A HIGH SPEED RASTER-TO-VECTOR CONVERSION USING SPECIAL HARDWARE FOR CONTOUR TRACKING

Shigeyoshi Shimotsuji, Akio Okazaki, Osamu Hori and Shou Tsunekawa
Research and Development Center
Toshiba Corporation

1, Komukai Toshiba-cho, Saiwai-ku
Kawasaki 210, Japan

ABSTRACT

This paper describes a high speed raster-to-vector conversion technique which is applicable to many fields in drawing processing. The major feature of the proposed technique is that raster-to-vector conversion can be realized by one simple raster scanning process to an input image and by processings performed only with contour tracking due to the fact that a significant part of a drawing image is only the boundary between black and white regions.

In order to speed up raster-to-vector conversion, the raster scanning process and processings with contour tracking, which consume much time but are repetitions of simple processings, have been implemented by a special hardware.

Reduction of about 90% in raster-to-vector conversion time is possible by using the accelerator. The effectiveness and flexibility of the proposed processings are discussed, showing experimental results of automatic extraction of polygons which represent houses from map images.

1. Introduction

Automatic data entry to geographic information systems and CAD systems using digital image processing and pattern recognition techniques has been studied to shorten the initial data input time[1,2,3, 4]. Conversion from image data (raster) to graphic data (vector) is one of the key techniques to accomplishing the above aim, and it is applicable to many fields in line-drawing processing, such as data compaction, line editing, and shape analysis.

This paper first proposes a new raster-to-vector conversion algorithm based on a repetition of pixel access along a contour of an object in an image or a skeleton in the thinned image (Section 2). Line-structure extraction by raster-to-vector conversion is an important process for understanding line-drawings. However, generally speaking, raster-to-vector conversion is time-consuming for large drawings because of the amount of image memory access. Considering that a significant part of an image in the raster-to-vector conversion is only the boundary between black and white regions, the authors have developed an efficient algorithm for the raster-to-vector conversion constructed without meaningless image memory access, such as an access to a region where every pixel is white. That is, using the algorithm, start points for contour tracking are first quickly obtained by raster scanning an image. Then, all image memory accesses in later processing, such as logical filtering, bending point detection and extraction of graph-like structure from a drawing, are carried out merely by contour tracking from the start points.

Next, it is proved that the proposed algorithm is easily implementated by hardware, while still retaining flexibility (Section 3). The basic idea for the hardware implementation is that the raster-to-vector conversion can be realized by one simple raster scanning access to an input image and by several processings only along contours. Thus, the developed hardware is composed of two parts; one is an accelerator for start point detection in raster scanning an image, and the other is for processings with contour tracking. In order to retain flexibility, the latter part is constructed from a kernel unit of image memory address calculation for contour tracking and attached units for processing during the contour tracking. Setting a flag to an image memory, logical filtering, bending point detection, and error calculation of an approximate line have been implemented as the latter unit.

Finally, effectiveness and flexibility of the proposed accelerator are discussed, showing experimental results of automatic extraction of polygons which represent houses from map images(Sections 4 and 5).

2. Raster-to-vector conversion based on contour tracking

Raster-to-vector conversion consists of the following two sub-processes;

(1) Generation of chain-code or (x,y)-coordinate streams representing the line-structure of a drawing image. In raster-to-vector conversion, a chain-code stream and an (x,y)-coordinate stream are equivalent. This kind of stream is called 'chain-code' for simplicity.

(2) Conversion from the chain-code streams to vector description, using a straight line or curve fitting algorithm.

It should be noted that image memory access is limited in the chain-code generation process (1), which is regarded as a conversion from two-dimensionally represented information (image) to one-dimensionally represented information (symbolical or numerical

description). The data amount dealt with in the process (2), i.e. the line fitting process, is relatively small, because of the data compression by the former process. Thus, it is necessary to accelerate the former process in order to speed up the whole raster-to-vector conversion time for large drawings. While several efficient algorithms have been reported concerning the line fitting process [5,6], there have been very few reports published on the chain-code generation process from the raster-to-vector conversion efficiency improvement viewpoint. Here, the authors propose a simple and general technique for raster-to-vector conversion, especially taking into consideration the improvement of the chain-code generation efficiency.

