オーディオビジュアル複合情報処理 27－13
(1999. 12. 16)

実時間で多角形を抽出・分類するアルゴリズム

ゲイツ ジョン　　長谷山 美紀　　北島 秀夫

北海道大学工学研究科
〒060-8628 札幌市北区北１３条西８丁目
Tel: (011) 706-7259
E-mail: john@meme.hokudai.ac.jp

あらまし　この論文は高精度に実時間で多角形を抽出と分類するアルゴリズムを示す。このアルゴリズムは複雑な入力画像から凸形、凹形両方の多角形を抽出することできる。このアルゴリズムは三角形を五つの種類に分類することできる。四辺形を六つの種類に分類することできる。アルゴリズムが 450 MHz のペンティアムII プロセッサでたくさんの 256 × 256 の８ビットを実験した。その実験の平均のフレーム率は一秒で６９フレーム以上である。このアルゴリズムの高精度を証明するために実験の結果を示す。

キーワード　パターン認識、多角形抽出、多角形分類

A Real-time Polygon Extraction and Classification Algorithm

John GATES　　Miki HASEYAMA　　Hideo KITAJIMA

School of Engineering, Hokkaido University
Kita-13, Nishi-8, Kita-ku, Sapporo-shi, 060-8628, Japan.
Tel: (011) 706-7259
E-mail: john@meme.hokudai.ac.jp

Abstract　This paper presents a high-accuracy real-time polygon extraction and classification algorithm. The algorithm is capable of extracting both convex and concave polygons from complex images. The algorithm can classify triangles into five distinct classes and can classify quadrilaterals into six distinct classes. The algorithm was tested with a variety of natural and synthetic 256 × 256 grayscale images and an average frame rate of more than 69 frames / second was obtained using a 450 MHz. Pentium II Processor. Experimental results are presented which demonstrate the high-accuracy of the algorithm.

key words　Pattern recognition, polygon extraction, polygon classification

1. INTRODUCTION

The extraction of lines from digital images is one of the fundamental problems in the field of pattern recognition [1]. Many solutions to this problem have been proposed [2-8]. However, the extracted line information provides very little information in itself to the user. What is needed is for the computer to connect these lines into more complex objects such as polygons. In this area, however, very little research has been performed.

Some researchers have been able to extract polygons by first extracting the lines by using the well-known Hough transform [9]. This method can extract polygons from simple binary images, but in complex natural images the Hough transform cannot accurately extract the line features. Also, the Hough transform is a very slow algorithm and cannot extract line features in real-time. The Hough transform is two orders of magnitude slower than the line extraction algorithm used in this paper. To improve the speed of extracting polygons, dedicated integrated circuits have been proposed [10]. Although this method greatly increases the speed, it is an inflexible and costly solution. A neural network approach has been proposed [11] that increases the accuracy of the polygon extraction algorithm at the cost of speed.

Although, these present methods can extract polygons, a new method is needed to extract the polygons in practical real-time applications. The three most important requirements for the extraction algorithm are accuracy, speed and cost. All of these conditions are met in the proposed algorithm.

2. ALGORITHM DESCRIPTION

2.1 The HART Line-feature Extraction Algorithm

The fundamental element of any polygon extraction algorithm is the line-feature extraction algorithm. The line-feature extraction algorithm used in this paper is a high-accuracy real-time (HART) line-feature extraction algorithm [12]. This algorithm is fundamentally different from other line-feature extraction algorithms in that it does not use pre-processing to detect edge pixels. Conventional algorithms use pre-processing to reduce the effect of blurred edge transitions that are common in natural images. The HART algorithm processes the natural image directly, thus maintaining the pixel intensity information. This intensity information is used by the algorithm to quickly and accurately extract the line-features.

Of all the conventional algorithms, the one which has the greatest potential for real-time applications is chain code [7]. The HART algorithm has many similarities with chain code but it also has many fundamental differences. The fundamental differences between the methods is based on the principles of chain code [6], which are:

(1) at most two basic directions are present, and these can differ only by unity, modulo eight;

(2) one of these two directions always occurs singly;

(3) successive occurrences of the single direction are as uniformly spaced as possible.

This is where the HART algorithm is superior to chain code. In the HART algorithm the directional variation permitted is governed by two principles only. These principles are:

(1) the direction of movement must always be advancing or staying at the same distance from the starting position;

(2) the error must always be below a given threshold. This error parameter forces the algorithm to closely follow the contour of the edge, thus curved lines can be accurately approximated as straight lines.

