patterns from contour lines. The contour lines are expressed by Freeman's chain code. We call this method the Improved Edge-flag method. The speed of this method is increased by using an edge-flag list which stores the coordinates of the edge-flag pixels. A comparison of two conventional contour-fill methods with our method shows that ours is about 4-8 times faster in filling character contours.

1. INTRODUCTION

The amount of data contained in character contour-pattern information is far less than that contained in image information. Therefore, contour information has many advantages; For example, it can be processed at high speed, and used in displays, printouts, and recognition processes. The need for high-speed character-image generation has risen sharply in recent years due to the rising demand for new machine interfaces such as desk-top publishing (DTP) using outline fonts which require high-speed character-contour filling.

Many conventional contour-fill methods have been proposed [1][2]. In this paper, we focus on two conventional methods using Freeman's chain-code (as shown in Fig. 1) to express the contour lines. One is the edge-flag method[3] which is an improved version of Bryan's edge-flag method[4]. The other is the edge-fill method[5][6] which is also derived from Bryan's edge-fill method[4]. The conventional edge-flag method has been proposed for general shapes. Thus, it has been applied to alpha-numeric characters and compared with the edge-fill method in terms of speed[5][6]. However, applicability of these two methods to Japanese Kanji characters has not yet been demonstrated.

This paper compares the speed performance of the two conventional methods when applies to Kanji characters. The results clarify the disadvantage of the edge-fill method relative to complex patterns such as Kanji characters. The edge-flag method is improved to facilitate applicability to Kanji patterns. Using an edge-flag list which stores the coordinate values of edge-flag pixels improves the speed of the edge-flag method. The speed of the two conventional methods and the improved edge-flag method were compared for alpha-numeric and Japanese Kanji character processing. In the conventional edge-flag method, the processing time for numerals was 2.4 ms, while in the improved edge-flag method, it was only 0.3 ms. For Kanji, the times were 2.8 ms and 0.7 ms, respectively. This comparison shows that our new method is about 4-8 times faster than the conventional edge-flag method.

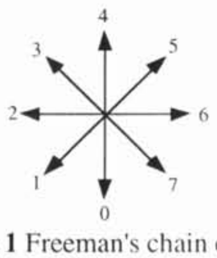

Fig. 1 Freeman's chain code.

2. CONVENTIONAL CONTOUR-FILL METHODS

2.1 Left- and Right-terminal Pixels

Figure 2(a) shows a digital bitmap (binary) image of the number "6". Figure 2(b) shows Fig. 2(a) expressed by Freeman's chain code. The chain-code numbers (from 0 to 7) arranged along the contour lines are stored as a list. The x-y coordinates of each contour-line pixel are generated from this chain-code list. Images can be expressed as a G(j,i) matrix. From here forward, filling process refers to using this list to control pixel filling.

Filling processes are predicated in this section the process of based on determining the edge-flag pixels from the chain-code list. Edge-flag pixels are classified into two types, left-terminal pixels and right-terminal pixels, as shown in Figure 3. A left- (right-) terminal pixel is actually the left-most (right-most) pixel for each run-length of the image pattern. To generate contour-filled images all pixels between the pairs of left- and right- terminal pixels are filled.

2.2 Conventional edge-flag method

The conventional edge-flag method consists of two steps, as shown in Figure 4. The first step is flag-image generation: by searching the chain-code list, all pixels on the contour line are judged as either left- or right-terminal pixels. This determination is done using maps, as shown in Figure 5. Fig. 5 shows two combination maps for neighboring chain codes. The map in Fig. 5(a) is used to identify left-terminal pixels, and the one in Fig. 5(b) is used for right-terminal pixels. In literature cited [1], the vertical edge-status is used; however, the status makes essentially no difference in these maps. The left- and right-terminal pixels are flag pixels set as marks in the flag image (Figure 6(a)) which is the same size as the character image.