Considering a significant part for drawing image processing is only the boundary between black and white regions, every operation in the chain-code generation process can be implemented by iterations of several operations based on such control as contour tracking. And the simplicity of the contour tracking operation enables the realization by a small scale hardware.

Furthermore, in the line fitting process, an operation by scanning a chain-code stream, such as bending point detection by calculating the angle of an arc composed of three points along a contour [7] and area calculation of a region constructed by a contour and an approximate line, is equivalent to a line (skeleton) tracking operation. Line tracking and contour tracking are the same operation, in the sense that each tracking direction is determined by the values of the 8-neighborhood pixels around a current tracked pixel.

Therefore, almost all the processings in raster-to-vector conversion can be realized by the operation based on contour tracking.

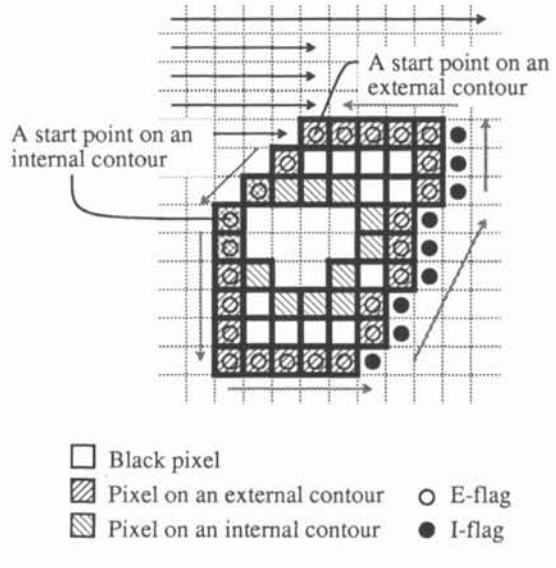

Fig.1 Start point detection and external contour tracking

It is trivial that line fitting processing can be performed based on line tracking. Thus, we concentrate on the discussion about processings in the chain-code generation. The following functions are generally used in the generation.

(a) Connected component extraction.
(b) Noise reduction.
(c) Logical filtering.
(d) Elimination of noise-segments caused by thinning.
(e) Generation of chain-code streams of contours or skeletons.

We show efficient contour-tracking-based algorithms for each function.

(a) Connected component extraction:

This processing could be implemented by the conventional region labeling method. However, it is not practical because a large amount of image memory is required for a large drawing. Therefore, the following algorithm is applied to connected component extraction. The basic idea of the algorithm is that, as each connected region has one external contour and several internal contours if it has some holes, the connected component extraction can be realized by the detection of external/internal contours.

This algorithm can be described by the following two steps (see Fig.1).

(i) Find a point on a contour by scanning an image.
(ii) Track the whole contour starting with the point.

The point which is detected in step (i) is called a start point for tracking. In order to avoid duplicated detections of the same contour, it is tracked by setting flags to a flag memory. Rosenfeld et al. pointed out that two bits per pixel for t. . flag are required in contour detection when a line drawing has several holes [8]. However, this paper shows that one bit is sufficient. Furthermore, as there is not a discussion to identify which detected contour is external or internal, it is also proved that the identification is easily realized.