The first principle prevents the algorithm from descending back on itself. The second principle prevents the algorithm from following curves that are not straight lines. The advantage of this over chain code becomes obvious when processing natural images. In this situation lines are rarely perfect and will need to be approximated as straight lines. By using the error as the limiting factor the algorithm can follow the line shape exactly until its end is reached and then it can approximate the best fit for the natural line by using the least-squares estimators [13].

After the algorithm has extracted the line information, a post-processing algorithm is used to connect and thin extraneous line segments. This reduces the number of lines entering the polygon extraction algorithm and thus reduces the computation time. To demonstrate how the polygon extraction algorithm works the simple figure shown in Fig. 1 is used as the input to the HART algorithm. The extracted lines and nodes have been numbered and are shown in Fig. 2.

2.2 Polygon Extraction Algorithm

The polygon extraction process resembles the way the human visual system operates. In the human visual system, individual elements are assembled into larger features and then these features are assembled into even larger and more complex objects. In a similar way the polygon extraction algorithm first extracts the line-feature information and then finds the nodes where these lines intersect. Then it uses this information to

form the polygons. Finally the polygons are classified into various categories based on a mathematical definition.

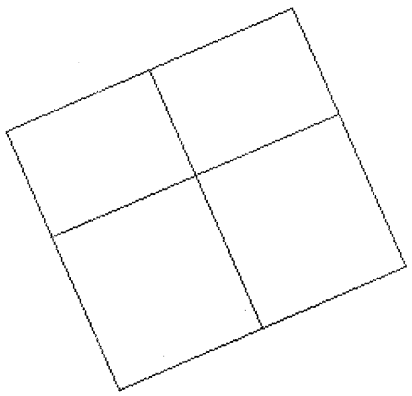


Figure 1. Input image.

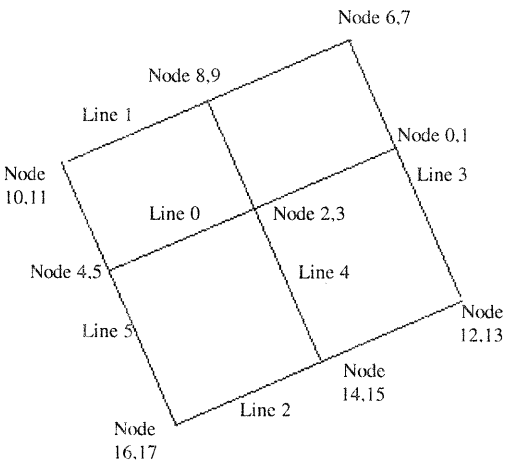


Figure 2. Extracted lines with line and node numbers.

To find the nodes where the lines intersect requires each of the extracted lines to be compared with all the other lines. This requires approximately $N^2/2$ comparisons. Where N is the number of extracted lines. Fortunately the y-range can be tested with only a single integer comparison and the x-range can be tested with only two integer comparisons. Thus almost all of the non-intersecting line pairs are eliminated using at most three integer comparisons. The remaining candidates are tested for intersection using equations (1) and (2).

$$x = \frac{b_2 - b_1}{m_1 - m_2} \tag{1}$$

$$y = m_1 x + b_1 \tag{2}$$

where m is the slope of the line and b is the intercept that was calculated by the HART algorithm. Small line breaks can be compensated for by allowing the length of the line segments to extend slightly beyond their actual length.

As each node is found, an ordered linked list is created. This list consists of the node number, the x and y coordinates of the intersecting lines, the intersecting line's number and a pointer to the next node. Two node numbers are needed to represent each intersection point, as individual node numbers are used for each of the two intersecting lines. These node numbers are chosen to be sequential. An example of the node list generated for lines zero and three shown in Fig. 2 is shown in Table 1.

Table 1. Node list for lines zero and three.