In the second step, all pixels between the pairs of flag pixels on each horizontal line are filled, and the far-left flag pixel of each pair is eliminated (Fig. 6(b)).

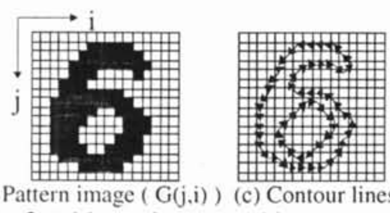

Fig. 2 A binary image and its contours.

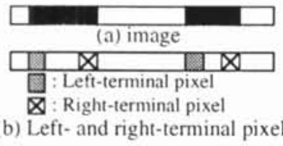

Fig. 3 Left-terminal pixel and right-terminal pixel of a scanline.

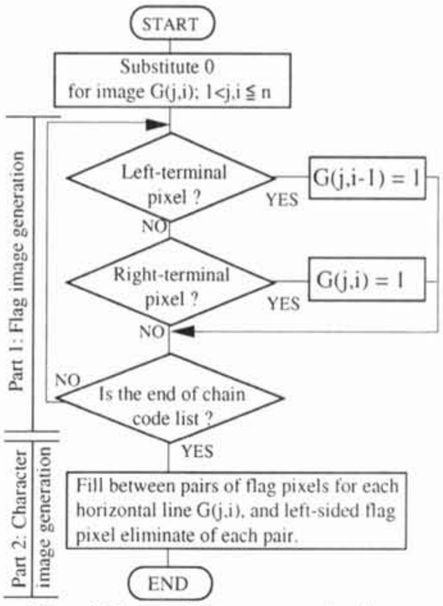
START
Substitute 0 for image G(j,i); 1<j,i ≦ n
Part 1: Flag image generation
Left-terminal pixel ?
YES
G(j,i-1) = 1
NO
Right-terminal pixel ?
YES
G(j,i) = 1
NO
NO
Is the end of chain code list ?
YES
Part 2: Character image generation
Fill between pairs of flag pixels for each horizontal line G(j,i), and left-sided flag pixel eliminate of each pair.
END

Fig. 4 Flow of the conventioal edge-flag method.

(a) Map for determining left-terminal pixels

d_{k-1} \ d_k	0	1	2	3	4	5	6	7
0	1	1	0	X	1	1	1	1
1	1	1	0	0	X	1	1	1
2	1	1	0	0	0	X	1	1
3	1	1	0	0	0	0	X	1
4	1	1	0	0	0	0	0	X
5	X	1	0	0	0	0	0	0
6	0	X	0	0	0	0	0	0
7	1	1	X	1	1	1	1	1

(b) Map for determining right-terminal pixels

d_{k-1} \ d_k	0	1	2	3	4	5	6	7
0	0	0	0	X	1	1	0	0
1	0	0	0	0	X	1	0	0
2	0	0	0	0	0	X	0	0
3	1	1	1	1	1	1	X	1
4	1	1	1	1	1	1	0	X
5	X	1	1	1	1	1	0	0
6	0	X	1	1	1	1	0	0
7	0	0	X	1	1	1	0	0

Note : X denotes an impossible combination.

Fig. 5 Maps for determining terminal pixels.

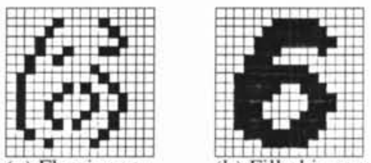

Fig. 6 A flag image obtained by the conventional edge-flag method and its filled image.

2.3 Conventional edge-fill method

The conventional edge-fill method reverses all pixels on the horizontal line of an image from a pixel on the character contour line to the right side of the image. Figure 7 shows an example of the procedure followed in this method. Figures 7(a)-(c) show the procedure for the outer contour image, and Figs.7(d)-(f) show the procedure for the inner contour image. As is clear, this method searches out the contour line and reverses to the far right side of the image, filling all pixels in its path.