The start point is defined as the first detected point on the contour in raster scanning a image, whose right or left neighborhood is white. A flag at every tracked pixel is set in both external and internal contour tracking (this flag is called the e-flag). If every external and internal contour are disjoint, only e-flag is necessary to avoid the duplicated detection. Another flag (called the i-flag) is prepared for general situation, where common pixels exist betweeen external and internal contours. The condition for setting the i-flag is

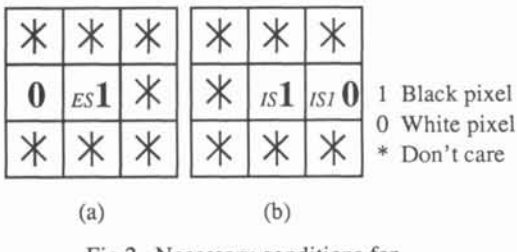

Fig.2 Necessary conditions for external start point(ES) and internal start point(IS)

that, in both external and internal contour tracking, the right neighborhood pixel is white and this white pixel belongs to the current observing white region. And the location of the i-flag is at this right neighborhood white pixel (see Fig. 1). Then, the necessary and sufficient condition for a start point can be described as follows:

(i) A start point of an *external* contour is a pixel whose 8-neighborhood pattern is matched to Fig.2 (a), and whose e-flag (i.e. a flag at ES in Fig. 2 (a)) is not set.

(ii) A start point of an *internal* contour is a pixel whose 8-neighborhood pattern is matched to Fig.2 (b), and whose i-flag (i.e. a flag at IS1 in Fig.2 (b)) is not set.

There is no collision between an e-flag and an i-flag, because the e-flag is set at a black pixel and the i-flag is set at a white one. Therefore, only one bit per pixel is sufficient in the algorithm. This flag manupilation can be easily accomplished by a table look-up method with 8-neighborhood values and a tracking direction.

(b) Noise reduction:

A noise region, such as a salt-and-pepper noise, and a character region in a drawing image which is not needed to be vectorized, are defined as a small black/white region. It is sufficient for such noise reduction that the start point information for a short contour is deleted from the start point table, except that an operation which alters an input image, such as thinning, must be carried out after the nose reduction. In the case, pixels inside the short contour must be altered.

(c) Logical filtering:

Every logical filtering operation can be performed in contour tracking. If the operation is topologically invariant with regard to input image, such as thinning and branch/terminal point detection, it can be easily implemented by contour tracking. That is, it is carried out by a repetition of a table look-up method with neighborhood values of each tracked pixel. Several thinning algorithms by contour tracking have been reported[9,10]. If the operation is not topologically invariant, such as shrinking, the control for the operation is complicated because the start point table, which represents the topological information for an input image, must be altered in each operation.

(d) Elimination of noise-segments caused by thinning:

Occurrence of noise-segments can not be avoided in a thinning operation. Such noise-segments, which are defined as a short line, can be deleted by line tracking.

(e) Generation of chain-code streams of contours or skeleton:

It is trivial that a chain-code can be generated by contour or line tracking.

Based on the above discussion, it can be concluded that every processing in raster-to-vector conversion can be realized by

(i) a raster operation for start point detection, and

(ii) operations which are carried out by contour tracking.

The latter operation is composed of a kernel part, which calculates the frame memory address for contour tracking, and several attached processing parts during contour tracking.

3. An accelerator for raster-to-vector conversion

A two-level hierarchical architecture has been designed for the raster-to-vector conversion discussed in the previous section. The design concept is that every operation involving image memory access by contour tracking is implemented by a special hardware (accelerator), and the control of the operation is carried out by a micro-processor. Fig. 3 shows a block diagram of the developed hardware system. The system consists of hierarchical processing units of the micro-processor and the accelerator, and 8K x 8K pixel frame memories, which can store the entire image of a JIS (Japanese Industrial Standard) A1 size (841 x 549 mm) drawing scanned at 200 dots per inch.

The micro-processor unit (MPU), which is composed of MC68020, first drives the accelerator to generate a start point table. Then, by controlling the accelerator with the start point table, the MPU proceeds with a chain-code generation process and a line fitting process.

The accelerator consists of two units. One is for start point detection and the other is for contour tracking operation.