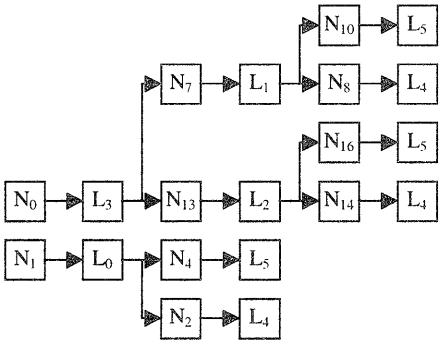
Line Zero				
Node	X	Y	Line	Pointer
4	37	104	5	2
2	123	141	4	0
0	207	177	3	-1
Line Three				
13	246	86	2	1
1	207	177	0	7
7	179	241	1	-1

Once all the lines have been tested for intersection, the algorithm uses a tree structure to connect the nodes into polygons. The tree begins with a pair of node numbers called the root nodes. From the root node numbers the corresponding intersecting lines can be found from the list. These two lines are the root lines. The next layer of the tree consists of the nodes and their corresponding lines which lie on the root lines and are higher in value than the root nodes. The final layer in the tree is calculated in the same manner. However, the third layer is only required for the top branch of the tree. To extract quadrilaterals, the lines in the third layer of the upper branch are compared with the lines in the second layer of the bottom branch. If a match occurs then a quadrilateral has been found and the node numbers are reported. If the line number in the third layer of the upper branch matches the bottom root line then a triangle has been found. This scanning process is easily extendable to N-sided polygons by simply increasing the number of layers in the tree. This approach also allows for the extraction of both convex and concave polygons. The node tree for nodes zero and one, for the example given in Fig. 2 is shown in Fig. 3.

2.3 Polygon Classification

Once all the polygons have been detected another algorithm is used to classify the triangles and

quadrilaterals into various geometric classes. The algorithm has five different triangle classes and six different quadrilateral classes.



Rec. 1: N0, N7, N10, N4
Rec. 2: N0, N7, N8, N2
Rec. 3: N0, N13, N16, N4
Rec. 4: N0, N13, N14, N2

Figure 3. Node tree. N represents the node number and L represents the line number.

All of the triangles can be classified by using angle information only. The first triangle class is the obtuse triangle. This class is found by testing for an angle greater than 90°. Right triangles are found by testing for an angle of 90°. If the triangle has two angles that are the same, and the angle is 60°, then the triangle is an equilateral triangle. If the angle is not 60° then the triangle is an isosceles triangle. Any triangle that does not fall into one of the above categories is considered to be a general acute triangle.

To classify the quadrilaterals the slope and line length information is required. The algorithm first tests to see if the slopes of the opposite sides are parallel. If this occurs then the quadrilateral is either a square, rectangle or general parallelogram. If the slopes of the lines are expressed as angles, and the difference between the slopes of two adjacent lines is 90° and the lengths of the adjacent sides are the same then the quadrilateral is a square. If the difference in the slopes is 90° and the lengths of the adjacent sides are not the same, then the quadrilateral is a rectangle. If the difference in the slopes is not 90°, then the quadrilateral is a parallelogram.

If the slopes of only two of the sides of the quadrilateral are the same then it is a trapezoid. If the adjacent sides of quadrilateral have the same length then it is a rhombus. If the quadrilateral does not fit into any of the above classes, it is classified as a general quadrilateral.

3. EXPERIMENTS

The image shown in Fig. 4 shows all of the triangles that the algorithm can classify. The classified triangle information generated by the algorithm is shown in Table 2. In Fig. 4 the point 0,0 is in the bottom left-hand corner, the x-axis proceeds from left to right and the y-axis proceeds from the bottom of the image to the top.

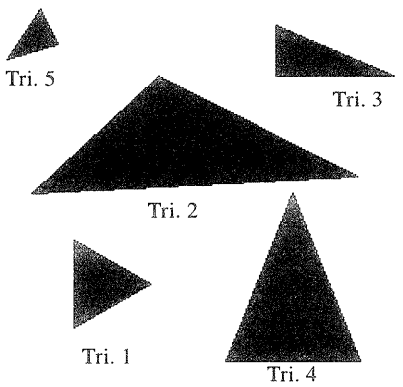


Figure 4. Input image for triangle classification.

Table 2. Triangle classification results.

Triangle	Type	Node 1	Node 2	Node 3
1	Equilateral	97,76	50,49	50,103
2	Obtuse	222,139	24,129	101,200
3	Right	245,199	170,199	170,231
4	Isosceles	221,29	180,131	140,29
5	Gen. Acute	41,219	30,241	9,209

The image shown in Fig. 5 contains all the quadrilaterals that the algorithm can classify.