The edge-fill method also uses the maps to determine flag pixels as in the edge-flag method. The logic of these maps (Figs. 5(a) and (b)) is given by the following two expressions of inequality. In identifying terminal pixels, either the maps or the following expressions of inequality can be used.

1) Identifying left-terminal pixels:
   If $(d_k+6) \bmod 8 > (d_{k-1}+1) \bmod 8$, reverse all pixels from the contour-line pixel to the far right side of the image, including the pixel on the contour-line.

2) Identifying right-terminal pixels.
   If $(d_k+2) \bmod 8 > (d_{k-1}+5) \bmod 8$, reverse all pixels from the contour-line pixel to the far right side of the image, excluding the pixel on the contour-line.

The d_k is a Freeman's chain code, which appears k times on a contour line. N mod 8 is the remainder when N is divided by 8.

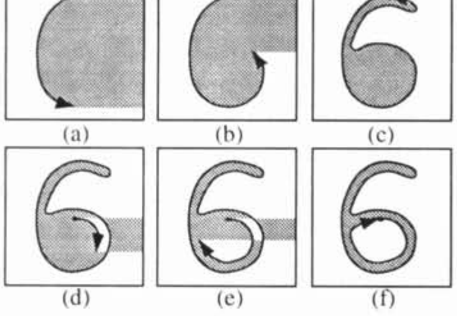

Fig. 7 An example of the edge-fill procedure.

3. IMPROVED EDGE-FLAG METHOD

The conventional edge-fill method suffers a significant loss of speed as the contour structures grow more complex because the demands of the reversal process to the right side of the image increase as the contour becomes more complex. The conventional edge-flag method suffers from a long processing time needed to search the flag image.

If the x-y coordinates of the flag pixels are stored in a list, called an edge-flag list, the processing time of the conventional edge-flag method can be decreased. Figure 8 shows the structure of an edge-flag list. The number of vertical elements in the list corresponds to the number of vertical pixels in the character (flag) image, and the vertical element values correspond to the y-coordinates of the flag pixels. The horizontal elements of the edge-flag list is stored the values of the x-coordinates of the flag pixels for each run-length. Because the x-coordinate values for each horizontal line appear along the contour line, these values must be rearranged in order of size. Figure 9 shows a flow of the improved edge-flag method.

4. EXPERIMENTAL RESULTS AND CONSIDERATION

In this experiment we used 280 numerals, 280 alphabetical symbols, and 140 Kanji characters. The experimental apparatus was a 76-MIPS workstation and the data structure for the images was 1 byte per pixel. Some example pattern images are shown in Fig. 10. The images were 56 x 70 pixels in size, with the pattern located nearly in the center of the image.

The two conventional contour-fill methods and our new improved edge-flag method were

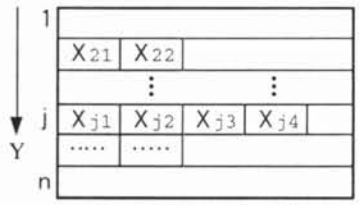

Fig. 8 Edge-flag list.

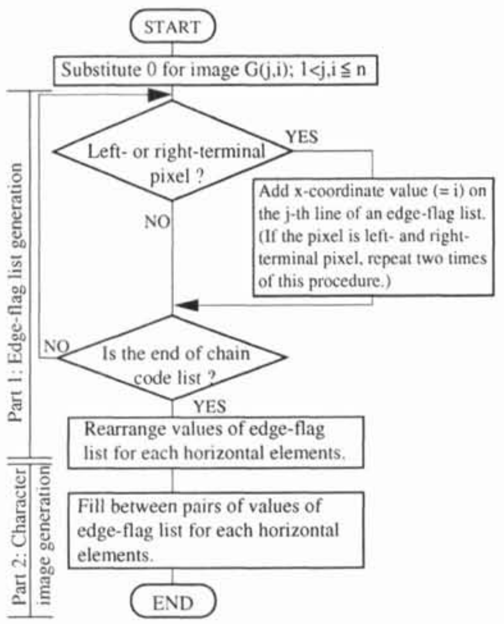
START
Substitute 0 for image G(j,i); 1<j,i ≦ n
Left- or right-terminal pixel ?
YES
Add x-coordinate value (= i) on the j-th line of an edge-flag list. (If the pixel is left- and right-terminal pixel, repeat two times of this procedure.)
NO
Is the end of chain code list ?
NO
YES
Rearrange values of edge-flag list for each horizontal elements.
Fill between pairs of values of edge-flag list for each horizontal elements.
END
Part 1: Edge-flag list generation
Part 2: Character image generation