The start point detection unit is first driven in raster-to-vector conversion. It uses two frame memories in the operation; one is for an original input image and the other is for a flag. According to the discussion in the previous section, a start point for an external contour must be a black pixel whose *left* hand neighbor is white and the flag is not set at this black pixel, and, vice versa, a start point for an internal contour must be a black pixel whose *right* hand neighbor is white and the flag is not set at the white pixel. Therefore, the unit is designed so that the only x-directional changes in value can be detected quickly. That is, the frame memory is designed so that x-directionally successive

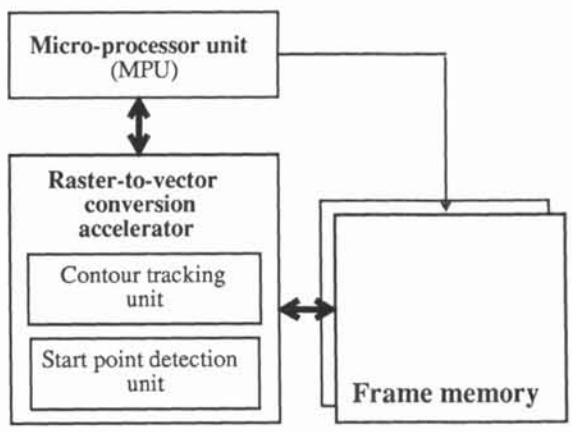

Fig.3 Block diagram of fast raster-to-vector conversion system

16 pixels can be simultaneously accessed. While the unit reads the successive 16 pixels of the frame memory, in which an original image is stored, it detects value change pixel(s) among them. If there is no such pixel in it, the unit accesses the next 16 pixels and tests them. Otherwise, the unit reads the corresponding flag frame memory and tests whether a flag is set. If there is not a flag, the unit informs start point detection to the MPU (see Fig. 4).

The contour tracking operation unit is driven by the MPU according to the start point table. That is, the MPU first indicates such initial conditions as the location of the start point and the initial direction for tracking. The MPU next starts contour tracking. The unit is designed so that one of such operations as logical filtering, angle calculation, and error calculation between a contour and an approximate line, can be performed during tracking. The unit continues the operation until a terminal condition is satisfied, such as arriving at a specified point by tracking, a processing region limit, or the limit number of pixels to be tracked, which is set by the MPU at the start of the operation.

According to the discussion in the previous section, the unit consists of a kernel part which calculates the frame memory address for contour tracking, and attached parts for processing during contour tracking. The kernel part is composed of the following four parts (see Fig. 5).

(1) Sequencer (controller).
(2) Cash memory.
(3) Frame memory address controller.
(4) Tracking direction tables.

In order to speed contour tracking, a method has been proposed wherein the 8 neighborhood values are assembled to one byte, has been proposed [11]. However, by this method, 8 bits are needed for every pixel in a frame memory. Therefore, a cash memory is used for fast contour tracking. The size of cash memory is 9 (= 3 x 3) bits, and the value of tracked pixel and its neighborhood values are stored. The next tracking direction is obtained by the tracking direction table according to the nine bit code of the cash memory and previous tracking direction. Then, the sequencer changes the frame memory address of the tracked pixel. In changing the address, the contents of the cash memory are altered efficiently in a way that only three or five bits of cash memory are loaded from the frame memory. Simultaneously, one of the attached operations is also performed.

The tracking contour order is determined by the tracking direction table. That is, the next address in contour tracking is determined by the values in the cash memory and the previous tracking direction. By changing the table content, both 4-connected and 8-connected tracking is selectable.

As the attached operations, an operation of setting a flag to a frame memory, logical filtering, angle calculation, and error calculation for line fitting have been implemented.

The flag setting operation is carried out by the sequencer according to a condition set by the MPU.

The 3 x 3 logical filtering operation is carried out with the values in cash memory and a logical filtering table. This operation is mainly used in thinning, and the detection of terminal and branch points from a thinned line.

In the angle calculation unit, the angle of an arc composed of three points along the contour is obtained. If the angle is less than a threshold value, its value and the location of the arc are memorized as candidates for a bending point.

In the error calculation unit, the area of the region constructed by a contour and an approximate line, which is used for evaluating of the approximation, is calculated.