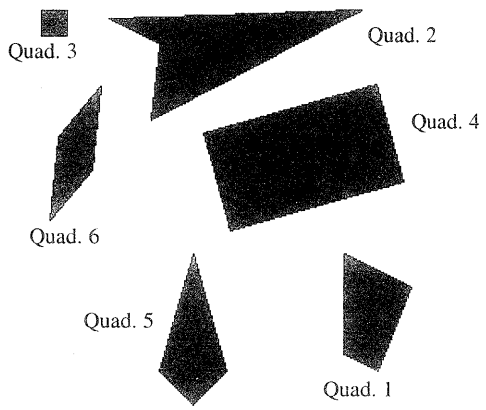


Figure 5. Input image for quadrilateral classification.

Table 3. Quadrilateral classification results.

	Type	Node 1	Node 2	Node 3	Node 4
1	Trapezoid	190,41	190,101	232,80	211,29
2	Gen. Quad.	51,240	202,245	75,179	80,225
3	Square	26,245	26,229	10,229	10,245
4	Rectangle	210,201	227,143	122,113	106,172
5	Rhombus	80,30	100,102	121,30	100,9
6	Parallelogram	41,150	15,118	20,170	46,201

The images shown in Figs. 1, 4 and 5 are simple binary images. Binary images can be processed by conventional algorithms with little difficulty. To show that the proposed algorithm is superior to the conventional algorithms, the image shown in Fig. 1 was corrupted with random noise and is shown in Fig. 6. The image was tested with the polygon extraction algorithm and the results are shown in Fig. 7. From Fig 7 it can be seen that the polygon extraction algorithm perfectly extracts the polygons from the noisy input image. The classified polygon data is shown in Table 4.

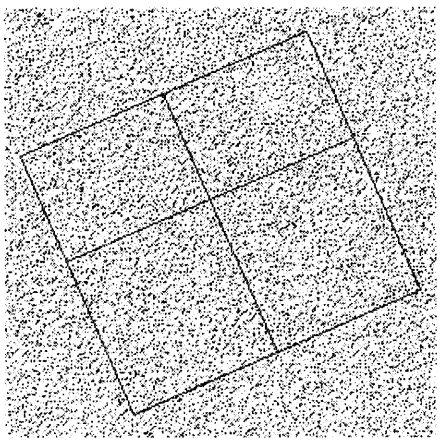


Figure 6. Corrupted image.

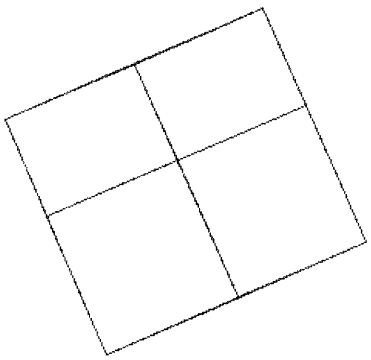


Figure 7. Image extracted from Fig. 6.

Table 4. Classification results for Fig. 6.

	Type	Node 1	Node 2	Node 3	Node 4
1	Rectangle	95,204	162,51	76,13	10,168
2	Rectangle	95,204	162,51	246,87	179,241
3	Rectangle	95,204	123,141	37,104	10,168
4	Rectangle	95,204	123,141	207,177	179,241
5	Rectangle	179,241	246,87	76,13	10,168
6	Rectangle	179,241	207,177	37,104	10,168
7	Rectangle	162,51	123,141	37,104	76,13
8	Rectangle	162,51	123,141	207,177	246,87
9	Rectangle	246,87	207,177	37,104	76,13

As can be seen from Table 4, nine rectangles have been extracted from Fig. 6. This result is correct, because there are four small interior rectangles, four rectangles composed of two interior rectangles and the large exterior rectangle. The algorithm extracts all the rectangles without preference for a certain class. This allows the algorithm to be used in many different applications.

Although the algorithm performed well on the corrupted synthetic image shown in Fig. 6. The algorithm should also be tested on a complex natural image. The image chosen for this test is shown in Fig. 8. The extracted polygons are shown in Fig. 9. It can be seen from the recovered image that the algorithm performs adequately on complex natural images.

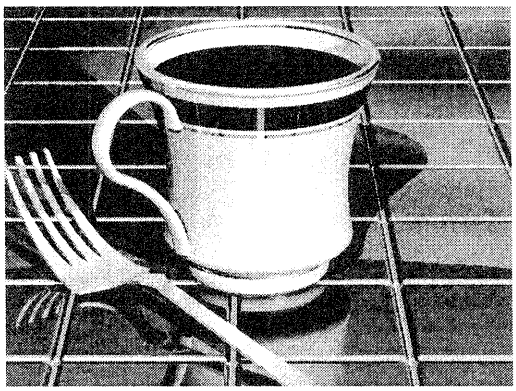


Figure 8. Natural Image.