Fig. 9 Flow of the Improved edge-flag method.

compared in terms of speed performance. The results are shown in Table 1. Processing time for the conventional edge-fill method for a Kanji pattern was 4.5 ms, clarifying the unsuitability of this method for complex contour patterns. With our new method, the processing time for numerals was 0.3 ms, for alphabetical symbols it was 0.3 ms, and for Kanji it was 0.7 ms, about one eighth, one eighth, and one fourth, respectively the times for the conventional edge-flag method.

In the conventional edge-flag method all pixels of the flag-image must be searched individually to detect whether or not to fill them because the flag pixels store the image. In contrast, the improved edge-flag method searches only the edge-flag list, which has far less entries than the number of image pixels. Thus, the decrease in the number of pixels requiring searching has effectively reduced the filling time.

5. CONCLUSION AND FUTURE WORK

Two conventional contour-fill methods and the improved edge-flag method were evaluated for Arabic numerals, Roman alphabet symbols, and Japanese Kanji characters. In doing so, we clarified the unsuitability of the edge-fill method for Kanji character filling. By improving the edge-flag method, we reduced the processing time and obtained satisfactory results.

The data structure of the binary images used in this study was 1 byte per pixel. The processing time for hardware implementation is expected to be affected by the data structure. Thus, the next step in our research will be to investigate the optimum data structure for images.

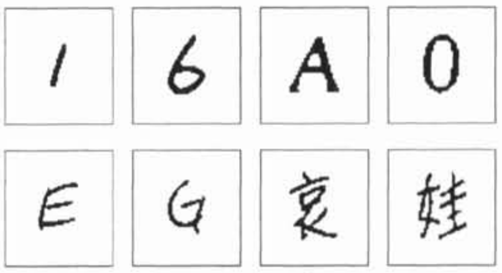

Fig. 10 Examples of pattern used for examination.

Table 1 Character processing times.

Pattern	C. edge-flag method[ms]	C. edge-fill method[ms]	I. edge-flag method[ms]
numeral	2.4	1.9	0.3
alphabet	2.3	2.1	0.3
Kanji	2.8	4.5	0.7

C. : Conventional, I. : Improved

REFERENCES

[1] Shani et al., "Filling Regions in Binary Raster Images: A Graph-Theoretic Approach," Proc. SIGRAPH'80, Vol. 14, No. 3, pp. 321-327(1980).

[2] Pavlidis et al., "Contour filling in raster graphics," SIGRAPH81, Vol. 15, No. 3, pp. 29-36 (1981).

[3] Li et al., "Border Following and Reconstruction of Binary Pictures Using Grid Point Representation," Trans. of Japanese IEICE., Vol. J65-D (No. 10), pp. 1203 - 1210 (1982.10, in Japanese).

[4] Bryan et al., "The Edge Flag Algorithm - A Fill Method Raster Scan Displays," IEEE Trans. C-30[1], pp. 41-47 (1981).

[5] Nakashima et al., "A Contour Fill Method for Alpha-numeric Character Image Generation," Proc. ICDAR'93, pp. 722-725, October 20-22, 1993.

[6] Nakashima et al., "A High Speed Contour Fill Method for Character Image Generation," IEICE Trans. Inf. & Syst., pp. 832-838, Vol. E77-D, No. 7, July, 1994.