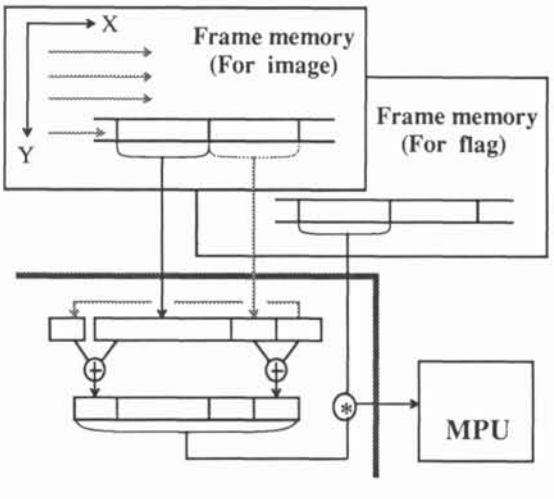

Fig.4 Start point detection unit

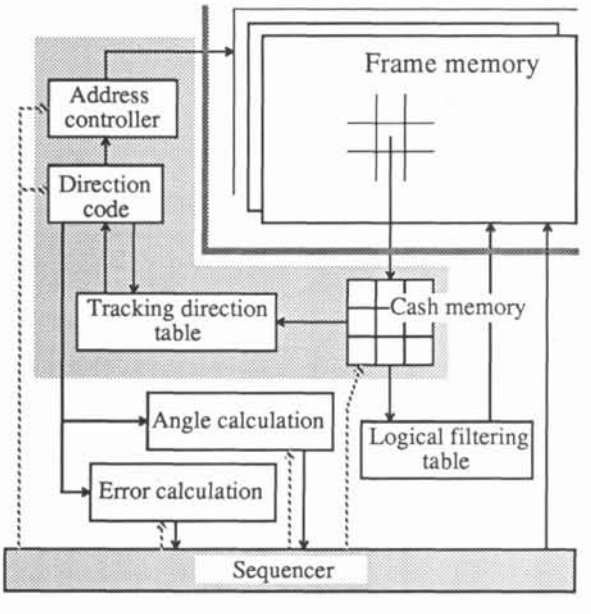

Fig. 5 Configuration of contour tracking unit

4. A map image analysis with raster-to-vector conversion

As an example of practical applications using the proposed raster-to-vector conversion, the following presents an automatic extraction of polygons representing houses from map images. (These polygons are called 'house-polygons' from now on.) An automatic extraction of house-polygons is very useful for the initial data entry to a geographic information system. For example, in a 1/2500 map for urban planning, whose sheet size is 800 x 600 mm, there are about ten thousand house-polygons, and more than 70% of all vectors in a map includes in house-polygons.

The extraction process consists of two sub-processes. In the first process, candidates for house-polygons are extracted with the proposed accelerator. Here, the accelerator is efficiently used. That is,

(1) Internal contours in a map image, which are candidates of house-polygons, can be extracted at very high speed by the accelerator, and

(2) Bending points of internal contours, which are important features in house recognition, are found very fast.

Then, in the second process, the MPU identifies true house-polygons by the following features;

(a) A house-polygon has at least four corners with about 90 degree angles.

(b) A polygon shape is simple in a sense (the simpleness measure is defined as (area)/(perimeter) **2 in this process).

(c) The polygon size is within some specific range.

The raster-to-vector conversion for house-polygon extraction is explained in detail. First, the start-points for both external and internal contours are quickly searched for by the accelerator. The locations of the start points are stored in a table. In this process, two start point tables are made; one is for external contours, and the other is for internal ones. Thinning operation is carried out only on internal contours. It is continued until half of the line is thinned. The resultant internal contours can be regarded as skeletons of the original lines. Finally, the bending points of the contours are detected using the accelerator. Here, a corner is defined as a bending point with a sharp angle less than 120 degrees. A sequence of the obtained corner locations (x-y coordinates) for each internal contour is memorized as a candidate for a house-polygon.

In the second process, the candidates are distinguished from other polygons by testing the condition (a)-(c).