To test the speed of the algorithm several natural and synthetic images were tested. These images were all standardized 256 × 256 grayscale images. Five synthetic images were tested. These images are Figs. 4, 5 and 6, a scanned blueprint and a circuit schematic. The four natural images tested are, Lena, the tea cup shown in Fig. 8, a boat and a computer motherboard. The frame rates are shown in Table 5.

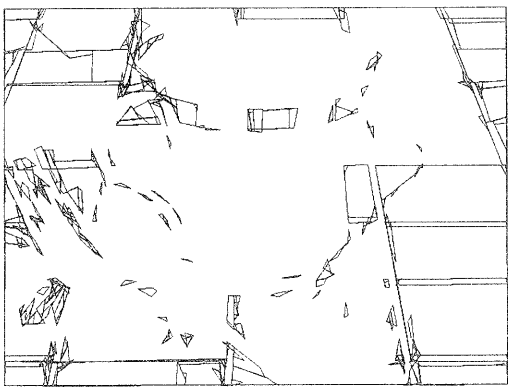


Figure 9. Polygons extracted from Fig. 7.

Table 2. Frame-rate Experiments

Image	Frame-rate (frames/sec.)
Figure 4	113
Figure 5	110
Figure 6	46
Blueprint	62
Schematic	52
Cup	59
Lena	72
Motherboard	51
Boat	64
Average	69

4. CONCLUSION

This paper has presented a high-accuracy real-time polygon extraction and classification algorithm. The algorithm is capable of extracting both convex and concave polygons from complex natural and synthetic images. The algorithm can classify triangles into five different classes and can classify quadrilaterals into six different classes. The algorithm was tested on a variety of 256 × 256 grayscale images and an average frame rate of more than 69 frames/second was obtained using a 450 MHz. Pentium II Processor. The algorithm has many useful applications such as identifying tumors in medical applications, increasing the accuracy of robotic manufacturing equipment, automatic quality control and as a building block for real-time complex pattern recognition systems.

5. REFERENCES

[1] J.W. Lee and I.S. Kweon, Extraction of line features in a noisy image, *Pattern Recognition*, vol. 30, no. 10, pp. 1651-1660, 1997.

[2] D. H. Ballard, Generalizing the Hough transform to detect arbitrary shapes, *Pattern Recognition*, vol. 13, no. 2, pp. 111-122, 1981.

[3] R. C. Gonzalez and P. Wintz, *Digital Image Processing. Second Edition.* Addison-Wesley, Reading Massachusetts, 1987.

[4] S. Y. Yuen and C.H. Ma, An investigation of the nature of parameterization for the Hough transform, *Pattern Recognition,* vol. 30, no. 6, pp. 1009-1040, 1997.

[5] D. H. Ballard and C. M. Brown, *Computer Vision.* Prentice-Hall, Englewood Cliffs, New Jersey, 1982.

[6] H. Freeman, Boundary encoding and processing, B. S. Lipkin and A. Rosenfeld, eds, *Picture Processing and Psychopictorics*, pp. 241-266. Academic, New York, 1970.

[7] J. Yuan and C. Y. Suen, An optimal O(n) algorithm for identifying line segments from a sequence of chain codes, *Pattern Recognition,* vol. 28, no. 5, pp. 635-646, 1995.

[8] J. W. Gates, M. Haseyama, H. Kitajima, A real-time line extraction method, *IEEE International Symposium on Circuits and Systems'99,* vol. IV, pp. 68-71, 1999.

[9] Rosenfeld A., and Weiss I. "A convex polygon is determined by its Hough transform.", *Pattern recognition letters*. v 16 n 3 pp. 305-307, 1995.

[10]Sastry R., and Ranganathan N. "PMAC: A Polygon Matching Chip." *Int. journal of pattern recognition and artificial intelligence* v 9 n 2 363-370, 1995.

[11]Mitzias D. A., and Mertzios B. G. "Shape recognition with a neural classifier based on a fast polygon approximation technique." *Pattern recognition*. v 27 n 5 627-635, 1994.

[12]Gates J., Haseyama M., and Kitajima H. "High-accuracy, real-time (HART) line-feature extraction algorithm." submitted for publication.

[13]E. R. Dougherty, *Probability and Statistics for the Engineering, Computing, and Physical Sciences.* Prentice Hall, Englewood Cliffs, New Jersey, 1990.