5. Experimental Results

An experiment was carried out using several 1/2500 maps for urban planning, scanned at 200dpi. Fig.6 shows the result of automatic house-polygon extraction from a map.

Table 1 shows the average processing time per map when using the accelerator and when using only the MPU which directly accesses to the frame memory. As a result, a reduction of about 90% in raster-to-vector conversion time was obtained.

The accuracy of the house-polygon extraction algorithm described in the previous section is shown in Fig.7. In the first process, 2% of true house-polygons could not be detected, while 33% of the detected polygons did not correspond to true houses. The second process eliminated 64% of such mis-recognized polygons. The failure of extraction in the first process was mainly due to the disconnection of internal contours by inserted characters or by noise. The

Fig. 6 A result of house-polygon extraction

Table 1 Processing time for raster-to-vector conversion

	By MPU	By accelerator
Connected component extraction	754 sec	18.5 sec
Thinning	190 sec	12.6 sec
Angle calculation	286 sec	21.6 sec

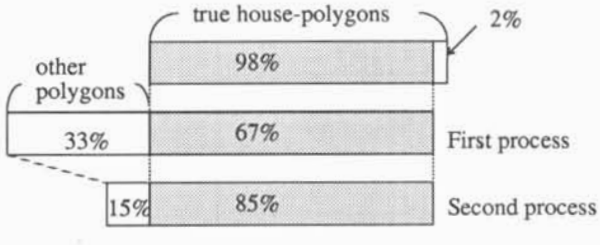

Fig. 7 Extraction accuracy

reason for over extraction of the second process was that the constraints for the recognition were a little weak.

6. Conclusions

A new technique for raster-to-vector conversion, which efficiently uses only contour tracking operations, has been proposed, and an accelerator for the operation has been developed in order to speed up raster-to-vector conversion.

By using the accelerator, reduction of about 90% in raster-to-vector conversion time was obtained from the experimental results of automatic extraction of polygons which represent houses from map images.

The developed accelerator can be widely applied to line-drawing processing in such fields from digitization and vectorization to feature extraction for drawing understanding.

References:

[1] K.Ramchandran: "Cording Method for Vector Representation of Engineer Drawings", in Proc. IEEE 68, pp. 813-817 (1980).

[2] M.Ejiri et al.: "Automatic Recognition of Design Drawings and Maps", in 7th ICPR, pp.1296-1305 (1984).

[3] M.T.Musavi et al.: "A Vision Based Method to Automatic Map Processing", in Pattern Recognition, vol.21, no.4, pp. 319-326,(1988).

[4] A.Okazaki et al.: "An Automatic Circuit Diagram Reader with Loop-structure-based Symbol Recognition", in IEEE Trans. on PAMI, vol. 10, no.3, pp. 331-340, (1988).

[5] C.M.Williams: "An Efficient Algorithm for Piecewise Linear Approximation of Planar Curves", in Proc. Comput. Vision, Graphics, Image Processing, vol. 8, pp.286-293 (1978).

[6] K.Wall and P.E.Danielsson: "A Fast Sequential Method for Polygonal Approximation of Digitized Curves", in Proc. Comput. Vision, Graphics, Image Processing, vol. 28, pp. 220-227 (1984).

[7] A.Rosenfeld and E.Johnston: "Angle Detection on Digital Curves", in IEEE Trans. Comput., vol.C-22, (1973).

[8] A.Rosenfeld and A.C.Kak: "Digital Picture Processing", Academic Press, New York, (1982).

[9] C.Arcelli: "Pattern Thinning by Contour Tracing", in Proc. Comput. Vision, Graphics, Image Processing, vol.17, pp. 130-144, (1981).

[10] T.Pavlidis: "An Asynchronous Thinning Algorithm",in Proc. Comput. Vision, Graphics, Image Processing, vol.20, pp.133-157, (1982).

[11] I.Sobel: "Neighborhood Coding of Binary Images for Fast Contour Following and General Binary Array Processing",in Proc. Comput. Vision, Graphics, Image Processing, vol. 8, pp.127-135 (1978).